GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Blockchains in Public Administration

A RADIUS on Blockchain Framework for Public Administration

Dissertation
for attaining the doctoral degree
of Natural Sciences

submitted to the Faculty of Computer Science and Mathematics
of the Johann Wolfgang Goehte-University
in Frankfurt am Main, Germany

by

Zeki Nejat Philipp Konstantin Lang
born in Offenbach am Main, Germany

Frankfurt am Main 2023
(D 30)

accepted by the Faculty of Computer Science and Mathematics of the
Johann Wolfgang Goethe University as a dissertation.

Dean:	Prof. Dr. Martin Möller
Expert assessors:	Prof. Dr. Udo Kebschull Prof. Dr. Volker Lindenstruth
Date of disputation:	

Acknowledgements

At this point, I would like to take the opportunity to thank all those who accompanied and supported me during my doctoral studies. Without their valuable help, not only the completion of this thesis but also its beginning would have been unthinkable. Far be it from me to establish a hierarchy in my thanks, which is why the order is not meant to represent a valuation. Each person mentioned here has contributed to my progress in his or her own distinctive way, whether in terms of content, form, or constant motivation.

First and foremost, I would like to express my deep gratitude to my doctoral supervisor, Prof. Dr. Udo Kebschull. It was thanks to his initiative that the research cooperation could be established in the first place. His support for numerous projects, such as the joint delivery of a lecture on blockchains at Goethe University, was indispensable. Likewise, I am indebted to the HZD for its valuable support. Markus Beckmann, in particular, has proven to be a reliable contact person to facilitate communication with HZD, our cooperation partner. His contribution to obtaining hardware and other resources for my research was invaluable. I would also like to thank the HZD as a whole, which expressed its interest and created the necessary conditions for me to conduct my research.

My special thanks go to Diyar Takak. Through our ongoing dialogue, motivation remained high at all times, and he proved to be an invaluable support both in teaching courses and in proofreading this thesis. Furthermore, I would like to single out Nathalie Zarbock and Lisa Bruder. The thesis I had the honor of supervising not only contributed to their own academic success but also provided valuable insights for this thesis. My thanks also extend to Cedric Bakhshi and Marvin Eifler. Although their work was under the auspices of HZD, I was pleased to be able to support them. Their contribution to building the research environment was instrumental and invaluable. I would also like to thank the Goethe University Writing Center for allowing me to answer any questions regarding writing.

Special thanks are due to Gil Sobol and Niklas Vogel. Their constructive feedback has been instrumental in improving the quality of this thesis, both formally and structurally. In this context, I would also like to mention Haya Shulman. Her patience and valuable feedback were invaluable.

A heartfelt thanks go to my family. The constant support of my parents and sister has always been there just when I needed it most. My special thanks also go to my wife and son. They enabled me to gain mental distance from academic work and ensured that I could leave the often quoted ’ivory tower’. In difficult times, they gave me the strength and motivation to stick with it and continue the work. You all have a firm place in my heart.

Abstract

The emergence of blockchain technology has generated a great deal of attention, as reflected in numerous scientific and journalistic articles. However, the implementation of blockchain for public administrations in Germany has encountered a setback owing to unsuccessful initiatives. Initial enthusiasm was followed by disillusionment. Nevertheless, technology continues to evolve. This paper examines whether the use of a blockchain can still optimize the processes of public administrations. Not only the failed projects are analyzed, but also more current applications of the technology and their potential relevance for the administration, especially in the state of Hesse.

To answer if blockchains are promising to administrations, a Design Science Research (DSR) research approach is chosen. The DSR method is a research-based approach that aims to create new and innovative solutions to real-world problems through the development and evaluation of artifacts such as models, methods, or prototypes. For this work, the implementation of a framework to realize an Authentication, Authorization and Accounting (AAA) system on the blockchain was identified as profitable. The framework aims to implement the aforementioned AAA tasks using a blockchain. The Remote Authentication Dial-In User Service (RADIUS) protocol has been identified as a potential protocol of the AAA system. The goal is to create a way to implement the system either entirely on a blockchain or as a hybrid system. Various blockchain technologies will be considered. Suitable for development, the framework AAA-me is named.

The development of AAA-me has shown that the desired framework for implementing RADIUS on the blockchain is possible in various degrees of implementation. Previous work mostly relied on full development. Additionally, it has been shown that AAA-me can be used to perform hybrid integration at different implementation levels. This makes AAA-me stand out from the few hybrid previous approaches.

Furthermore, AAA-me was investigated in different laboratory environments. This was to determine the expected resilience against Single Point of Failure (SPOF). The results of the lab investigation indicated that a RADIUS system on top of a blockchain can provide benefits in terms of security and performance. In the lab environment, times were measured within which a series of authorization requests were processed. In addition, it was illustrated how a RADIUS system implemented using blockchain can protect itself against Man-in-the-Middle (MITM) attacks.

Finally, in collaboration with the Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD), another test lab demonstrated how a RADIUS system on the blockchain can integrate with the existing IT systems of the German state of Hesse. Based on these findings, this work reevaluated the applicability of blockchain technology for public administration processes.

The work has thus shown that the use of a blockchain can still be purposeful. However, it has also been shown that an implementation can bring many problems with it. The small number of blockchain developers and engineers also poses the risk of finding people to develop and maintain a system. In addition, one faces the problem of determining an architecture now that will be applied to many projects in the future. However, each project can, in turn, have an impact on the choice of architecture. Once one has solved this problem and a blockchain infrastructure is available, it can be established quickly and be more SPOF resistant, for example, for Public Key Infrastructure (PKI) systems.

AAA-me was only applied in lab and test environments. As a result, no real data ran over its own infrastructure. This allowed the necessary flexibility for development. However, system-related properties could appear in real situations that are not detectable here in this way. Furthermore, the initial stage of AAA-me's development is still in its infancy. Many manual adjustments need to be made in order for this to integrate with an existing RADIUS system. Also, no system security effort in and of itself has been carried out in the lab environments. Thus, vulnerabilities can quickly open up on web servers due to misconfigurations and missing updates. For the above reasons, productive use should be discouraged unless major developments are carried out.

Contents

List of Figures

List of Tables

List of Codes

Akronymes

2PA	Two-Party Authentication	41
3PA	Three-Party Authentication	41
AAA	Authentication, Authorization and Accounting	162
ACL	Access Control List	56
AP	Access Point	58
API	Application Programming Interface	193
Aura	Authority Round	203
BFT	Byzantine Fault Tolerance	14
BIP	Bitcoin Improvement Proposals	207
CA	Certification Authority	150
CDN	Content Delivery Network	140
CHAP	Challenge-Handshake Authentication Protocol	49
CRL	Certification Revocation List	150
DAG	Directed Acyclic Graph	28
DAO	Decentralized Autonomous Organization	135
DHT	Distributed Hash Table	9
DRF	Django REST Framework	88
DL-	Distributed Ledger-	151
DLT	Distributed Ledger Technology	161
DNS	Domain Name System	149
DoS	Denial of Service	194
DPoS	Delegated Proof-of-Stake	55
dPKI	decentralized Public Key Infrastructure	149
EAP	Extensible Authentication Protocol	118
EAP-GTC	EAP Generic Token Card	50
EAP-MD5	EAP Message Digest 5	50
EAP-TTLS	EAP-Tunneled Transport Layer Security	50
ECDSA	Elliptic Curve Digital Signature Algorithm	50
EIP	Ethereum Improvement Proposals	207
EVM	Ethereum Virtual Machine	94

Chapter 1

Introduction

The emergence of blockchain technology has received major public attention over recent years and has been extensively studied by the research community [256]. While providing promising results, existing research focuses on applications and development in the context of finance or the realization of economic business cases. But even if there is a minority in research publication, technical improvements for technical systems like Internet of Things (IoT) and digital identities have been shown.
This work aims to extend the scientific understanding beyond the context of financial and economic gain by investigating if and how blockchain can be applied to a proliferating problem for increasingly digitalized societies; how can public administration utilize technology to improve processing efficiency and protect itself from modern threads by centralized malicious threat actors? Blockchain provides a potential answer to this question by offering a decentralized and secure solution for resilient public administration. Investigating how blockchain can be applied to administrative processes, including a comprehensive analysis of inherent problems in the technology, implementation hurdles, and potential security benefits, will be a focus of this research.
To illustrate the researched concepts, a Design Science Research (DSR) concept is employed. DSR is a research methodology used to create and evaluate IT artifacts, aiming to solve identified organizational problems. In this context, an artifact can be software, technology or methods. An implementation of the Remote Authentication Dial-In User Service (RADIUS) protocol on a blockchain will be developed, demonstrating how such a system can be seamlessly integrated into the existing IT systems of public administrations, and the potential challenges that may arise. The research project was developed in collaboration with the Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD). As the IT service provider for the state of Hesse, HZD is an essential player in the digital transformation of administrative processes. In a world where technology is becoming increasingly complex, it is constantly looking for innovative solutions to ensure efficiency, security, and reliability.
In the period of arrangements of cooperation, the blockchain hype subsided. During this phase, which was triggered by numerous projects through the Initial Coin Offering (ICO), the initial impression was that a blockchain could be utilized and proved beneficial in numerous areas. However, it was the failure of the many projects that provided the motivation for exploring the research question. Is a Blockchain capable of enhancing the efficiency and security of public administration processes?
Public administrations are highly complex in their structures and operations. Since this is a work in computer science, the focus should be on a technical implementation. Therefore, sociological and economic considerations are only briefly outlined, and further research is required.
The Blockchain technology garnered significant attention in 2017, owing to the ICO hype that resulted in a flurry of projects. One of the promises was, and still is, "decentralization". The ability to implement processes without a Trusted Third Party (TTP) sounds promising. Whether this property can be held, and whether this property is needed, is what this thesis explores.
Much less known, but more widely used, is the so-called RADIUS protocol. It allows tasks to be performed

across different networks, which can be grouped under the acronym Authentication, Authorization and Accounting (AAA). A widely known example among university staff and students is the eduroam network. This access to the WLAN infrastructure makes it possible to be registered at one university, but to be able to authenticate with this registration at all universities participating in eduroam. The forwarding of authentication requests is realized via a TTP. So the question of implementing such a system with a decentralized technology, like blockchains, arises immediately.

It is noteworthy that certain terminologies are not clearly defined or possess ambiguous definitions. One of these terminologies is derived from the development of blockchains. Especially, the technologies developed in the first generations try to realize the consensus on the construction of chained data blocks. However, it turns out that such solutions have problems with scaling. This results in further new technologies that break these chains and implement further data structures, leading to the more general term Distributed Ledger Technology (DLT). However, the literature more often examines the more explicit blockchain technology and subsumes the generalization with blockchain. Therefore, this thesis either exclusively uses the term blockchain, or DLT is explicitly mentioned for clarification.
In the context of blockchain programming, the term smart contract comes up. However, the term is misleading. The already existing understanding of the term contract suggests that this technology functions similar to classical contracts. In fact, smart contracts are smaller programs that are executed on the corresponding blockchain. The Hyperledger Foundation is therefore endeavoring to use the term chaincode in order to resolve this confusion. In this work, the term blockchain program is often used because not all DLT systems use chained elements in blocks as their data structure like Bitcoin (BTC). Nevertheless, this term is also difficult in the context of implementation in 5 chapter, as it complicates the distinction between ordinary and blockchain programs. Therefore, both terms, smart contract and blockchain programs, are used depending on the situation.

This thesis has the following structure and outline: In chapter 2 you will find basic explanations of the basic technologies used. This encompasses the aforementioned blockchains and the RADIUS protocol. Other aspects, such as authentication or Public Key Infrastructures (PKIs), are also included.
Subsequently, the state of the art is shown in chapter 3. This encompasses the most recent advancements in authentication, blockchains, and a comprehensive literature review on the utilization of blockchains for AAA tasks.
Chapter 4 is divided into three main sections. Firstly, the implementation of AAA tasks on a blockchain is shown in general terms, with problems and solutions. These insights are then required for the other two areas. The second section delineates and conceptualizes the architecture of AAA-me, a framework for implementing the RADIUS system on a blockchain. Thus, a comprehensive porting process, as well as a hybrid implementation, can be accomplished. The third section pertains to the potential integrations into the HZD.
In 5 chapter, the conception is implemented. For this purpose, experiments are conducted to test important blockchain promises. The results of the findings are then discussed and evaluated in chapter 6. The thesis ends with an outlook on possible further developments.

Motivation

As mentioned, the HZD is interested in technological innovations and solutions that will make its own processes more efficient. As this organization is an IT service provider for public administrations, it serves as a suitable platform for conducting research in this particular field. From the perspective of public administration, it is interesting to investigate whether a decentralized architecture can support or benefit federal systems in their digitization. Furthermore, another project, namely the Hesse WLAN, has demonstrated itself to be an intriguing interface for the implementation of a RADIUS system. Therefore, it is obvious that

these research areas should be combined.
Further ideas emerge from the considerations so far. For instance, the inquiry arises whether the implementation of the RADIUS system on a blockchain can be augmented to include the capability of implementing a Single Sign-On (SSO). The methodologies used to implement AAA would be used to do so. Furthermore, the question arises whether an implementation of a SSO could even be possible on a physical level by integrating door locks and palm vein scanners. The authentication of a person would then only have to be done once, possibly via a blockchain. Authorized access would then be granted not only to the person in question, but also to the hardware the person is carrying.
Finally, the question of decentralization is still one that needs to be addressed. How does decentralization work, and is it desirable? It is recommended that the advantages and disadvantages, as well as integration possibilities, be presented and discussed in relation to the example described above.

Problem Statement

The motivation suggests that there could be many possible applications being investigated here. But what are the issues that need to be addressed?
On the one hand, there are the public administrations. In Germany, these are federally structured, with the primary intended to protect democracy. However, during development and research, a difficulty becomes apparent. Numerous approaches are repeatedly investigated on each state. Due to the autonomy of the 16 states, or even the respective municipalities, there is often a highly heterogeneous system architecture landscape. Agreements between each other are very complicated and not always possible because of the number of possible locations.
In contrast to the structure of the Federal Republic of Germany just described, eduroam has a hierarchical structure. The eduroam project is realized using a RADIUS protocol. A TTP exists therefore for forwarding authority. There is also a TTP in other applications of the protocol. But this authority may be compromised if the central authority fails. It is irrelevant whether the failure was unintentional or a deliberate act by an unsavory entity. This centralized exposure is also described as Single Point of Failure (SPOF). If a decentralized architecture can solve the problem, it should be considered.
There are many types of blockchain technologies. Each solution has its strengths and weaknesses, according to its specifications. However, there are also issues that are currently inferior to all current blockchain systems, although there are different systems for different applications. These include the transparency, oracle, and scaling issues. These problems will be explained later. Possible solutions and implications should be discussed and highlighted.

Research Objectives

This thesis addresses the question of the possible implications of blockchain technology for public administration. This is accomplished by illustrating and implementing an exemplary blockchain application using a RADIUS implementation. From the described motivations and problems, further questions can be derived, which are relevant and should be investigated in advance of the research. Consequently, it is possible to encounter overlapping of responses to the questions.

- **What are the capabilities of a blockchain, and what are the limitations?** Many blockchain projects have failed. This is indicated by a look at the number of cryptocurrencies with their price trends. Especially the many failed Blockchain projects should be an indication of a questionable use of the technology. What are blockchains capable of, and where do their reach limitations?

- **How can a blockchain be incorporated into existing systems?** The applications presented in this thesis can mostly be implemented in hybrid form. This also makes sense, since an existing system would not have to be completely replaced. The possible approach of embedding a blockchain system in an existing system is to be examined and answered.

- **What are the reasons for implementing a blockchain in public administrations?** If the technical feasibility has been shown, the question remains, why go to the effort. This thesis aims to answer that question.
- **Is blockchain technology capable of enhancing democracy?** The aforementioned federal structure in Germany entails a distribution of power and a separation of powers. Is a decentralized IT system capable of supporting such federated structures?
- **What are the prerequisites for a successful blockchain implementation?** Here, too, the failed projects show that blockchains are difficult to implement. Once the appropriate points of a meaningful deployment are identified, what are the possible complications? How can a project be successfully implemented?

Chapter 2

Fundamentals

In this chapter, we will discuss the foundations of this work. The chapter provides the theoretical background to understand the subsequent parts of the thesis. The utilized technologies, specifically blockchain and Remote Authentication Dial-In User Service (RADIUS), are fundamental components of the conceptual framework (as exemplified in chapter 4 Methodology and Conception on page 57). Furthermore, cryptography is essential, motivating its introduction in this chapter. If the reader feels confident in the technologies, he can also go directly to the chapter 3 Current Trends and Development on page 49.

2.1 Cryptography

This section will only briefly discuss the most commonly used cryptographic methods in blockchain technologies. Since blockchain procedures differ significantly and therefore also frequently use different cryptographic systems, this section cannot go into full depth. Furthermore, cryptography is a specialized, intricate discipline. To explain this in detail is beyond the scope of this thesis. Nevertheless, the reader should be able to understand the methods used and identify the respective properties, which will ultimately lead to the properties of blockchain and Distributed Ledger Technologies (DLTs). Further information can be found in the literature, specifically [22] was used for this excerpt. Many of the points mentioned in this section can be found there. The literature mentioned is much more extensive and is recommended for interested readers.

The most common goal of cryptography is to make texts and messages available only to selected people and to ensure their integrity. At the same time, the procedure should be as simple and inexpensive as possible (e.g., in the size of the encrypted file). Cryptographic signature procedures can be used to realize digital signatures.
The military in particular has developed great interest in cryptographic systems, especially in the past. If the communications of army commanders could be intercepted or even manipulated, they would have a great advantage over the enemy. With the introduction of computers and their networking, cryptography has played an increasingly important role in the civilian sector as well. You can control specific web pages on the Internet (HTTP, HTTPS, etc.), use online banking, or reach the authorities digitally. The latter example is currently only possible in limited circumstances in Germany.
The possibilities of cryptography are very diverse. Even if we only use a small excerpt from this science, the general properties of cryptographic systems can be derived for our use cases. These properties include the fact that they are easy to use in digital media and that current methods can theoretically be made arbitrarily secure. In most cases, it will be sufficient to increase the key lengths to adjust the security level. For the specific case of potential quantum computers, one is not conclusively sure in this regard whether adjusting the key lengths will be sufficient. More precisely, the cost of the necessary key lengths could have a negative impact on operability, since simple encryption and decryption steps would require too much time and resources.

There are other requirements for cryptographic systems. However, these requirements are very specific and must be identified with the respective use cases. For example, the encryption of simple texts requires different requirements than a signature would require. In summary, these requirements are according to Beuteslpacher [22]:

confidentiality, authenticity, integrity, non-repudiation, anonymity

Confidentiality means that the messages are processed by the sender in such a way that they can only be read by the receiver with the secret key. All other entities cannot read this message. The authenticity ensures that the message from the sender can be traced to its origin. This means that the message can be identified that it was actually sent by the sender. With the integrity, one understands the (unnoticed) unchangeability of data. It is not feasible for an encrypted or signed file to have undergone any modifications unnoticed during the transfer process. Given the non-repudiation, it follows that the evidence of message authenticity can be transmitted. Therefore, the recipient of a message would be able to prove the authenticity of the message from the sender. This requirement is more robust than the requirement solely based on message authenticity. The last requirement of anonymity states that the sender as well as the message itself remain unknown to third parties.

As mentioned above, this section provides a generic introduction to cryptography and will instead focus on three main areas, as these are used for blockchains and DLTs. These are symmetric and asymmetric encryption, digital signatures, and cryptographic hash functions. I highlighted the digital signatures because they find substantial use in DLTs. In fact, however, this is an application of asymmetric encryption. Encryption of information is actually not required for the use of a blockchain or DLT infrastructure. However, some projects implement this in the form of virtual tunnels among communicating parties [117].

2.1.1 Symmetric and Asymmetric Cryptographic Methods

Throughout history, attempts were made to encrypt texts so that they could only be read by the recipient. The primary objective was to keep the procedures secret. The Caesar cipher, for instance, demonstrated the necessity of keeping the principle of encryption, the encryption, and decryption procedure, secret. After understanding this principle, it was possible to decrypt the ciphertext with minimal effort. The fundamental principle of the Caesar cipher is founded on the alternation of letters. This was determined once and given to the recipient as a decryption key in advance. According to Beutelspacher [22], other systems that relied on the secrecy of the procedures were the A5 and Comp128 algorithms, which were used in the mobile communications industry. The first procedure was revealed in a secret publication, while the second was revealed through a reverse engineering effort, rendering it unusable. It has been demonstrated that this type of cryptographic secret does not provide the necessary level of security, as it is solely based on obscurity.
Many of the systems mentioned are referred to as symmetric methods because a key generation and subsequent exchange must occur in advance. If a fraudster obtains this key, they will also be able to decode the encrypted messages. Furthermore, the key exchange must be securely guaranteed.
The need for a high degree of security led to the idea of keeping secret, not the procedures themselves, but the keys used (is also called Kerkhoff's principle [22]). The Rivest–Shamir–Adleman (RSA)-procedure is the first one based on this principle, published in 1978 [216]. This was also the breakthrough for numerous cryptography-related systems that were only able to emerge subsequent to this. The ability to exchange keys on public way was essential for RSA. The sender gets a public key from the receiver to encrypt a message, and nobody else is able to decrypt the ciphertext. Not even with the public key. This type of method is called asymmetric cryptography.
The majority of the asymmetric methods that were developed rely on factorization of large numbers as a fundamental principle, specifically, taking the e-th root in $\mathbf{Z}_n$, as documented in [237, S.30-32]. This includes the famous RSA method. It is assumed that factoring a number into its prime elements is very hard

to compute, but that computing the product is effortless. It should be noted that any number N can be represented by a unique prime decomposition $\prod_i^n p_i^{e_i} = N$ [49, p. 21]. Calculating the product of two primes is easy to follow, and simple to implement. Since the proof of the difficulty of the inversion has not yet been given, one must assume that an algorithm could be found which deciphers our common methods. Another mathematical principle that finds application in cryptographic systems is the prime number decomposition, or the discrete logarithm. Here, verifying $e^x = y$ is straightforward given x. However, computing $\log e^x = y$ is sufficiently hard in so-called $\mathcal{F}_p$ number fields. We thus compute in number sets besides the trivial $\mathbf{N}$, $\mathbf{Z}$, $\mathbf{Q}$ or even $\mathbf{R}$ fields. Thus, we can make the discrete logarithm more difficult to compute. If the computation is more challenging, it is feasible to accomplish cryptographic tasks with smaller keys. Another group, to which one assigns a heavy computability, are the so-called elliptic curves. In such groups occur the numbers which satisfy the equation $y^2 = x^3 + ax + b$. There are further requirements, like $4a^3 + 27b^2 \neq 0$. These numbers have the advantage that addition and multiplication can be defined on it. Thus, the discrete logarithm can also be calculated on it. Due to the higher degree of difficulty, the keys here are shorter while the security level remains the same. [22, chap. 12]

Table 2.1: Comparison of symmetric and asymmetric crpytographic methods by [22, S. 114]

Symm. Algorithmen	**vs.**	**Asymm. Algorithmen**
very many	Quantity	very few
It is highly probable.	Security	It is highly probable.
usually well	Performance	bad
yes	Previous key exchange necessary	no
no	Possibility of digital signature	yes
Encryption	typical areas of application	Signatures Key exchange

Having gained knowledge about the symmetric and asymmetric methods, it is imperative to examine their characteristics. In the table 2.1 Comparison of symmetric and asymmetric crpytographic methods we can easily understand the comparison of both categories.
From a first glance, one can see that there are many algorithms for the symmetric methods. Finding asymmetric methods is much more difficult. It has already been mentioned that many procedures rely on the complexity of the prime factorization or discrete logarithm, and this security cannot yet be proven. Hence, one can only assume a good level of security here based on empirical values. The speed and size of the key are significantly higher with asymmetric methods than with symmetric methods. Therefore, the asymmetric procedure is usually used to exchange the keys for the symmetric procedures. In order to benefit from the performant conversion of large amounts of data, but also to come the advantages of asymmetric encryption in the possibility of the ad hoc key exchange. Ultimately, the final line of the table containing the application areas is derived from the aforementioned properties.

The principle of symmetric encryption states that a common key is used by both the sender and the receiver to secure and decrypt a message. The disadvantage lies in the poor distribution of a newly created key, since the receiver needs it for decryption. If the sender and receiver do not have a secure connection for the exchange, the handover is a major problem.
With the asymmetric method, no prior key exchange is necessary. The mathematical functions used create a public key and a secret key, with the public key being derived from the secret key. The reverse, to calculate the secret key from the public key, is not possible. The term not feasible signifies that there is no algorithm available for the calculation that is more efficient than relying on trial and error (commonly referred to as

brute force). The public key can now be transmitted from the receiver to the sender via all transmission paths. The sender takes this public key to encrypt the message. And although the procedure itself and the public key are known to a possible attacker, the attacker has no way (besides brute-force attacks) to get at the decrypted message. The receiver, however, can use the secret key to decode it and receives the message from the sender [22, Chapter 9].
The asymmetric method just described is for encrypting messages. The special feature of many asymmetric procedures is that digital signatures can also be created with the same algorithm with only slight adaptation. If a sender wants to sign a message, he must again create a public key from a secret key. At this point, however, the sender will now use the secret key for a message m to create the digital signature. Then, the sender will pass the message with the signature to the receiver (or the receiver will take the information) and can verify the signature by evaluating the information at hand. The presented RSA method is such an asymmetric method, which can be used for encryption as well as for signatures.
In practice, asymmetric and symmetric encryption methods are often used together. Asymmetric encryption is often used to provide a secure key exchange for symmetric encryption. Thus, a symmetric key can be securely exchanged between parties by encrypting it with the public key of the receiving party. Then, this symmetric key is used for faster data communication. This combines the speed advantages of symmetric encryption with the increased security of asymmetric encryption. [22]

2.1.2 Cryptografic Hash Functions

First of all, cryptographic functions are mathematical functions with further properties. A function has a preimage and an image set, and the function describes a procedure for assigning elements from the preimage set to the image set. A good example of such a function is time. It takes 24 hours for the earth to rotate around itself once and thus complete one day. On analog clocks, however, there are only 12 hours. Thus a function can be written, with which starting from 13 o'clock on the available numbers is assigned.
Hash functions are now functions that map data of any size into a fixed size. This implies that the numbers < 10 would have to be represented with a leading zero, for example, 03. In general, we can assume that this property makes the preimage larger than the image set. This causes collisions, with which one finds two elements from the preimage, which are assigned to the image. For instance, the clock displays the times of 1 o'clock noon and morning to 01. Therefore, a hash function is almost never injective, with the exception of constructed or uncommon instances. [22, p. 7-8]
In order for a hash function to become a cryptographic hash function, we require the following properties. [65]:

- PREIMAGE RESISTANCE: For a given element from preimage x, it is easy to compute the image using the hash function $h : h(x) = y$. However, the inverse does not hold, so it is not feasible to get from a given y to the preimage x.
- SECOND PREIMAGE RESISTANCE: Given x from preimage, we cannot find y from preimage with $h(x) = h(y)$.
- COLLISION RESISTANCE: One does not find x, y from the preimage with the property $h(x) = h(y)$.

The preimage resistance indicates that the clock illustration is not a cryptographic hash function. Because, in the case of the clock, there is a system for the function from which the preimages can be determined. In conclusion, cryptographic hash functions exhibit some degree of randomness. The distinction between second preimage resistance and collision resistance is the free choice of the variable x. Thus, collision resistance is the stronger condition.
A cryptographic hash function is considered to be broken if one of its properties can be shown to no longer hold. Even though, a random finding of a collision does not necessarily mean that one has discovered or

developed a system. Further vulnerabilities and attacks on cryptographic hash functions are discussed in the paper Cheval et al. [65] demonstrated. In the paper [189] methods are shown that investigate findings for current functions from already broken hash functions.

Integrity checking is often found in the application of cryptographic hash functions. If I desire to ensure that the received data remains unchanged, I compute a hash value over the data and compare it with a provided value. If both hash values are the same, I can assume there is no tampering. In blockchains, the functions are also utilized in what are referred to as Merkle trees. These are also known simply as hash trees. This procedure generates hash values for the information that must be safeguarded to preserve its integrity. Afterward, one takes two of these hash values and forms another hash value of them. This process is continued until a root is obtained. This procedure results in the formation of a data structure, which is a binary tree, and its primary branch is also known as the Merkle root. The data structure provides the advantage of being able to prove the existence and integrity of data. In Landau notation, one can be performed here at space and time of $O(\log n)$. Further optimizations are possible, according to [245].

Besides Merkle trees, there is another use of the hash functions. The fixed length of the hash values coupled with the collision resistance feature are well-suited for the creation of key-value pairs with them. This allows for data integrity to be combined with a location access. A data structure can be formed using these pairs, also known as a Distributed Hash Table (DHT). This is used in distributed Peer-to-Peer (P2P) networks. For this purpose, they provide an efficient way to locate and provide data without relying on a central instance. This description is similar to the description of a blockchain. The explanation of a blockchain is provided in the next section; however, the differences should already be clarified here. This is because a DHT focuses more on efficiency in the integrity and retrieval of data. The focus of a blockchain is on the immutability and transparency of the data. This immutability is enabled by consensus. Thus, a blockchain does not provide data from one node to other nodes, but rather the same data on all nodes. In addition, the data is in different data structures. The DHTs have the aforementioned key-value pairs while blockchains rely on chained blocks with transactions inside.

2.1.3 Zero-Knowledge Proofs

In cryptography, the concept of Zero-Knowledge Proof (ZKP) was developed in the 1980s. In a ZKP, it is possible to prove knowledge to a counterparty without having to reveal that knowledge itself. The concept is intriguing and has applications in various areas such as authentication, privacy, and more.
An application example here could be proving a password. Without having to disclose the password itself, this concept can be used to prove ownership. Thinking further, this can also be applied to secret keys from asymmetric cryptographic methods.
In a ZKP, the prover and the verifier interact in a dialog that is similar in some respects to a commit-and-reveal-like procedure. A commit-and-reveal procedure consists of two components: the commit, in which a message is processed along with a random number called nonce or challenge, and the verify, in which the message is checked against the commit. In some cases, the nonce is passed as a challenge by the verifier, and the prover accepts that challenge.
Nonetheless, it is imperative to emphasize that in the context of ZKP, the actual information, including the password itself, is not utilized. In particular, if one wants to prove possession of a password, the password must not be used directly in the proof, or it would be revealed to the verifier. Instead, a special ZKP protocol is used that is based on the password without directly revealing it.
These special protocols are part of what distinguishes ZKP from more general challenge-response techniques. There are a variety of techniques that can be used to create ZKP evidence, and not all of them use the commit-and-reveal method or even a challenge-response structure.

The ZKP principle remains the same: the prover convinces the verifier of the truth of a statement without revealing anything more than the truth of that statement. Thus, the prover's "knowledge" remains protected, while the verifier still gets the confirmation he needs. This balance between secrecy and verification makes ZKP a powerful tool in modern cryptography. [124]
There are three main properties that are subject to a ZKP:

- Completeness: If the statement is true and both parties are honest, the verifier will always be convinced by the prover.
- Soundness: If the statement is false, the verifier will not be convinced, assuming the verifier follows the evidence protocol correctly.
- zero-knowledge: If the statement is true, the verifier learns nothing about the statement except its truth, even if he is dishonest and tries to extract additional information from the proof.

There are several types of protocols that serve different requirements. In interactive and non-interactive protocols, a first characterization takes place. In interactive protocols, there is a regular exchange between verifier and prover. E.g. to transfer the challenges. In contrast to this are the non-interactive systems. In such, only one message is exchanged. This feature is useful for applications where storage space and bandwidth are limited, such as blockchain systems. The drawback is that they require a "trusted setup" that can raise potential security concerns.
Among blockchain systems, there are a few types in use. These include the zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) and the zk-STARKs (Zero-Knowledge Scalable Transparent Arguments of Knowledge). The zk-SNARKs are a form of ZKP that are particularly known for their compactness and efficiency. They require only a small, constant size for the proof, regardless of the length of the input. SNARKs are "non-interactive." Unlike zk-SNARKs, zk-STARKs do not require a "trusted setup", making them more transparent and potentially more secure. They also provide post-quantum security, meaning they remain secure even as powerful quantum computers are developed. However, the trade-offs for these advantages are larger proof sizes and higher verification costs, which can make them less efficient than zk-SNARKs.
Moreover, there are the Sigma protocols. These are a class of ZKPs that are often used in challenge-response formats. They are interactive, meaning that prover and verifier communicate back and forth in a series of messages. Sigma protocols are useful in many applications and provide strong security guarantees. They can be converted to non-interactive protocols by using a fiat Shamir transformation. [19, 124, 257]

2.2 Blockchain Technologies

Since the introduction of Bitcoin with the paper "Bitcoin: A peer-to-peer electronic cash system" [192], the technology has developed significantly. In 2009, an implementation and launch of an instance of the presented technology was conducted, as documented in [33]. In addition to the further developments of Bitcoin, from which some so-called forks (more on this later) have also emerged, new blockchain technologies have developed. These efforts aim to enhance scalability by adopting a novel data structure that departs from linear and chained blocks. Hence, a novel generalizing term has been identified in the literature, namely Distributed Ledger Technology (DLT). Through generalization, DLT also encompasses blockchains. Nevertheless, the term blockchain continues to be found in the literature. On the one hand because the term is more familiar. But on the other hand, also because the other technologies are very similar to blockchains in their basic principles. Therefore, the literature uses the term blockchain in a way that integrates DLTs. In this thesis, we also proceed in this way.
A blockchain can be understood as a list of data blocks that are chained within itself. The chaining is created by using the ID of the previous block in the current block itself. The necessary information for a block is

encoded into a cryptographic hash function, and the resultant output is utilized as the identification number of the new block. This will then be used in the following block. See chapter 2.1.2 Cryptografic Hash Functions for a detailed explanation of cryptographic hash functions. This concatenation creates a data structure that allows minimal changes to be identified quickly. However, this comes at the expense of the changeability of data. If a person wishes to modify or eliminate a date, the hash value of the block will be modified in accordance with the aforementioned date. Subsequently, the hash values of all following blocks change. The earlier the date is in the blockchain, the more blocks have to be recalculated. This is usually not a difficult task for commercially available computers. Hash values can be calculated in fractions of seconds. Therefore, this data structure is still provided with a consensus in the application of blockchains. The objectives of the consensus are to ensure data integrity within the own data structure and to synchronize data within the distributed network. No node in the network may have processed the data. This would lead to inconsistency and endanger the network. Consensus is thus of great importance, especially when comparing different nodes. Bitcoin (BTC) uses with Proof-of-Work (PoW) a relatively expensive procedure. This repeatedly leads to the argument that Bitcoin consumes too much energy [80, 18]. However, there are also arguments in favor of the energy requirement and consensus methods that consume significantly less energy. Some famous consensus systems are later elaborated in more detail. For our data, which is currently contained within the blockchain data structure, it is imperative that consensus is adhered to when making any modifications. So in addition to matching the hash values, there are other factors that are important that can make it difficult or even impossible to make a quick change. We have a data structure that is referred to as an append-only structure.

In Satoshi Nakamoto's paper [192], BTC is described as a P2P system. This refers to a distributed system in which individual nodes are interconnected. Communication protocols are used to exchange data among the nodes. One of these protocols is the Gossip protocol. This is a communication protocol used in distributed systems to spread information, similar to the spread of gossip in a human community. The basic idea is simple yet effective. For example, in blockchain systems, transactions, and blocks are disseminated among each other. Information about node addresses is also incumbent on this protocol. A new initiated node can thus request connected node addresses from an existing node. With the aforementioned consensus, it is possible to create a shared dataset among nodes. As a distributed system, blockchain systems are also subject to the same problems as other distributed systems. A good illustration and description is provided by the CAP theorem. This explains the lack of feasibility of a system realizing all three properties from the name, **C**onsistency, **A**vailability, and **P**artition tolerance. There is always a trade-off between these three properties, such that one property is not realizable. A formal proof was given in 2002 for this assertion [120].

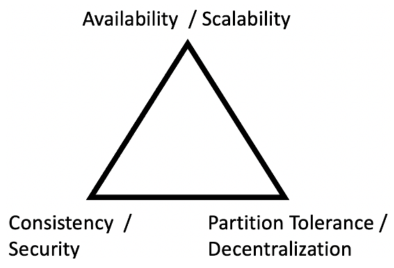

Availability / Scalability
Consistency / Security
Partition Tolerance / Decentralization

Figure 2.1: Representation of CAP Theorem with Blockchain Trilemma by [46]

The term 'Consistency' refers to the parallel data processing on the system nodes. It is desired here that all nodes have the current data state, and a change must be adapted almost immediately everywhere. Availability is defined as the access of a client to a distributed system. A request from a client must be answered within an acceptable timeframe. The Partition Tolerance indicates that a failure of certain nodes must have minimal to no impact on the overall stability of the distributed system. [46]

In addition to P2P systems, there are other architectures of distributed systems. For instance, the conventional client-server architecture enables a user to submit requests to a server, which will then provide them with the required services. In order to enhance the utilization of servers while simultaneously reducing costs, it is feasible to implement representation, logic, and data storage on distinct systems. Due to their three-tier architecture, these are also referred to as a tripartition. By employing of cloud processes, this limitation to three instances can be extended. Hence, the so-called n-tier architectures have emerged. When the workload

is high, it is possible to activate additional systems, whereas when the workload is low, subsystems can be disabled. By utilizing this approach, the distributed system can be regulated and scaled as required, resulting in a reduction in expenses while still guaranteeing a high level of availability.
Examples of distributed systems at the CAP theorem are ([259]):

1. **[AP]** Domain Name System (DNS) or Distributed Hash Table (DHT)
2. **[CA]** Relationales Datenbank Management System (RDBMS)
3. **[CP]** Banking Application

It is important to have a high availability at all times for DNS systems. In the sense of response time and failsafe. This comes at the expense of consistency. DNS propagation may take several hours [141]. In RDBMS clusters, data availability and integrity is mainly important, as these systems are usually run on well networked and equipped servers. Accordingly, failover (partition tolerance) is less important. For banking applications, however, availability is the least important feature. On the other hand, the correctness of the data is very essential, even in the event of failures. Therefore, availability can be omitted here for a gain in consistency and partition tolerance.
Brewer's CAP thesis gives us a representation of the realization difficulties of distributed systems. Since most blockchain systems are P2P systems, the aforementioned problems apply here as well. Nevertheless, there is an adaptation of the theorem to blockchain and DLT systems with the so-called Buterin Trilemma or Blockchain Trilemma. This involves slightly adapting the properties for which the same rules should apply, as in Brewer's CAP theorem. While Availability describes the response time of a request, the counterpart in DLT is scalability. Scalability is the maximum number of transactions that can be realized per second on average. A transaction in a blockchain signifies the modification of a date. In the case of BTC, this can be the transfer of tokens. In the case of systems that are suitable for blockchain programs, the entire programs can be initialized or operated with a transaction.
When security is mentioned in the context of blockchains, it usually means data integrity. How easy is it to manipulate data in the system, possibly even afterward? This is where the consensus procedures take effect and determine the strength of security of a blockchain. Security is comparable with consistency, since the integrity of the data within the system nodes is also considered here.
Partition tolerance is the last property in CAP. For historical reasons, the process of securing the system in blockchains is also known as decentralization. This refers to the number of operating nodes in the network. Since all nodes must perform the same tasks, it is sufficient to evaluate resilience by evaluating the absolute number. In contrast to three-tier or even n-tier architectures, where nodes are assigned different tasks, bottlenecks could arise as a result. The term is, however, not well-chosen here. Blockchain communities are currently undergoing a lot of debate and disagreement regarding which systems are decentral and what exactly that implies. There is no definitive clarification of the term at this time, as a significant number of factors associated with it remain unidentified and would not serve any purpose in comprehending this particular section. Please note that the decentral properties are gradually gaining in importance depending on their perspective. This means that the respective systems can partially fulfill the properties. Also, there are projects that claim to have solved the CAP problem for blockchains. Although there has been an attempt made to improve a desired property, it cannot be realized at the expense of the other properties. As an example, we can mention the lightning network [208] in Bitcoin. The proposed approach is intended to enhance scaling by bundling bilateral transactions and subsequently executing them on the Bitcoin blockchain. Nevertheless, this solution comes at the cost of decentralization and security, as BTCs consensus only partially applies to individual transactions. A detailed description of Lightning can be found in chapter 2.2.1 Bitcoin Blockchain. Also, with the update from PoW to Proof-of-Stake (PoS) in Ethereum, scaling through sharding is improved. But again, sacrifices are made on decentralization and security. In chapter 6.3 is a detailed explanation and discussion of decentralized systems.

Table 2.2: Properties of Blockchains and DLTs

NR	Characteristic	Characteristics and Comment
1	Reading Access	Public / Private (Transparency)
2	Write Access	Permissioned / Permissionless
3	Consensus Method	Scale, degree of decentralization, Security (Persistence of the Network / Consistency)
4	Smart Contracts	Depending on Smart Contract - definition (Turing completeness?)
5	EVM Compatible	With Ethereum as a global identification for Smart Contracts

With CAP-Thereom, we have addressed the architecture of distributed systems. These are greatly influenced by the consensus process and provide comparability between projects. Furthermore, we deliberated on the topic of distributed systems. Blockchain technologies, however, are described as decentralized. The distinction can be roughly described as distribution on a physical level and decentralization on an organizational level. Later in the dissertation, we will go into further detail on the clarification of the term. For a better comparison and assessment of blockchains, let's look at other characteristics.

It is important for a characterization of blockchains and DLTs to distinguish between public and private blockchain instances. We need to distinguish between reading (both public and private) and writing (permissioned and permissionless). These two dimensions can be represented by a matrix, whose cells can be used to allocate the many projects. In the table 2.3 Reading vs. Writing Dimensions for Blockchain Technologies on page 14, one can view some blockchain projects on the two dimensions described. The table provides a good overview of the different systems in comparison. However, this representation focuses solely on the two characteristics mentioned. A non-exhaustive, but in this work used, overview of the properties of blockchain systems provides the table 2.2.

Bitcoin (BTC) and Ethereum (ETH) are two well-known representatives of public blockchain systems. But even if a public decentralized instance exists, it is still possible to use the software to start your own private instance. This can prove to be advantageous for both research and teaching purposes. However, it may be advantageous to establish a consortium of partially trustworthy representatives who are unwilling to be influenced by the price fluctuations and development challenges of the public system. This method allows for quicker implementation and distribution of your updates.

Such private instances are mainly relied on by projects in the Hyperledger Fabric space [7], as well as ETH. This increases the likelihood of the technology being adopted in the economy, as it can be constructed with its own blockchain systems, thereby enhancing its visibility and testing. Nevertheless, there is a continuing strong dependency on the public entities, as the fundamental further development of the software takes place in the public developer groups. One can choose to carry the development or decline if the public software undergoes development, with the consequence of not being able to carry any further compatible developments from the open-source developer group.

A key component of blockchain technologies is the consensus process used. Many other properties, such as scalability, can be derived directly from this, which we have already seen with the Blockchain Trilemma. Furthermore, consensus still has an impact on the ability of a desired system to be decentralized. However, it is difficult to quantify the notion of decentral or assign it to a metric. There is research that has attempted to quantify decentralization [178]. This work focused more on mining distribution and ignored other aspects (number of nodes, open-source developers, number of ways to participate in discussions, mining pools, number of different clients, etc.). It is this absence of a metric that renders it challenging to debate whether or to what extent a blockchain system should or can be preferred over an existing centrally structured system.

The consensus method in BTC is PoW. It bears direct responsibility for the elevated energy requirements and sluggish transaction processing, commonly referred to as scaling. The procedure was originally developed for

Table 2.3: Reading vs. Writing Dimensions for Blockchain Technologies

Source: According to [253]

	Public	**Private**
Permissionless	Bitcoin, Ethereum (1.0)	Quorum
Permissioned	Hyperledger Indy (Sovrin), EOS, Ripple, Eth (2.0)	Hyperledger Fabric, Distributed DB

spam prevention, but it is now used for probabilistic consensus in BTC. In this process, someone is randomly selected to write to the blockchain. The randomness is contingent upon the hardware utilized in the mining process. Importantly, this randomness cannot be manipulated without providing new or better hardware. Given the hardware requirement, this randomness provides a system that makes deliberate manipulation much more difficult and expensive. A potential attacker would have to invest similar power compared to the current network, so presumably the potential return will be less than the necessary cost. [197, ch. 2] The method has a poor reputation because of its poor energy balance and high hardware requirements. As a result, this system is also said to be responsible for a high percentage of electronic waste [81].

Besides the PoW there are other consensus procedures. First and foremost is the Proof-of-Stake (PoS). Ethereum (ETH) carried out the change in September 2022. The main change with PoS compared to PoW is that not the computing power, but shares of the system's own tokens, provide for the random write authorization. This makes it easy to see that the disadvantage of energy consumption can be quickly eliminated [75]. However, new problems come with this system, such as the significantly increased complexity of the new mining and lack of experience in the execution of consensus. [8]
Other methods based on a consensus approach attempt to emphasize chance of writing in the ledger in other aspects. The Proof-of-Authority (PoA) focuses on trust in entities. In contrast to the systems already mentioned, here the entities involved have to authenticate themselves. Their reputation becomes important because if someone attempts to harm the blockchain system, they will be kicked off the voter list and will no longer have permission to write to the blockchain. The system is much more energy-efficient, but it comes at a cost of anonymity. Very closely related to PoA systems are the Byzantine Fault Tolerance (BFT) systems used. Here, redundant BFT (rBFT) and practical BFT (pBFT) are widely used representatives. In the systems used, authorities are elected and can be deselected based on their performance. Consensus is achieved here after a predetermined delegation confirms all transactions with their respective signatures. With Federated Byzantine Agreement (FBA) another type of BFT-systems is mentioned, which, however, allow other conversion possibilities into read and write authorizations. There is more information on consensus methods later in this thesis. In the table 2.4 Reading vs. Writing Dimensions for Consensus Algorithm on page 15, one can look at some consensus procedures in the access dimensions considered earlier. Here the consensus procedures are assigned for their suitability in the respective dimensions.

Other distinguishing characteristics of blockchain technologies include the capability to implement so-called "Turing-complete programs." These programs are also often described in the literature as "smart contracts," but they are not comparable to contracts in the classical sense. These smart contracts are mostly if-then relationships that cause a reaction to a pre-determined input. Nevertheless, since the programming languages are Turing-compliant, one can program arbitrary programs. Constraints are determined by the consensus process of the technology used. Essential is the implementation on all involved nodes, so that the large public networks can implement rather smaller programs only. The implementability is controlled by costs retained by the internally formed tokens or coins.

Table 2.4: Reading vs. Writing Dimensions for Consensus Algorithm

Source: [253]

	Public	**Private**
Permissionless	PoW, PoS	FBA, IPDB
Permissioned	PoS, PoA, BFT	BFT

Many newer projects attempt to compare themselves to the most famous blockchain with smart contract suitability, Ethereum. In Ethereum, smart contracts are implemented in the Ethereum Virtual Machine (EVM). Therefore, all major blockchains are eager to be able to run the already programmed contracts on their chain, or at least provide converters [58, 50]. Since this adoption capability of existing programs could be important for deployment, this feature is additionally investigated and is included in 2.2 Properties of Blockchains and DLTs. It is noteworthy that if an individual has initiated a private instance and developed programs on it, these can be transferred to a partially public blockchain, provided that the private instance is a blockchain capable of EVM. This could be exciting if, at the start of a project, it is not yet certain which blockchain instance may or can be used.
Blockchain technology still has a significant disadvantage with its data structures. Large amounts of data cannot be stored in a suitable manner. Furthermore, numerous initiatives are open to the public, resulting in a transparency issue, highlighting security concerns. Data must also be modifiable or deletable for this purpose. Two approaches try to tackle this issue. There are so-called Oracle services. These services enable the inserting of blockchain-agnostic data on the ledger. The second approach is a distributed data storage system called InterPlanetary File System (IPFS) and InterPlanetary Database (IPDB). Detailed explanations of those solutions will be provided later in this dissertation.

2.2.1 Bitcoin Blockchain

Satoshi Nakamoto's paper [192] published the first blockchain technology, known as Bitcoin. This technology was conceived as a disruptive application for banking. An instance of the implementation was delivered a few months after the paper was published, creating what is known as the Genesis Block [38]. From this point forward, all subsequent blocks are constructed sequentially. Bitcoin has been running ever since, despite some minor problems to be solved. It appears that the number of users is increasing, although the benefits and the deployment goal of the technology continue to be viewed very critically. The criticism is mainly directed against the consensus used, which is the Proof-of-Work (PoW) in Bitcoin.
This section is devoted to the technical structure of Bitcoin, linking to the section on blockchain (chapter 2.2 Blockchain Technologies). This connection will be important as all advancements in the technology, be it blockchains or even DLTs, will be analyzed in comparison to Bitcoin. As mentioned in the opening chapter on blockchain technology (2.2 Blockchain Technologies), its strength lies in its use in a trustless environment. In the following sections, we will explore how this has been technically implemented and what impact it may have on other systems. Furthermore, additional realizations of the distributed trustless system problem can be considered in subsequent chapters. It should also be noted here that Bitcoin is understood to be a first-generation blockchain, with further generations explained in subsequent chapters.

First, we consider the three central roles in BTC: the users, the miners, and the node operators. While the miners and the node operators are essential for the security of the system, the user, often referred to as the wallet, plays an important role in the adaptation. After all, if there are no users, the technology is useless. The designation of a user as a wallet derives from the wallet software through which the user interacts with the Bitcoin blockchain. Machine-driven access to the blockchain has also been considered, so the term wallet is understood in a broader sense.

Not to be confused is the double meaning of the term "wallet". On the one hand, it refers to the software that can be used to create and send direct transactions with the Bitcoin blockchain. On the other hand, it also stands for a pair of keys from asymmetric cryptography (2.1 Cryptography) used to create signatures. In most contexts, the meaning should be clear to the reader. When both terms are used contemporaneously, I will use the term wallet software to refer to the software and wallet to refer to the cryptographic keys.

Later in this chapter, the terms mining and node will be explained in detail. For now, the following description should suffice: A node is a computer that stores a blockchain and distributes this information to other nodes. The miner ensures that all nodes have the same state, with only minimal variations possible that adjust over time. Thus, all nodes in the network have the same state, and a wallet can retrieve the current data from any node.

The respective dependencies are thus clearly defined: The data resides on the nodes and is checked for equality by the miners. The user can then access any node with his wallet and carry out transactions. The node receives open transactions from the wallet, which the miner puts into blocks. After a matching block is found, the data is sent back to the node so that the user can receive confirmation of the successful transaction.

Transactions

If a user wants to send his Bitcoins, he uses a software, which is also called a wallet. This software creates the key pairs in the necessary format. With this key pair, it is able to create a transaction that realizes the sending in the network. The format of a transaction is coordinated throughout the network, and changes require approval from a majority of the network. This transaction is then sent to the nodes, where it is checked for correctness. Only then can the transaction be validated into a block by a miner. If a transaction does not fit for various reasons, it is not accepted by the node and thus not validated by a miner. However, it can also happen that the fee is set too low and a lot of transactions are created in the Bitcoin network. In the worst case, the own transaction is then discarded because the memory becomes full of transactions with higher fees that have a higher priority. The own transaction is then deleted without validation. The user still has the option to respond with a second transaction that contains a higher fee. Other information matches. One creates a second transaction from one's own wallet, so to speak. This is then also called Replace by Fee (RBF). The new transaction with higher fee can then be validated faster, and the old transaction is discarded as already spent. In doing so, the transaction fees do not go back to the sender to avoid a possible attack vector for Denial of Service (DoS) attacks.

A transaction is schematically described in the appendix B JSON Interpretation of a Bitcoin Transaction. There, a transaction is mapped into human-readable JSON format. However, before we go into the technical depths of a transaction, let's take another look at a transaction from a high-level perspective. There is an important concept here that needs to be clarified first. Bitcoin is a so-called transaction-based system. Here, there is no account balance like in account-based systems that has to be adjusted after transactions. A bank account at the bank is the classic example of an account-based system. There is a bank account with a current state. If a transaction takes place, the balance on this account is changed. In a transaction-based system, the respective transactions form an account balance, which must first be determined. To accomplish this, one looks at the history that the Bitcoin has already built up through transactions. This then allows the currently available Bitcoin to be determined.

If you look at the details of a transaction in the appendix, you will notice that input and output information are included (appendix B JSON Interpretation of a Bitcoin Transaction). This transaction contains one input and two output pieces of information. This is because a transaction necessitates the initial definition of a wallet address that already possesses bitcoin. This can then be used to send them to other wallets, provided that one is in possession of the secret keys. The output from the end of the transaction chain is also called the Unspent Transaction Output (UTXO). The outputs of all transaction ends are included. The UTXO are very exciting in that they sum up to indicate the current number of bitcoins available worldwide that can be sent. Lost keys and burned[1] Bitcoins are not included in this. If one has lost the key to his wallet, the bitcoins on this wallet referred to no longer accessible to anyone. This actually means that the bitcoins will remain unusable until the cryptography is broken and one can recalculate the private key. Accordingly, the number of Bitcoins ever available in the future is definitely less than 21 million[2] because people will lose their private keys occasionally, fail to organize a handover in time before passing away, or transfer bitcoins to wallets whose private keys are not known due to system or application errors.

The value of the bitcoins in the input must exceed the value in the output in total. A small positive difference remains, which then represents the fee of the transaction. So we have a transaction that wants to transfer bitcoin from one or more wallets to at least two or more wallets. The input wallets already own bitcoin in this process. If we continue the transactions, a tree is formed that represents the course of sending the bitcoin. This course can be checked until the origin is reached. The origin in bitcoin is the so-called Coinbase transaction. This is the transaction that rewards the miner for finding a block. The exact process of mining will be discussed later in chapter 2.2.2 Consensus Procedures.

A transaction must still be signed after it has been created with the wallet software according to a predefined scheme. There are currently several methods for doing this. What is essential for us at this point is that a signature procedure is used to pass off a transaction as its ownership. Much more, the proof of ownership of the input accounts is provided by these signatures. This is done thanks to the secret key. The signature, transaction, and public key are sent to the node for validation. With the information provided, the Node can verify that the transaction was actually created by the given sender. It is important to note that no personal information is available and that Bitcoin tries to be a largely anonymous system. Therefore, it can only verify that the owner(s) of the secret keys made the transaction. If the secret key is compromised, then in the event of theft, the thief can create a legitimate transaction from the relevant wallets.

Due to the tree described above, which is created by the transaction histories, it is not possible to speak of an anonymous system in Bitcoin. With sufficient effort, the identity of a wallet address can be identified, and the subsequent history can be traced. Therefore, Bitcoin is also referred to as a pseudonymous system. However, there are more and more approaches to further strengthen privacy. This can happen through so-called mixers, which combine several transactions from different actors and thus make the outputs no longer easily attributable. [197]

The lack of anonymity is echoed by other blockchain projects. Monero and Zcash are two projects implementing an anonymous blockchain on the main layer. In order to accomplish this, they employ distinct strategies. In Monero, so-called Pedersen Commitments are used to keep Monero Coins secret. In addition, extended Schnorr signatures are used to create extended ring signatures. Not only single key pairs, but whole vectors of key pairs, are used. [166] In Zcash, a so-called Zero-Knowledge Proof (ZKP) is used. Here, the possession of a piece of information can be proved without having to publish this information. A famous example here is proving that one is of legal age. A common method is to publish one's birthday. With a ZKP proof, one can now prove that one is of age without publishing the date of one's birthday. In the case of Zcash, the proofs are needed to provide proof of the private key and to keep the Zcash Coins amounts secret. [67]

Let's take a closer look at the keys of a wallet, as shown in the sample transaction in the appendix. There we find the generated signature in lines 14, 30, and 39, respectively. It should be noted that this is a simplified transaction; usually more complex transactions are created with multiple signatures required. These

[1]The consensus Proof-of-Burn (PoB) uses an address without a private key to burn bitcoins. This means that without the secret key, the respective bitcoins can no longer be accessed, and thus the bitcoins are counted as lost forever

[2]There is also the fact that due to rounding in the halving, the amount 21,000,000 Bitcoins can never be reached. The maximum number of bitcoins available can be calculated by the formula $\sum_{h}^{32} 210000 \frac{50}{2 \cdot h} \approx 209999.9769$. Where 32 is the number of upcoming "halvings", 50 is the first maximum reward for the miner (subsidy), and 210000 blocks are the halving from each other. [243]

Table 2.5: Verification process of a Bitcoin address (P2SH): This table is read from above. In each step, the change of the stack is mentioned in the respective column. In the first line, you see an empty stack. After first execution, the constants are pushed to the stack and removed from the Script. That is shown in line 2. An OP-Code with a number like OP_0 means that the number 0 is pushed to the stack. After all, the "OP_CHECKMULTISIG" function will be executed on the data on the stack. The result is shown in the stack column on line 3. This is a very short and easy example of Bitcoin Script.
Source: [122]

Step	**Stack**	**Script**	**Description**
1	Emtpy	OP_0<sig1><sig2>OP_2<pubKey1><pubKey2><pubKey3>OP_3 OP_CHECKMULTISIG	Only the sript-Sig is used
2	0<sig1><sig2>2<pubKey1><pubKey2><pubKey3>3	OP_CHECKMULTISIG	Constants are added to the stack.
3	true	empty	Signatures validated in the order of the keys in the script.

signatures are stored in a field called 'script'. This is because Bitcoin (BTC) transactions are executed using a custom scripting language known as Bitcoin Script, which is similar to the Forth programming language.
The language is left-right oriented and governs the processing of a stack, where the term 'stack' means the data storage according to the last-in-first-out principle. Data is stored sequentially in memory, but only the last element added can be read or modified again first. The changes are described by so-called OP commands, where a list of existing OP commands can be taken from [31].
Example executions of a command chain are given in the tables 2.5 Verification process of a Bitcoin address (P2SH) on page 18 and C.1 Verification process of an Bitcoin address (P2PKH) shown on page 201. In these examples, important methods were used, and the respective commands were executed. The individual explanations of the steps can be found in the table descriptions.
It is important to emphasize that Bitcoin's scripting language is not Turing-complete, unlike other blockchain projects such as Ethereum. Satoshi Nakamoto himself has not explicitly stated the reasons for this decision. Other developers interpret it to mean that the complexity would increase unnecessarily should Turing-completeness be achieved [171]. Since there is no way to undo errors in Bitcoin, the potential for errors should be reduced as much as possible.
Another drawback of Turing completeness is the so-called halting problem, a fundamental question in theoretical computer science. It examines whether an algorithm with arbitrary inputs can be examined for a possible ending, which Alan Turing turned out to be impossible. In Ethereum (ETH), this issue arises in the form of calculating gas charges. It is noteworthy that the actual fees of a smart contract cannot be precisely determined prior to execution, but rather, they can be estimated (as discussed in section 2.2.3 Ethereum and Blockchains 2.0). Consequently, the gas expense for a similar execution may fluctuate, and in rare instances, the initiator may send insufficient gas, resulting in the termination of the execution.
The one programming language functionality that is missing in bitcoin Script for Turing completeness is loops. This applies specifically to while loops, which isn't supported. In these instances, the program is provided with a termination sequence, and the code is executed until the termination sequence is reached. If this sequence does not occur, the program continues to run until it is manually aborted. There is no stopping of the program.

In contrast, there exist the for-loops, which provide an operation space with a beginning and end. The program runs through this operation space. The loop itself can be 'unrolled' by manual programming each step. This is doable since the iteration steps with the end are known. This can, however, escalate to a considerable extent, resulting in an overall increase in the memory requirement. A memory limit of a transaction is given by the block size. However, each byte of information must also be paid for with transaction fees. The larger a transaction, the more expensive it is. This gives an economic reason not to simulate for-loops. [228, 264]
To execute transactions on Bitcoin, the presented OP operations must be executed. These are defined in the script field of the transaction and are located in the output of a transaction. There is now a special OP command that has expanded the possible uses of Bitcoin with its introduction. This is the so-called OP_Return command. This command indicates that the transaction cannot be used further. If an amount is attached to the fee, this amount is irretrievably lost. One speaks here about burned. This is an integral part of the Proof-of-Burn (PoB) consensus procedure, in which coins are burned to be able to be selected as a write-authorized. Nonetheless, the command was not introduced for this particular purpose. Many messages and images were perpetuated on the Bitcoin Blockchain, thus increasing the UTXO database. To find a solution to this, the OP_Return command was introduced. This enabled the storage of data in the blockchain for an indefinite period without the necessity to maintain the respective transactions in the UTXO. If one desired to store data on the blockchain, they would create a transaction, send it with zero bitcoin, and then pay the fee to have the data stored on the blockchain. Based on this, various alternatives have been proposed for enhancing the functionality of the Bitcoin Blockchain. However, these ideas have always been very controversial, as for many the use of Bitcoin as a currency would be sufficient. Further applications would further endanger scalability and thus the entire project. [16]

The detailed explanation of how transactions are processed is important to this work in two areas. First, it indicates that data can be stored on a blockchain. In Bitcoin, there are currently 80 bytes of storage. This storage space could be used to store small certificates. These certificates could then be used in challenge-response processes for authentication. Here would be an initial interface for implementing a RADIUS protocol on the Bitcoin blockchain.
The second significant area is the opcodes. This makes it possible to write programs that define a routine of executions. In these programs, methods for processing the certificates described above could be implemented.
The two application areas show that a Turing-complete programming language is not absolutely necessary. This is one of the main arguments made by Bitcoin's proponents against Turing-complete blockchains like Ethereum. The lack of programmability is justified by increased security and lower storage requirements.
Thus, the question that arises from the discussions is whether a project has a need to execute programs on-chain. In contrast, checkpoints from the program could be stored as bitcoin transactions in the OP_Return field, or other outsourced processes could be used, such as the Lightning network.

Blocks

The transactions are compiled and subsequently interpreted as a block comprising additional meta information. It is stored in the form of a Merkle tree. The data structure enables quick verification of the existence and integrity of one of these transactions. One particular transaction stands out in each block. This is the so-called Coinbase transaction. This is responsible for paying out a portion of the reward. The miner enters the address of a wallet he controls in this field. A payout via Coinbase transaction is blocked for 100 blocks. Only after that, the miner is able to access and spend these bitcoin. This is built in as a protection against accidental forks so that the miner does not accidentally or intentionally double-spend through a new fork.
Further information is a so-called Difficulty, the Nonce. Both fields are required for mining. This will be explained in detail later. The block hash ID from the previous block, a version number, the Merkle root of

all transactions and a time stamp are all combined, creating a hash value. This hash value now forms the ID of the current block. The hash value does not require additional information, but those does provide further information for analysis. For example, each block is given its size and a transaction counter.

When talking about blocks in Bitcoin, a very historical event needs to be outlined. The 'Block Size War' in the Bitcoin community was a heated and controversial debate about the appropriate size of blocks in the Bitcoin blockchain. At the time, the original size of blocks was limited to 1 MB. Since each transaction takes up space in a block, this limit imposed a restriction on the number of transactions that could be processed per second. As Bitcoin grew, this limitation met with more and more resistance. Two solutions emerged from this problem. However, with these solutions, two camps also formed. One was the group that argued with increasing the block size. The larger blocks should allow for more transactions, which could lead to faster confirmation times and lower fees. The inability to reach consensus on block size eventually led to a hard fork of the Bitcoin blockchain in August 2017, creating Bitcoin Cash (BCH). Bitcoin Cash implemented a larger block size of 8 MB, while the original Bitcoin blockchain (BTC) stayed at 1 MB and later implemented SegWit. The Block Size War had a significant impact on the community, including a partial split and the emergence of several competing Bitcoin implementations. It also led to intense discussion about the governance and future of decentralized networks. [27]
The Block Size War highlights the complexity of decentralized decision-making and the difficulties that can arise when a community attempts to change fundamental technical aspects of a decentralized system. It also highlights the importance of consensus mechanisms and community participation in decentralized systems. [27]
The current block size is currently limited to 1 megabyte. This can be extended up to 4 MB of data with SegWit transactions. Data to witness a transaction for the inputs is excluded from the scope of measuring the size of the transaction. [159]

2.2.2 Consensus Procedures

As already noted, the consensus process is of particular importance in blockchain technologies. This is what makes it possible to use it in an environment that is almost trustless. It achieves this by ensuring an equal data state across all participating nodes in a distributed network. Even though those procedures for blockchains are significant and have only been around since 2008, scientists were interested in consensus methods very early on. Lamport's paper was published in 1978 and is considered to be a fundamental contribution to distributed system's theory [169]. The paper helped for the understanding of distributed systems. It not only provided the fundamental instrument of logical clocks, but also facilitated an appreciation of the significance of the sequential and coordinated execution of events in a distributed setting. Another publication by Lamport elucidates the issue of the Byzantine generals [170]. This problem describes the siege of the city of Byzantium by three generals. They must agree on a time to attack. The generals, however, are physically separated and can only communicate via messengers. The issue becomes complicated because some generals might be treacherous and send false information. The challenge is to reach a consensus, even if some members of the network are unreliable or hostile. In the paper, it was shown that the problem is intractable when out of n generals $(n-1)/3 = m$ generals are malicious. Today, there is a whole class of algorithms that address this problem. This is the class of BFT algorithms.
In addition to the aforementioned publications, there exist numerous additional sources. Nonetheless, the protocols that are utilized in blockchain systems are of significance for the purposes of this dissertation. Herein lies a comprehensive overview of the most significant procedures that are currently in use. The list is not comprehensive due to the substantial number of procedures. All described methods have in common that there is a kind of random function that selects a participant of the network to be allowed to write data for a short moment in the ledger. The choice and generation of the random number differ significantly.

- Proof-of-Work (PoW): Different information is required for the formation of a block. At this point, we limit ourselves to the hash value of the previous block, the nonce, and the difficulty. The additional information is not of significant importance for the PoW consensus. In order to select a miner from all miners, it is assumed that the miner is motivated by a reward. Through the write permission, he is allowed to transfer to a wallet of his choice a maximum fixed amount of Bitcoin (BTC). Due to this motivating factor, every miner aspires to obtain write permission on a frequent basis. Write permission is weighted based on resources provided. If an individual contributes more resources to the consensus, there is a greater likelihood of obtaining write permission.

  This is technically made possible by the cryptographic hash value, which was previously explained as the block ID. Because not every hash value will be accepted by the network. As the hash value is a numerical value, it is possible to attach a mathematical condition to it. The condition is that this must be smaller than the difficulty defined in the block. If the hash value generated does not satisfy this prerequisite, the miner has the option to alter the nonce value and hash once more. This is done repeatedly until a suitable value is found. In that case, the block would be propagated. The Difficulty cannot be chosen by the miner arbitrarily. It contains a program that adjusts its value about every two weeks. This program checks the average time between the blocks. If the average is less than 10 minutes, the Difficulty is made more difficult and vice versa.

  There is a peculiarity with PoW. Projects that use the same algorithm as BTC are at risk of a mining attack. A miner with a large hash rate may decide to use their hardware for another project at short notice. The risk of a majority of attack is high if the other project is much smaller than BTC. Thus, there can effectively be only one blockchain project with the same PoW algorithm used. By tweaking the algorithm, Aceth could be protected from such an attack before hoping to PoS. [197]

- Proof-of-Stake (PoS): The basic principle here is similar to PoW. Here, too, miners are to be motivated by incentives. However, they are no longer called miners because the process changed. As a rule, the term validator is used here. In a PoS system, validators are the nodes responsible for validating and confirming transactions. To become a validator, a node must deposit a certain amount of cryptocurrency as a "stake." This stake serves as an incentive to trade honestly, as dishonest behavior can result in the Stake being lost ("slashing").

  In addition, there are epochs in PoS systems. Epochs are a concept used in many PoS systems to divide time into segments. An epoch is a fixed number of slots (time units), and in each slot a validator is entitled to create a block. The assignment of validators to slots is usually determined at the beginning of each epoch and can be done in various ways, e.g., by random selection, where the probability is proportional to the stake of a validator.

  There are different evolutions from the PoS. A distinction can be made among the algorithms used. For example, for the voting among the validators of an epoch. Or also for the random selection of the validators for an epoch. In addition, there is the so-called delegated PoS. Here, several participants can give a partial take to a selected validator. This validator has the same tasks, but then shares the reward proportionally with the supporters.

  There is a stronger debate whether a PoS system is still decentralized or has centralizing properties. The economic factor is emphasized above all, that simply larger assets can easily authorize themselves as validators. This should make it easier for token rich people to become even richer, and inequality should become increasingly greater. [211]

- Proof-of-Authority (PoA): The PoA mechanism is a consensus mechanism utilized in certain blockchain networks. It is a variant of PoS, but with a select group of validators that are usually recognized and trusted entities. There are two well-known variants of in this regard.

Clique is the PoA consensus algorithm used in theETH chain Goerli and some other private Ethereum networks. There exists a predetermined set of validators that are responsible for validating and confirming blocks. These are selected when the network is launched, but can be changed later through a voting process. A validator is selected in a round-robin fashion to create the next block. If a validator is unable to create the block within a specified time frame, the next validator in the list is selected. [244]

Authority Round (Aura) is another PoA consensus algorithm used in some Ethereum-based networks, such as the Kovan test net. As with Clique, there is a set group of validators. These could be companies or other trusted entities, for example. In Aura, block creation is done in a scheduled round-robin fashion. Each validator knows when it is their turn to create a block, based on a predefined schedule. [270]

PoA is useful for networks that want fast transactions with a known set of validators. It is less decentralized than PoS or PoW, but offers advantages in efficiency and scalability. As a result, its use is more likely in consortial or private blockchains.

- Proof-of-Burn (PoB): With PoB, randomness is also weighted to a stake set by the miner. However, the stake is not maintained by the miner, but invalidated. This can be done, for example, by sending a transaction to an invalid address. For instance, there exist projects that evince a clear intention to safeguard tokens on their instance by transmitting a bitcoin transaction to an invalid address. This is the case, for example, with the project Stacks [210]. By means of this procedure, it is envisaged that one may utilize the security level of an existing blockchain, such as Bitcoin, for their undertaking.

- Proof-of-X (PoX): The previous consensus procedures demonstrated that randomness is the primary factor in determining a miner or validator for submitting transactions to the ledger. Besides the methods already shown, there are a variety of other methods that seek to execute randomness in different ways. For instance, by utilizing a trusted execution environment at the hardware level or by utilizing memory in lieu of hash rate. These further procedures are often also subsumed as PoX.

- Byzantine Fault Tolerance (BFT): BFTs are a class of algorithms for implementing a consensus mechanism. They are used in distributed systems to reach agreements among different network participants, even though some participants may act erroneously or maliciously. It refers to the Byzantine general problem, where communication and coordination are required despite possible traitors. Famous candidates here are the practical BFT and the redundant BFT procedure. In pBFT, five phases are presented. In the first phase, a transaction is sent to a validator. In the second phase, initial preparation by this validator takes place. The transaction is prepared for validation and provided with a signature by the first validator. Then, this signed transaction is sent to all other validators, if possible. This third phase is the preparing. Now again, all validators send the transaction signed by them to as many other validators as possible. Each validator should now have the number of validator transactions with two signatures each. In the commit phase, the decision is finally made whether the transaction is accepted as valid in the ledger. Here one quickly becomes aware that a chaining of the transactions does not necessarily have to be specified. Hence, no chained blocks. These procedures are thus attributed to the more general definition of DLTs. [59, 13]

Updates and Forks

There are three different meanings of a fork in bitcoin. There is a fork in the code base as a first understanding. This was used to implement the own blockchain projects. Furthermore, you can take the codebase of Bitcoin and customize it for your needs. A list of forks can be taken from the GitHub site [103]. However, this meaning occurs most rarely in the context of Bitcoin. A more common meaning is that of a chain split,

which can occur occasionally. If two miners discover a block simultaneously, they will be propagated simultaneously. A longer chain will become established over time, and the shorter one will be disregarded as an 'orphaned' chain.
The last meaning of forks is an essential one. It signifies a revision of the consensus layer. In the consensus layer, the rules are established for the network. The algorithm used for consensus is essential, but other rules, such as the block size limit or even which transactions exist, are also defined here. The process of an update requires a discussion in advance and a provision of a so-called Bitcoin Improvement Proposals (BIP) [32]. In a BIP the necessity and the procedure of the update is described. Should a reconciliation become necessary, this is executed by an update of the own node. To get a projection of the output in advance, there may be a signaling phase. In this, the version field of the block is used to make known the own intention of the update. If one adjusts the number, one signals the readiness to apply the update.
Also in the BIP it is determined whether the fork is a hard or soft fork. A soft fork is a backward-compatible update. Accordingly, nodes that do not apply the update are not immediately disconnected from the network. This contrasts with a hard fork, where there is a clear separation between updated and unchanged nodes. Examples of soft forks are new transaction types. These can usually be easily added if they look like old transactions. This is the case with the Taproot update, for example. If you have not updated your node, the Taproot transactions appear like ordinary transactions. The fact that there is more technology going on in the background is irrelevant from the node's perspective.
For soft forks in Bitcoin, there are two main approaches to activation: User Activated Soft Fork (UASF) and Miner Activated Soft Fork (MASF). The two soft fork types differ as follows: In UASF, the update is determined by the nodes, i.e., the users. The users of the network set a specific date when the new rules take effect. Anyone who does not accept the new rules risks being cut off from the rest of the network supporting the fork. Since a UASF does not necessarily have the majority of hashing power behind it, there is a risk of a network split if there is not enough support. In MASF, the miners determine the update. Miners signal their support for the change by inserting specific information into the blocks they mine. When a certain percentage of miners (e.g., 95%) signal their support, the soft fork is activated. Challenges here are that in the absence of consensus in the update discussion, a MASF can be stuck for a long time without being activated. Also, there may be incentives for miners to block changes that are not in their financial interest.
An example of a hard fork has arisen from the block size war. Updated nodes accepted blocks with a size of 8 MB. Nodes without updates discarded them. Thus, a separation was quickly realized. It is partly assumed that a hard fork would execute less coercion on an update, since a vote among all participants of the network is assumed. With a soft fork, the nodes would have less influence, should the miners agree. This is because they produce the blocks and could thus technically implement a soft fork without their consent. The node accepts these blocks because they are backwards compatible. [54]

2.2.3 Ethereum and Blockchains 2.0

After Bitcoin was introduced as the first blockchain, numerous other projects were developed through code forks. These projects, which contain more or less adaptations to the program code, are considered first-generation blockchains. A prevalent characteristic of these projects is the organization of data into blocks, and their consistency is ensured through a consensus process.
Ethereum was launched in 2015 by Vitalik Buterin and marks the beginning of a new blockchain generation. In contrast to the previous projects, Ethereum is mainly characterized by the extended functionality of its scripting language. This expansion enabled the implementation of loops and, as a result, the emergence of the famous smart contracts. All blockchains that support smart contracts belong to this second generation. However, the term "smart contract" can be misleading. As discussed in chapter 1 Introduction, this thesis therefore uses two terms: the smart contracts commonly used in the literature and the term "blockchain programs."

Ethereum was not only the first blockchain to introduce a second generation. This innovation has led to the birth of a project that has been compared to Bitcoin in numerous debates. Ethereum pursues the goal of creating a “world computer” in which resources such as memory and CPU can be provided and used globally. This naturally raises the question of the meaningfulness of on-chain computations, a central feature of this world computer. To achieve this ambitious goal, some technical modifications were necessary compared to Bitcoin.

The initial of these modifications is already known: the Bitcoin Script required modification to directly implement for and while loops. This step transformed the script language into a Turing-complete programming language. The OP codes are interpreted in the so-called EVM. Many subsequent projects that provide blockchain-based programs are EVM-compatible. Consequently, programs originally developed for Ethereum are also compatible with other platforms. This greatly broadens the scope of potential uses for other blockchain-based initiatives.

In order to realize the expansion of the OP codes, further technical adjustments were necessary. A purely transaction-based system would have compromised the efficiency of the entire blockchain. Therefore, ETH was designed as an account-based system. Unlike many other systems, Ethereum does not use a Merkle tree to prepare transactions for blocks. Instead, it uses Merkle tries. These do not require a strict binary structure and therefore offer more flexibility in their creation. In addition, tries are more efficient, especially for modifications and state storage. The increased complexity of blockchain applications and the resulting storage requirements make a Merkle trie the optimal data structure for storing transactions. [263]

Another key difference is the consensus mechanism used. Although plans for a PoS system were discussed when Ethereum was launched, Ethereum started with a PoW. However, this was modified to perform calculations most efficiently on graphics cards. In comparison, Bitcoin was already using FPGAs and ASIC hardware. Without this adaptation, Bitcoin miners would have been able to use their hardware for Ethereum in the short term, significantly increasing the risk of a ’majority attack’. In September 2022, the transition to PoS was completed through "The Merge." This change required prior customization and cost many years of development time. A key aspect of the consensus process is randomness. While this is generated in PoW by finding the right hash value, an appropriate solution must be developed for a PoS system. In Ethereum, this is done by the ’RANDAO’. In this system, a random generator is used to select the appropriate validators based on the weights of the stakes. This process must be both random and transparent. Since machines are usually deterministic, it is difficult to generate non-deterministic randomness. By changing the consensus process, the variable block time interval was set to a relatively constant time of 12 seconds. This is the amount of time a validator has to generate a block before it is the next validator’s turn. [8, 21]

Furthermore, efforts have been made to enhance the metrics of the Ethereum Blockchain, thereby enhancing its efficacy as a global computing platform. This includes setting the block time. For Bitcoin, this is 10 minutes. Due to the high complexity of smart contracts, one could expect a similar or even longer duration. However, with an average of 14 seconds in the PoW and a constant 12 seconds in the current PoS system, the interval times are significantly shorter. This allows for much faster execution. Furthermore, a mechanism has been implemented to destroy tokens when a transaction occurs. To counteract excessive inflation, this burn function is intended to compensate for constantly issued tokens. [211, 263, 89]

Blockchain technology offers the promise of decentralization. Critics criticize in particular that the Ethereum Foundation gives the ETH project a central decision-making authority. The question whether the Foundation can be considered a significant central authority remains a matter of debate. On the one hand, it was pivotal in influencing the modification of the consensus process, which was not in the best interests of the miners. Furthermore, there are weaknesses in the apparent decentralization that manifest themselves particularly through development processes, such as centralized services like Metamask and Infura. Both of these services provide access to the blockchain. However, as a centralized entity, they contradict the promises of a decentralized ledger. On the other hand, there is little evidence of actual Foundation influence on the ongoing project. One argument in favor of extensive decentralization is the diversity of clients. There are many software solutions to run an Ethereum Node. If a bug is discovered in one software, another might not be affected. An overview of the distribution of clients in use can be found here: [66].

Ethereum was first introduced in 2013 with the white paper [93]. This document presented the vision and ideas behind the technology and network. The later-released Yellow Paper offered a detailed technical guide to assist programmers in deploying their client version, [263]. Ethereum itself emerged from an Initial Coin Offering (ICO). In it, interested investors could exchange their BTC for ETH. The term was chosen regarding Initial Public Offerings (IPOs), in which companies offer their shares for sale on an exchange for the first time. In 2016, the network was finally launched, and it was possible to exchange tokens and mine blocks.

A defining event occurred in 2016. A programming error in a smart contract was exploited to steal the funds deposited on it. This vulnerability was the Reentrancy issue. It refers to a vulnerability in smart contracts where a function within a contract can be called again during its execution before it is finished. This can lead to undesired or unexpected results, especially when it comes to transferring Ether or changing contract states. The attacker exploited a re-entrancy vulnerability to repeatedly withdraw Ether before the contract could update the account balance. To prevent re-entrancy attacks, developers should ensure that state changes or Ether transfers are only performed at the end of functions, after all other actions have been completed. There are also special modifiers and security libraries in Solidity that help prevent such attacks. This victim smart contract was called Decentralized Autonomous Organization (DAO) and was designed as a digital democratic venture capital fund. Participants could deposit their tokens and vote on how they should be used. Due to the large participation, many tokens were stored in this contract. The attempted theft represented the equivalent of $50 million. In response to the hack, a vote was held in a short period of time to return the attacker's funds to the aggrieved parties via an update. This hard fork was highly controversial in the community and led to a chain split and the creation of Ethereum Classic (ETC). The event has remained controversial until now, casting shadows over the Ethereum Foundation [203].

In the following years, Ethereum experienced several hype phases due to its innovative technology. The ICO hype, for example, enabled the launch of numerous projects via this ICO method [83]. Although ICOs still takes place today, it does so with significantly less capital and attention. In 2021, the industry experienced another hype: the Non-Fungible Tokens (NFTs). Using smart contracts, digital assets were created on the blockchain to represent the uniqueness and ownership of digital artwork, collectibles, and other media content. Through NFTs, artists and creators can directly monetize their works, while buyers acquire authentic and unique versions of this content. This led to record-breaking sale prices and intense debate about the value and sustainability of digital art and collectibles. After peaking, the enthusiasm waned, and the hype leveled off [86].

Ethereum has evolved blockchain technology with a variety of innovations and adaptations. From the introduction of smart contracts to technical tweaks to controversial decisions like how to handle the DAO hack, Ethereum has shaped both the technology and the community. Despite its successes, Ethereum continues to face challenges, particularly in the areas of decentralization and scalability.

2.2.4 Solutions for Scalability Problem

A major problem with decentralized systems is that they scale poorly. For cryptocurrency projects like BTC or ETH, this is problematic if high global usage is expected. Poor scaling could even stand in the way of global usage. Since this issue was recognized very early, there are already some ideas that try to counter it. Some solutions try to accumulate transactions in different ways in order to only have to execute a few transactions themselves on the blockchain. Others are trying to shift the burden to different instances. Either by systematically dividing the blockchain or by severing the blockchain's data structure. No matter what the solution may be. Ultimately, the solutions are still subject to the CAP tehreom, from the chapter 2.2.1 2.2.1.

Second Layer Solutions

Second-layer solutions (also referred to as layer 2 solutions) are technologies that build on top of the main blockchain (the "first layer") to increase scalability and reduce transaction costs and delays. These solutions leverage the security of the underlying main blockchain while attempting to overcome some of its limitations. In idea, these solutions are similar to the PoB approach. However, no tokens are burned here, but the main blockchain is used in an ordinary way for transactions.

Lightning Network: The Lightning Network is a second-layer solution for BTC. Here, payment channels are established that determine the current account balance through the bilateral exchange of signed transactions. Only when a payment channel is closed does the most recent account balance come onto the Bitcoin blockchain as a closing transaction. Such a closure can happen amicably, after the transfer of the maximum available balance, after the expiration of a predefined period, or unilaterally in the case of suspected fraud. The latter situation requires more detailed consideration and protection mechanisms. In addition, with the Lightning Network, it is possible to create so-called routes of payment channels. Thereby, for a transfer of BTC it is not necessary to have a direct connection to a payment channel with the remote person. It is enough if a connection can be found through third-party payment channels. A widely used implementation can be implemented with Lightning Network Daemon (LND) [177] can be found and tried. [208]

Channel solutions: There are similar solutions for Ethereum and other systems. For Ethereum, in particular, these are the channel solutions. Here, too, payment channels are formed, which are only realized on the blockchain after completion. [113] Nevertheless, all channel solutions also have problems. Since transactions still need to be carried out, such a solution is only a postponement of the concern. If the application progresses faster than the solution can remedy it, the scaling difficulty remains. In addition, routes pose a problem. If one wants to have the cheapest route, the solution needed for this ends up being a solution to the traveler problem, which is classified among the NP-hard problems. Other issues arise due to complexity. [233]

Rollups and Zero-Knowledge Application: Rollups are a solution where transaction data is processed off-chain and then transferred to the main chain as a “bundle” in a single batch. There are two main types of rollups: zk rollups, which use ZKP, and Optimistic rollups, which are based on game theory and challenges. The basic idea is to process transaction data off-chain and then commit a summary proof to the main chain. Optimistic rollups assume that all transactions are correct, unless there is evidence otherwise. Transactions are processed immediately, but there is a “waiting period” or “challenge period” during which someone who notices an error or malicious action can present evidence against it. If someone successfully reports an error, there are penalties for the malicious actor. The zk rollups use zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) or other zero-knowledge proof techniques to prove the validity of transactions without having to reveal the underlying data. Unlike Optimistic Rollups, zk rollups validate transactions without the necessity of a challenge period.

Schnorr-Signatures: Schnorr signatures are a digital signature algorithm that is being discussed in the context of Bitcoin and other cryptocurrencies because of its efficiency and other advantages over the currently used Elliptic Curve Digital Signature Algorithm (ECDSA). In this regard, Schnorr signatures stand out for several reasons. First, they are more efficient than the ECDSA signature method, which already saves memory for the same signature applications. In addition, Schnorr signatures, by using the discrete logarithm in their implementation, provide a way to aggregate signatures. This means that multiple signatures from different senders can be combined into a single, common signature. This is made possible by the computational rules of the logarithm functions. This can be used in multi-signature processes or other more complex processes. The aggregation of various transactions into a single one is thus already technically possible.

Sharding

Sharding, originally a concept from database technology, was developed to optimize the scalability and performance of database systems. In this process, data is divided into smaller segments, so-called "shards." In large databases, segmentation into shards can achieve more efficient data processing since each shard acts independently, and queries can therefore be processed in parallel. This increases the overall efficiency of the system.
In the context of blockchain, especially Ethereum, sharding refers to the segmentation of the entire network into several such shards. Each of these shards autonomously processes its transactions and smart contracts. The implementation of sharding is expected to increase the capacity and throughput speed of the Ethereum network, as transactions can now be executed in parallel in different shards instead of linearly in a single chain.

Alternative Data Structures

We have already looked at the first and second-generation blockchains. The range of functions was expanded during further development. Third-generation blockchains are those that improve interoperability and scalability. Such solutions are understood, for example, with Polkadot, Cardano, IOTA, or ETH 2.0 with sharding.
In order to deliver on the promise of better operability or scalability, some projects are trying to break the strict linear data structure of blockchains. In IOTA, for example, an entire tree of transactions is spanned, and it should be possible to easily integrate new subtrees. This should make it easy to integrate offline transactions, as long as there is no contradiction in the transaction sequence. [271]

The solutions presented seem promising. Nevertheless, they all come at the cost of decentralization, as implied by the CAP theorem. At this point, a complete presentation of all solutions has not been proposed, as it is beyond the scope of the thesis. Further reading, such as [271, 255], is recommended.

2.2.5 IPFS and other Off-Chain-Databases Solutions

Owing to the immutable nature of many blockchains and DLTs, introducing larger data sets into these systems presents challenges. Additionally, the capability to modify or delete previous entries is typically absent. Yet, certain applications deem such functionalities vital, and in some instances, legislative mandates necessitate them (as seen with the GDPR). Chapter 2.2.3 Ethereum and Blockchains 2.0 (page 23) elucidates one potential approach for addressing this issue. However, the reliance on smart contract-enabled blockchains and Oracles poses concerns over excessive centralization.
To mitigate these challenges, one can consider the InterPlanetary File System (IPFS) as a potential solution. As described by [226], "IPFS is a distributed system for storing and accessing files, websites, applications, and data". In essence, the structure of IPFS mirrors that of contemporary systems like Content Delivery Network (CDN). In a CDN, servers form a networked alliance to distribute data efficiently and cost-effectively. The data is replicated across several servers, and a client's request is directed to the server that promises the quickest response. Unlike CDN, however, IPFS decentralizes the organizational and structuring aspects, prioritizing economic efficiency in data retrieval while ensuring a decentralized organizational layout. This design not only combats censorship but also minimizes the risk associated with single points of failure. Since IPFS functions as a P2P network for data distribution, parallels are often drawn with the World Wide Web (WWW). Yet, the two systems differ fundamentally in their architecture. [226, 73]

The IPFS system is enabled by using a content-based link. The location-based links used in WWW are much more familiar. This refers to an IP of a server that has the file. A content-based link, as used in IPFS, consists of a hash value of the date and a name. If one wants to search for a file, one needs this hash value and can thus search globally on multiple nodes of IPFS. In comparison to the WWW, where one asks a server specifically for the file. Due to the property of the hash value created from the date using a cryptographic hash function, the integrity of the date can be checked and ensured. Similar to how CDN can benefit performance, the developers, and researchers of IPFS also promise to speed up the information. This is because instead of having to request a server that may be far away, the data is potentially near you, thus speeding up the transfer significantly. [20]

In addition to the differences in the links, the architecture of the system must be fundamentally different from WWW. This is because if we want to find a file, the search must be efficient, but the hash value may be unknown. Therefore, we need systems that make finding information easier.

To meet these goals, remedial technologies already exist. To achieve this, let's look at the data structure of how a datum is stored in IPFS. First, we require an ID that can be used with the link. As mentioned earlier, this is done using a cryptographic hash function that gives us a collision-free value and thus can serve as an ID. If we now change the date, we get a new hash value. The links and the representation on our system have changed. To keep the changes minimal, the data is divided into blocks. This data is managed on a Git system. Git is a version control system where changes to files, their versions, can be managed and viewed. In addition, Git is a distributed system, so the data is not located on a server, but a version is contained in all participants. Chained hash values prevent the subsequent manipulation of historical records. Chaining old information with current information to ensure integrity also occurs in blockchains and many DLTs. Therefore, the comparison of these two technologies is also frequently used, especially by opponents of the technology. Through the Git system, we obtain a so-called Merkle Directed Acyclic Graph (DAG). The addition Merkle is meant to reference the analogous Merkle Tree, where a concatenation of hash values takes place. In this way, a link between the individual data blocks can be created, and thus integrity can be guaranteed rapidly. [20, 226]

Through the described systems, it is already possible to identify data on the IPFS and to manage them distributed on the servers. Only fast access to this data is denied to us because we can first send the ID to all nodes and wait for their response. A speedup and reduction of requests can be realized by DHT. A DHT is a hash table that can be distributed on multiple nodes. Here, it consists of the IDs of the respective data blocks contained on a node. This information can be searched very efficiently ($\theta(\log(n)$ [245]). [20, 226]

By distributing data across nodes, there is a major drawback. At least the metadata is public to everyone. This includes the information on which nodes the respective data is stored because the DHT provides this information. One can compare this problem with the transparency problem of blockchains and DLTs (2.2.6 Problems, Disadvantages, and Concerns of Blockchain Technologies on page 29). Moreover, the data is not encrypted on the nodes. This makes sense if the data was designed to be public. However, if one is developing a system where personal or similarly sensitive data is to be processed and stored, then this would not be an option. To increase security, there are a few options, depending on the weighting, all of which would have to be used. First, there is the possibility that the data is encrypted in advance. The nodes then all have an encrypted file with them in the administration and cannot read the file themselves. Due to the versioning, the file would then have to be re-encrypted and stored when a change is made. This would also allow updates to the encryption itself, should the algorithm currently used show weaknesses (keyword: post-quantum cryptography). What cannot be prevented is the retention of old files. A user cannot force a node to delete a file. When a deletion request is made, only the transaction is received by the nodes. Whether the file is then also deleted is up to the node operator. This can be critical if data does not change much but was stored with compromised encryption. In this case, the file would no longer be secret and would be publicly accessible.

Another option is to create a private IPFS network. Here, only shared nodes would be allowed to participate

in the system and manage the respective data. But this solution comes at the expense of the advantages of the IPFS system, such as distribution and efficient access. Furthermore, a possible use by non-trusted node operators is no longer guaranteed since they know each other in the private network and trust is given.
In summary, IPFS can be seen as a fusion of Git with a DHT. Quite analogous to Bitcoin, which was developed from already existing P2P elements such as the Gossip protocol or the Hashcash algorithm with the cryptographic Chain of Data. In both concepts, already existing technologies were used and built together in a meaningful way. And this is undoubtedly where the added value of this technology lies. Through IPFS, a single-point-of-failure and censorship-resistant system is to be given, which is to map our known Internet. In addition, further technologies are to be represented, such as a blockchain or versioned data systems (file systems). For the actual realization of a distributed internet, one has to give the IPFS a naming system similar to DNS. This is because the data changes its link when minimal changes are made, and thus one has to enable the transfer to other participants of the new link. Such a system has already been implemented and is called InterPlanetary Name System (IPNS). [20]

The IPFS is not the only way to deploy big data blockchains. In addition to the IPFS, there are a variety of other P2P data networks that can be considered in this regard. These would include. *Swarm, Arweave, Storj, SAFE, Hypercore*, to name a few [73]. In the source just cited, one can find a comparison of the solutions mentioned. There is also a project in the Hyperledger universe that deals with off-chain data. With Avalon, this is to be ensured on hardware provided by Intel, especially for secure transactions [154].
It can be seen that this topic is quite complex, and a system architect has to look at many projects. No projects have yet proven to be superior to others. This is another point that makes it difficult to build a meaningful blockchain architecture. Especially when the use cases that could potentially run on this architecture have not been fully clarified at the current time.

2.2.6 Problems, Disadvantages, and Concerns of Blockchain Technologies

In the evolving landscape of technology, while breakthroughs and innovations are hailed for their potential benefits, it is equally crucial to cast a discerning eye on their inherent challenges. Every technological advancement, no matter how revolutionary, comes with its own set of drawbacks and issues that need to be addressed. This chapter delves into the problems, disadvantages, and concerns associated with blockchain technology. By analyzing these aspects, we aim to present a holistic view that goes beyond the initial excitement and looks into the potential pitfalls and repercussions. Through a comprehensive exploration, readers will gain a more in-depth understanding of the complexities involved, fostering informed decision-making and critical thinking in this domain.
The following is a summary of some problems and concerns. A detailed consideration cannot be given here. Nevertheless, an attempt is made to give a detailed view of current discussions. Problems that have a deeper impact on later development are described in more detail.

Criticism of Blockchain Systems

Blockchain technology, despite its potential and disruptive nature, is often the focus of criticism. Many critics see it as not fit for purpose in a socio-economic context. A common argument is that the technology is mainly useful for criminal purposes. Critics argue that the demand for a currency that cannot be controlled by states only serves criminal activities. Moreover, the lack of traditional security mechanisms, such as deposit insurance, would impose an excessive level of responsibility on users. This could lead to widespread disadvantages and further encourage criminal activity. The following sections discuss these and other criticisms in detail.

Scalability: Detailed information about the problems and solutions to scaling was given in 2.2.4 Solutions for Scalability Problem and 2.2 Blockchain Technologies spoken. In summary, current blockchain solutions are reaching their processing limits at high usage.

Energy: Blockchains and their energy requirements have attracted considerable attention recently. In particular, the widely used consensus method PoW (proof-of-work) has come under criticism for its significant energy consumption. The amount of energy consumed by large blockchain networks such as Bitcoin is already comparable to the energy requirements of entire countries. However, there are alternative consensus methods that are significantly less energy intensive than PoW. Examples include PoS and BFT processes that aim to increase energy efficiency and reduce the environmental footprint of blockchains. However, the exact extent of blockchain costs and energy consumption is difficult to quantify because there are many factors to consider, including the type of hardware used, the efficiency of mining pools, and the energy sources they use. Interestingly, blockchains could also indirectly help promote renewable energy. By constantly searching for more cost-effective power, miners could be encouraged to use more renewable energy sources, which could eventually be cheaper. Finally, it is important to consider the energy consumption of blockchains in the context of other sustainability issues. These include not only environmental considerations, but also social and economic ones. This underscores the need for a holistic approach when assessing the impact of this technology on our world. This issue remains highly contentious and will likely continue to be debated for some time. [207, 223, 163]

Problem first, Second Development: One of the most common criticisms of blockchains is that they attempt to solve a problem that may not yet exist in its current form. Historically, technologies and scientific discoveries have often been developed to address existing problems. With blockchains, however, it appears that the technology is ahead of the curve and looking for applications that fit its unique mode of operation. This criticism is not entirely unfounded, as many proposed applications for blockchains could typically be implemented more efficiently and easily with existing, centralized systems. Nevertheless, it is important to recognize that many scientific discoveries and technological breakthroughs do not find practical applications immediately, but are often decades or even centuries in advance. A classic example of this is the study of prime numbers. Prime numbers were studied for thousands of years, long before they found a concrete, practical application in modern technology. It was not until the discovery and implementation of asymmetric cryptography in the 20th century that the value of prime number research to modern technology and security became apparent. Likewise, it can be argued that although blockchains are currently seeking clear applications in some areas, the time may come when their uniqueness and decentralized properties become indispensable. It is possible that future problems and challenges will require the very solutions that blockchains can provide. Overall, scientific and technological research should not be judged solely based on immediate practical applications. History has shown that what is considered purely academic or theoretical today may be the basis for revolutionary breakthroughs tomorrow.

Centralization on Blockchains: While decentralization is highly valued as one of the main advantages of blockchains, centralizing tendencies can still be observed in practice. Some of these centralizing aspects can be observed in services such as Infura and Metamask, which are further discussed in 2.2.3 Ethereum and Blockchains 2.0. Furthermore, there are numerous cryptocurrency exchanges that not only allow the exchange of cryptocurrencies, but also offer custody. Such services could be interpreted as signs of a creeping centralization of the system. Another aspect of centralization lies in the increasing number of services and applications implemented directly on a blockchain. In the past, each application often had its own separate system landscape. Now, if more and more of these applications are consolidated on a single blockchain, this could become a key point of attack or vulnerability. Therefore, it is important to carefully balance the benefits of consolidation with the risks of over centralization.

Criminal Usage: Blockchain technology and cryptocurrencies are not only known for their revolutionary applications and transformations in the financial sector. They also attract criminal activity due to their characteristics. One of the main reasons for this is the pseudonymity offered by many cryptocurrencies. Although transactions are stored on the blockchain for all to see, they are often not directly attributable to a real person. This allows criminal actors to move funds without being easily tracked. As a result, cryptocurrencies are used for money laundering, ransomware attacks, or trading illegal goods on the dark web, for example. Another criticism often leveled at cryptocurrencies is the claim that they may constitute a "Ponzi scheme." These schemes promise investors high returns that are mainly funded by deposits from new investors, rather than legitimate business activities. When new investors can no longer be found, the system collapses. Some crypto projects, especially ICOs, which entered the market without a clear business plan or real applications, have been exposed as such schemes in the past. However, it is important to emphasize that not all cryptocurrencies or blockchain projects should be considered pyramid schemes. Many have legitimate applications and are bringing real innovation to a wide variety of industries. As with any technology, there are both legitimate and harmful applications. It is up to regulators, institutions, and the community to ensure that the technology is used as intended and that abuses are minimized. In addition, with reconnaissance measures, the identities behind pseudonyms can be elucidated. The subsequently clarified transaction history could help with further investigations here.

The Oracle Problem

The Oracle problem must be considered here in the context of blockchain. In other domains, such as AI and Machine Learning (ML), it is also defined and describes the problem of being able to evaluate a response from a Machine Learning (ML) system.
A blockchain is a closed system. Here, information is created and updated only within the system (on-chain data). Such information can be the (virtual) account balance. However, if one wants to add other information (off-chain data), it requires the creation of a new instance. This instance is called Oracle. The problem arises when a dependency on this instance arises. A dependency that one would actually like to reduce or avoid by using a DLT. This could allow manipulated data to enter the system.
Such an Oracle could be implemented in the form of a smart contract. This contract would inject data into the blockchain regularly or on demand, depending on the need. Here, one realizes that such solutions are also expensive. This is because the (off-chain) data must be stored in the Distributed Ledger- (DL-) using a transaction.
Measures to avoid centralization include several possibilities that, taken together, can have the greatest potential. The first measure would be to not rely on one Oracle service, but to find multiple service providers for the same use case. However, this implemented decentralization also comes at a price, as the same data would have to be requested multiple times. This variant exposes the Oracle operator to economic pressure, so that the costs of a manipulation would increase, and thus the revenue from a successful attack would no longer be profitable. However, this measure can only detect manipulation. For automatic avoidance, a consensus is needed, which takes effect in the case of a contradiction and automatically increases your trustworthy value. Such a consensus can be implemented, e.g., by a Proof-of-Authority (PoA) model with numerous Oracle services and contracts.
Another measure would be to set up an Oracle service on trusted hardware. Here, the hardware vendor promises tamper-resistant execution of programs. An attack by the Oracle operator would thus be made more difficult, and the game-theoretical cost-benefit calculation would no longer be worthwhile. [56, 234]

The Transparency Problem

The transparency problem describes the fact that in public and semi-public blockchains (blockchains with limited write but public read permissions), every transaction can be seen and analyzed by anyone.
Although this transparency has some advantages, such as reducing fraud and increasing the trustworthiness of the system, it also raises privacy and confidentiality issues. Because transaction data is publicly viewable, third parties may be able to draw conclusions about participants' identities and activities, even if the information in the blockchain is pseudonymized.
It also poses the risk of personal data being written to the blockchain in projects from the business sector or public administrations, which could subsequently be decrypted and thus put the individuals concerned at risk. Alternatively, only links to this data could be included, but this could mean an amplification of the scaling problem.
Proposed solutions could include more restricted blockchains and DLTs. So-called permissioned and private blockchains (see also 2.2 Blockchain Technologies). However, since the risk of loss of personal data persists, albeit among a much smaller set of potential attackers, this is not a sufficient solution.
Another approach is to strengthen privacy through cryptography. One implementation of this is Monero [166], for example. Here, ring signatures, stealth addresses, and confidential transactions are used in an attempt to fully encrypt information.
Despite these developments, the transparency problem remains in many blockchain networks, and there are ongoing discussions about how best to ensure privacy and confidentiality without compromising the security and trustworthiness of the system. [229]

2.3 Authentication

There are currently many authentication methods being developed and used. At this point, I will give an overview of the common methods, but also of the methods used by RADIUS that are essential to the thesis. Thereby, the processes in the architecture differ significantly when considering network authentication, or web service authentication. In this chapter, a general definition and description of authentication are given. There is no need to distinguish between authentications because the roaming property of RADIUS (see chapter 2.7.1 RADIUS Protocol) makes the boundaries of the application areas disappear to some extent.
Authentication is the process of ensuring the identity of a person or device. In this process, the person or device usually proves that it is also the particular entity it is impersonating. Authentication is needed in many places. A person in real life can identify himself with the ID card, and a house can be found with the address. A PC on the Internet can be identified by the MAC address of the network card, along with the IP. In the digital world, such processes are generally more difficult. Digital goods are typically not unique and can be copied at will. This means that methods such as ID cards are initially out of the question. In addition, the link between the digital world and the physical world is just as difficult to assign unambiguously. Anyone can introduce themselves as someone else on a forum; MAC addresses can be spoofed, and IP addresses can be obfuscated through VPN and TOR connections. For the thesis, electronic authentications are essential, and physical ones can be neglected for now.
Authentication can be done in numerous ways, such as username and password entry, fingerprint or facial recognition, security tokens, or a combination of factors. In the past, the username and password combination has emerged as the first authentication method in the digital world. Even now, most websites and software products use these combination solutions. But the difficulty of managing passwords, both from the provider side and the user side, shows that these methods are not applicable everywhere [114]. Especially with devices in mind, methods should be developed that provide fast and secure authentication.
Effective authentication is an important aspect of information security and is used in many fields, including IT security, banking, government, healthcare, and other critical infrastructure. Because this process is

necessary in so many applications, the requirements are also changing. The so-called Request for Comments (RFC) documents define standards that are helpful for further developments. These are published by the Internet Engineering Task Force (IETF). The IETF is an open, international community of experts working on the development and standardization of Internet protocols and technologies. And thus, many implementations exist. A small overview is listed below. It should be noted that this listing is incomplete.

In an authentication process, the claimant (applicant) attempts to authenticate itself to the verifier (verifier). This process is only possible to a certain extent because a potential attacker could compromise or break this connection at any time. Therefore, processes usually take place to keep one authenticated. This can happen with a time-limited token, or in repeated authentication processes that take place at intervals. The possible attack vectors (see below), as well as the increasing complexity of information technology, require a more complex role representation for these processes. Thus, there are so-called Public Key Infrastructure (PKI) systems that use Certification Authorities (CAs) to organize and secure certificates. Here, the claimant can have a certificate authorized so that the verifier has more security in the verification. This concept can be found in the TLS/SSL encryption of websites. Here, the user can use a certificate to ensure that the address was entered correctly and that the traffic was not tampered with. An essential part of PKI systems is the Certification Revocation List (CRL). Certificates are usually created with an expiration date. When a certificate reaches its expiration date, it is marked as invalid. Additionally, the owner of a certificate can also mark it as invalid, which is called revoking it. All invalid certificates are maintained in the CRL, viewable by all. [43]

A Relying Party (RP) typically accesses the authentication service of a Identity Provider (IdP). This checks and verifies the identity in the process. In the context of Single Sign-On (SSO), the RP performs the task of authenticating and authorizing the user with data from the IdP for the application.

Evolving Self-Sovereign Identity (SSI) concepts are also making the classic authentication scheme more complex. In the SSI concept, there is the issuer (claimant), the holder, and the verifier. We already know the verifier, and again, it verifies the information at hand. The claimant has been split into a holder and an issuer. This means that the user himself creates a form of identity (ID), and another instance attaches verifiable credentials to this ID. The verifier can now check the ID on the one hand, and the credentials to it on the other.

Authentications can be divided into six categories. These categories are formed based on the factors essential for authentication. These categories differ significantly in security and application. However, it is recommended to take at least two methods, which correspond to category 5. It is not possible to give a general recommendation due to the different needs of the various application possibilities. [52]

1. **Knowledge:** These methods require knowledge from the user. This knowledge can be, for example, a password, PIN, or an answer to a question.

2. **Ownership:** The user has things necessary for authentication, such as a token, smart card or mobile device.

3. **Biometrics:** A biometric feature is used by the user for authentication. This can be a fingerprint, an iris scan or facial recognition.

4. **Behavior:** The user must authenticate himself by a behavior. This can be the way one types or moves the mouse.

5. **Context:** This method uses contextual information such as the user's location or device to increase the probability of the user's identity. Credit card companies use such information to identify credit card theft and misuse.

6. **Multiple Factors:** This method requires the user to combine several of the above factors to authenticate. For example, a multifactor authentication (MFA) method may require a combination of knowledge (e.g., password) and possession (e.g., security key).

As with everything in IT, authentication can never be impregnable. This means that there are ways that an attacker can either tap information from an authentication process, or sabotage the authentication itself. Tapping information is more serious because, from it, the attacker could gain the ability to authenticate as someone else. Such an attack is difficult to trace and detect. This means the attacker could spend a long time authenticating himself as the other person or device. He would also have enough time to destroy any traces. In the event of sabotage, the attack would be noticed immediately, and it would be possible to react to it. Nevertheless, it is important to avoid any possibility of attack. In addition to deliberate damage by an attacker, unintentional damage can also occur in an authentication process due to the negligent actions of key players that enable an attack vector for an attacker. Central offices in particular can facilitate a weakness here.

One type of attack at the protocol level of authentication describes the so-called *eavesdroppers*. During this type of attack, it is assumed that the attacker can intercept the communication of the authentication system. The attacker hopes to obtain information that will enable him to further decrypt the continuing communication. In this type of attack, the attacker has the opportunity to perform the high-demand decryption offline. Besides the eavesdropping attack, it is also possible for an attacker to simply guess the *password*. To accomplish this, the attacker tries different passwords and hopes to find a suitable one. Due to a poor choice of passwords, this is actually a threat that opens the network to potential attackers. Statistics indicate that users often fall into password patterns, or use fragile passwords overall [63]. However, in addition to weak passwords, targeted attacks on them are a popular target. As the statistics on the main reasons for ransomware attacks show, phishing is a major threat to networks [242]. In this process, the attacker deliberately creates manipulative emails or forges websites in order to obtain passwords. The victim thinks he is safe and hands it over to the attacker. With the password, the attacker can easily authenticate himself as the victimized user. Another type of attack is the *replay* attacks. The attacker manages to grab this data from a successfully performed authentication and apply it repeatedly. In doing so, the attacker does not have to decrypt the data of the communication, as would be attempted in an eavesdropping attack. This type of attack also differs from hijacking attacks (see below) in that the authentication of verifier and claimant does not have to be maintained. Rather, an already terminated session is attempted to be used repeatedly for authentication.

The aforementioned *HiJacking* attacks attempt to exploit an existing authenticated and still active connection for themselves. In doing so, the legitimate user can simply be observed in order to obtain further information. Or the legitimate user's privileges can be actively abused for the attacker's desired activities.

Finally, we want to describe *Man-in-the-Middle (MITM) attacks* . Here, depending on the attack, the attacker pretends to be a verifier or a claimant. He can do this, for example, after a successful hacking attack on a router, website, or user device. The claimant or verifier believes himself to be communicating with the right partner and reveals his information (e.g., password, user token, or similar). [52]

Measures to defend against threats must be well-thought-out and implemented. Different approaches are necessary for the various types of attacks. Of course, one assumes that the state of the art is implemented in the sense of using the right algorithms. This necessarily includes not developing one's own cryptography, but rather using existing procedures. Regular updates are also necessary. With this prerequisite in mind, we now want to look at countermeasures for the various attacks. Since future attack patterns are difficult to foresee today, we will increasingly describe general defense patterns here and, in some cases, refine them with an example.

In eavesdropping, the goal is to make eavesdropping as lucrative as possible for an attacker. Since the attacker is trying to use the active communication for future communications, the focus is to minimize the success

of an attack. If one changes keys quickly and continuously in communications, even if the communication partners remain the same, the attacker cannot do much with the potentially obtained information.
When it comes to password guessing, a tried and tested method can be used. Here, the focus of the defense is to increase the cost of an attack. This is achieved by increasing the time for failed authentications. This time is usually increased exponentially, i.e., with a higher number of failed attempts, the time to re-authenticate increases significantly. The user, who typically does not need many attempts to enter a password or similar, does not notice the few fractions of seconds. However, password-guessing software that enters millions of passwords in seconds makes successful retrieval much more difficult.
Adding information from an old authentication process to a new one helps define the timeliness of these processes. Such information can simply be a counter that is incremented at each authentication step. If the user wants to authenticate again, the counter is incremented, and the already-created tokens cannot be used once more. If one changes this information to a shared secret, one can turn the replay protection measure into a hijacking protection measure. It is important to note that information about the text content belongs in the protection process as well as the shared secret.
A protection mechanism that can help against many types of attacks is the use of a Public Key Infrastructure (PKI). Here, certificates are created and confirmed by the PKI. Only with such a certificate, it is possible for the owner to impersonate himself. The PKI offers a lot of potential for protection and is indispensable in today's network architecture. However, PKI systems are not protected against attacks. Exactly how a PKI works is discussed in 2.4 Public Key Infrastructure. With a successfully implemented PKI, threats of all attack types described above can be mitigated. In this protection measure, it is essential that the authenticity and integrity of a digital person, or device, can be verified more securely. This gives the procedure a great deal of importance, but also a great deal of potential danger due to centralization. The chapters described above also discuss the weaknesses of these systems.
The protective measures mentioned are common measures used against the types of attacks mentioned. However, protecting a network remains a competition against attackers. This is because they arm themselves against protection and identify new methods of attack. Therefore, a listing can never be complete. Other well-known protection measures, such as ZKP Password protocols (Encrypted Key Exchange (EKE), Simple Password Exponential Key Exchange (SPEKE), or SecureRemote Password Protocol (SRP)) can be found in the literature [51][52].

2.4 Public Key Infrastructure

Asymmetric cryptography gives us a way to establish identities on the Internet. This identity verification is realized with cryptographic signatures, which are also called certificates in the IT context. More about the procedures can be found in chapter 2.1.1 Symmetric and Asymmetric Cryptographic Methods. But there is still a problem with the signature methods, which is related to the Oracle problem. A detailed explanation of the Oracle problem can be found in chapter 2.2.6 The Oracle Problem. In summary, the Oracle problem describes the challenge of reliably getting data outside the blockchain ecosystem and into it. Any solutions through Oracle contracts provide a central point of failure, potentially compromising the decentralization and security of the blockchain system. In relation to our identity problem, we have to trust the sender because the signature procedure only allows us to inquire about the possession of the private key for the signature sent. But not whether the right person sent us the signature. This circumstance has been repeatedly exploited in MITM attacks.
One approach to solving this problem is the so-called Public Key Infrastructure (PKI) system. These serve as a Trusted Third Party (TTP) and keep a database of confirmed certificates. In the broadest sense, a certificate is a signature created by a user in response to an input given by the user. The input could be a reference to a digital identity, such as the address of a website. The exact structure of a certificate is defined based on need. Prominent definitions and agreements for certificates are X.509 or Pretty Good Privacy (PGP)[100].

In addition to the information already mentioned, other inputs may be required, or optionally added. Let's look at X.509 certificates as they are described. Typical for a certificate are a serial number, a validity period, information about the issuer, information about the owner, a public key, and a digital signature. The serial number describes the identity of the certificate. The validity period defines the period during which the certificate is valid. In numerous instances, the issuer is the CA. Information about the owner can be limited to an e-mail address and company name, but it can also be other values, depending on the certificate implementation. The public key is required for verification, and the signature is ultimately the information that confirms the contents of the certificate. [44] [22, ch. 17.6]
An PKI consists of several components. A detailed description and definition of all components are provided in the RFC documents [44] and updates to this document. The RFC 6818 [267] thereby describes an update of the Certification Revocation List (CRL). The two RFC documents 8398[187] and 8399[142] extend the specifications for international naming. This leaves the architecture in the RFC 5280. The major components are the Registration Authority (RA), Certification Authority (CA), and Certification Revocation List (CRL). The CA is the trusted authority, or TTP, with which the certificate applicant or even the certificate verifier are in contact. The CA can optionally hand over essential management functions to the RA. All interactions are stored and distributed in a certificate, or stored in the CRL. This defines a callback list. This list is important because certificates must be invalidated after a certain period of time or, manually, if keys are lost. This validity is thus managed and can be viewed in the CRL.
The scheme of a PKI registration is as follows [44]:

1. Request Certificate Signing: The user requesting a digital certificate (the certificate subscriber) creates a certificate request. To accomplish this, he generates a key pair consisting of a public and a private key. The public key is then included in the certificate request (Certificate Signing Request (CSR)) along with the user's personal information, such as name, e-mail address and organization.

2. Submit Request: The certificate holder sends the CSR to a Registration Authority (RA) or directly to a Certification Authority (CA), depending on the structure of the PKI.

3. Review of Request: The RA or CA verifies the identity and validity of the information contained in the application. This can be done by manual procedures (such as checking identification documents) or by automated procedures (such as checking e-mail addresses or domain names).

4. Certificate Creation: After successful verification, the CA creates the digital certificate. The certificate contains information about the owner, the public key, the validity period, the serial number and other important details. The CA signs the certificate with its private key to ensure its authenticity and integrity.

5. Deliver Certificate: The signed certificate is returned to the subscriber. The subscriber installs the certificate on its server or in its application so that other users and systems can verify its authenticity.

After a certificate has been created, it is up to the users to verify it. An example of such checks takes place in the web browser. Here, the certificates for the websites are checked so that the user does not fall victim to a MITM attack and end up on a false website.
The check includes several steps, similar to the creation. First, the certificate must be obtained. In the web page example, this is obtained when this is called up. The browser now checks whether the CA contained in the certificate is evaluated as a trusted instance (has been saved locally as such). This is followed by the signature check, which ensures the integrity of the certificate. In chapter 2.1.1 Symmetric and Asymmetric Cryptographic Methods can be found more about the cryptographic signature procedures. The validity and domain name checks ensure that one has the certificate obtained for the correct process and that it has not

expired. Finally, it remains to check whether a certificate has been revoked. This is done by checking the CRL, which can also be retrieved independently of the CA, depending on the implementation of the PKI.

There are several structures for how a PKI can be constructed. The most prominent structure is a strictly hierarchical structure, with a root CA. This CA distributes permissions in the form of certificates to subordinate CAs. These subordinate CAs can now also build a new level. Or it is already the lowest level, and the corresponding CA is allowed to issue certificates within its scope. There is also the structure where a CA is not hierarchical and directly issues certificates for authorizations. A hybrid of the two structures is exciting, in which users of a system themselves become a CA and certify each other (cross-certificates). Regardless of which structure is implemented, the so-called chain of trust is important. This is formed by an unbroken chain of trust between two users over other users. Either this chain is formed by certificate chaining via a hierarchical, flat, or mutually certified structure. There is always an instance at the end that must be trusted. [22, ch. 17.6.1]

PKI systems are usually implemented where the assurance of integrity and authenticity, or also for authentication methods, are realized. In chapter 2.3 Authentication we could see that PKI systems can be used to defend against many attack vectors on authentication systems. RADIUS (2.7.1 RADIUS Protocol) also suggests PKI systems for avoiding the potential of many attacks. Prominent uses are SSL and TLS certificates used in web browsers. Or for email security, so that a signature can be created for an address to ensure authenticity.

Wherever digital signatures are used for security, there could be a need for a PKI. This includes Virtual Private Network (VPN) or Internet of Things (IoT) systems, among others. But PKIs are also needed in the code signing process to sign software and applications with digital signatures to ensure the integrity and authenticity of the software and to protect users from tampered or malicious versions.

Besides the presented PKI systems, there are also specializations. One of these specializations is the Resource Public Key Infrastructure (RPKI). A RPKI provides security and integrity when exchanging routing information between different autonomous systems on the Internet. This involves providing IP address blocks and autonomous system numbers with certificates. [53] [69]

During the development of blockchain technologies came the idea of applying these developments to PKI systems. Such systems are called decentralized Public Key Infrastructure (dPKI). The decentralized ledger is intended to serve as storage for certificates, but management and distribution of keys can also be implemented. The advantages are seen primarily in four areas. One of the four arguments is the higher security against attacks, since there is no Single Point of Failure (SPOF). This makes DoS and MITM attacks ineffective. Another reason for a dPKI is the possible trust establishment without a Trusted Third Party (TTP). No entity would have to be trusted to manage and implement certificates prudently. In addition to the transparency, which can only be obtained in public ledgers, the high availability is a frequently mentioned argument. Among the disadvantages, scalability is mentioned, which could already be identified as a general problem of DLTs. In addition, the implementation of dPKI is currently still very complex, which could induce a higher susceptibility to errors. In addition to the current lack of acceptance, the legal and regulatory framework conditions are noted as a difficulty. As a potentially international operating system, it is difficult to define generally applicable rules and implement them accordingly. [206, 136] Further specializations will not be discussed here in order not to unnecessarily increase the scope of the thesis.

A Online Certificate Status Protocol (OCSP) is an internet protocol used to check the status of a digital certificate in real time. In a PKI, it is critical to know if a certificate is still valid or if it has been revoked. A certificate can be revoked if the private key associated with it has been compromised or if the certificate is no longer needed. This is where OCSP comes in: OCSP allows applications to send a real-time query to a OCSP responder server to check the status of a particular certificate. This is much more efficient compared to older methods, such as distributing CRLs because CRLs may only be updated and distributed periodically and therefore could have some delay. With OCSP, the status of the certificate can be queried and verified immediately. The OCSP responder, a server that answers the OCSP requests, checks the status of the certificate in its database and sends back a response indicating whether the certificate is good, revoked, or unknown.

We could see that PKI systems are indispensable for many security applications on the Internet. But they are also used in smaller networks, such as corporate networks. The hierarchical structure and complexity induced by the application of cryptography to a PKI also induce disadvantages and problems. A first indication of this is provided by the overview of documented failures in CA [247]. The overview provides data on failures between the years 2008 and 2018. The cases are documented and accordingly provide an overview of the so-called "light rate." It is naturally difficult to find information about the dark figure. In the paper [174] it is shown that "Recent researches and events have shown that due to malicious attacks or misbehaviors, CAs might issue fraudulent certificates, which may cause Man-in-the Middle (MITM) attacks or phishing attacks to the users".
The slightly older work [118] clearly shows the impact of complexity. Many security risks arise that can lead to false certificates and MITM attacks.
An estimate of the number of unreported cases can be taken from the "State of Machine Identity Management" report of 2021 from the Keyfactor Ponemon Institute [160]. There, it is clear that about 55% of companies are not hiring enough people for PKI positions.
A very detailed elaboration on trust issues towards CAs can be found in the paper [110]. Here, the tasks of a CA manager are described and how they can be implemented. In addition, detailed data on the current implementation of a CA manager is provided. "This work has analyzed and reported the security issues of trusted CA certificate management in current OSs and PKI applications".
The cost of a PKI is not negligible. If one wants or needs to set up one's own PKI system, there are some components to consider. First, one should distinguish between acquisition and runtime costs. A good overview of the costs of a private PKI can be found in the paper [246]. If one wants more explicit numbers, Global Sign [79] gives the sum of $1.8 million over 5 years for a company of 500 to 1000 employees. For comparison, in the Forbes article regarding a comparison of public and private, PKIs the cost of a single certificate is put at $1000 [129].

2.5 Man-in-the-Middle Attacks

MITM attacks represent a very prominent type of cyberattack. In this case, the attacker can compromise integrity and confidentiality, two security goals of cryptography (2.1 Cryptography). The potential threat arises from the attacker's intervention in the communication between two parties. The communication may simply be observed (compromise of confidentiality) or even manipulated (compromise of integrity). In the case of a successful attack, it is assumed that none of the parties is aware of the attack.
According to [24] and [68], MITM attacks can be divided into four classes.

1. **Spoofing-based:** In a spoofing attack, the attacker impersonates a legitimate entity on the network. This starting point can be used for other types of attacks besides MITM, such as DoS or session hijacking. In spoofing, there are different types. For example, the attacker can manipulate Address Resolution Protocol (ARP) messages and thus adjust the MAC addresses of the devices in his favor. DNS spoofing works similarly. However, it is not the MAC addresses that are the target, but the names and IP address affiliations. This enables the attacker to redirect his victims to websites he has created. In IP spoofing, the attacker pretends to use a legitimate IP address. This allows him to impersonate a trusted party, and his victims route traffic through him.

2. **SSL/TLS based:** In such an attack, the attacker tries to bypass the encryption and authentication of data communication between two parties. This can be done by obtaining a valid certificate (HTTPS interception). In addition, however, the attacker can also propose weaker, or no, encryption in the SSL/TLS handshake, so that he is able to bypass it.

3. **Border Gateway Protocol (BGP) based:** BGP is a core routing protocol used by internet service providers (ISPs) and large networks to enable the exchange of routing information and the determination of paths for data transmission on the internet. A Man-in-the-Middle (MITM) on BGP can cause traffic to be routed over unwanted or even malicious networks, resulting in privacy violations and performance issues. In this case, the attacker modifies the routing tables so that traffic is routed through himself.

4. **False Base Station (FBS) based:** This type of attack comes from the mobile communications sector. Here, the attacker provides fake access through a legitimate base tower, e.g., a radio tower or, in the computer sector, a Network Access Server (NAS), via which the victims connect. The attacker can thus monitor and possibly even manipulate all data traffic.

Against the above-mentioned MITM attack types, there are procedures that can significantly reduce the success of the attack. Depending on the type, the procedures have varying degrees of success. The first goal must be to prevent an attacker from being able to carry out an attack in the first place. To achieve this, security devices must be in place, maintained, and managed. These include, for example, security software to detect attacks or unknown patterns in the system, and firewalls.
Since an important goal of the attacker is the confidentiality of data, it remains essential to use strong encryption. For this reason, it is essential to stay up-to-date so that adjustments to procedures, or even the complete change of procedures, can be carried out successfully and securely. It is imperative to secure both the data itself and its transportation.
However, encrypting the data is not very successful if the attacker has successfully integrated himself into the communication. Therefore, encryption alone is not enough to fend off the attack. It is much more essential to check and ensure the authenticity of the other party. This can be done well via signatures, or better with certificates issued by a PKI. In this case, certificates are created with which the parties can mutually authenticate.
Another point of defense is the segmentation of networks and isolation of critical systems. A MITM attack can be made more difficult by limiting access to sensitive information and resources.
The last factor remains a major problem in larger networks. If users are poorly trained or untrained, warning signs are easily ignored, giving the attacker the opportunity to penetrate the best protected systems and execute successful attacks. One such warning sign could be a simple missing SSL certificate in the web browser. Once the attacker is in the system, it is still difficult to detect or even fight.
However, there is no measure that can eliminate an attack. Therefore, it will be necessary to combine all the above measures. The consequences of a successful MITM attack can be far-reaching. Therefore, it is significant that a system continuously faces danger and becomes better equipped.

2.6 Progressive Web Apps

A Progressive Web App (PWA) is a new form of web application. The goal was to provide a cross-platform development environment that combines the advantages of web-based applications and native applications. Therefore, this solution is often seen and named as a hybrid of the two application types. The advantages of web or native applications being united are the usability of hardware features, such as GPS, memory, or push notifications in native applications, and the increased interoperability of web applications. As a cross-platform solution, PWAs offers the possibility of being deployed on diverse devices. The screen display is automatically adapted. A significant advantage of PWAs is the possible installation from the web browser. This avoids the necessary installation via an installation file, which must be searched for in advance, or the installation via a native app store. This increases the application possibilities enormously, since only a web browser has to be supported. However, it also serves a higher purpose of convenience for the user, as the process of installation is shortened. This interoperability mostly exceeds the interoperability of native apps, even when developed with cross-platform development environments, such as React Native [214], Flutter from Google [102] or Microsoft's Xamarin [266]. [185][218]

A PWA can be used offline, unlike a web-only application. The user can therefore use the app even if the Internet is interrupted. This can be particularly advantageous in areas with poor Internet connections. The data is then processed offline by the user and subsequently synchronized if an Internet connection is active. As a cross-platform solution with very high interoperability, PWA development can be less expensive than a web or native application. Methods and functions provided to the user via PWA can be updated easily and quickly. In doing so, the user usually only needs to establish an Internet connection briefly for the synchronization to take place. This enables quick updates in the event of errors that could become potential attack vectors. [218]

In addition to the many advantages, the disadvantages should also be considered. If one decides to develop a PWA instead of a native application, one does not have the full potential of the hardware available. For example, a PWA cannot access the haptic sensors of a smartphone. In addition, the lack of presence in the app store can be interpreted as a lack of an advertising platform. Since people typically search for applications in these stores, their solution would not be found there. Interoperability is mainly ensured by web browsers. However, this also means that one is strongly dependent on their developments. Older browsers are poorly supported or not supported at all. In addition, there is a strong dependence on the companies that develop the browsers. If this company suspends support for your development, it can no longer be provided to the user. It is strongly recommended by developers that a PWA be developed with TLS or SSL connections only, meaning that the HTTP connection is encrypted. If one does not do this, then attacks can easily be carried out from the outside, which can have a direct impact on the user's hardware, depending on how the functions were programmed. This includes access to memory, cameras, or GPS systems. [185]

A PWA is composed of five integral components. Together, these enable the above benefits. All of them have different tasks to perform and form a PWA. It should be noted that not every browser supports each of these components. For example, the Push API and BackgroundSync API are not supported at all, or are supported by only a few versions of Safari.

The first important component of a PWA is the Service Worker (SW). This serves as a proxy for applications and networks and registers them there. It provides features such as caching application data, caching resources offline, and managing push notifications. When the user is offline, the service worker can access the cached data and deploy the application without a network connection. When the user is back online, the service worker can synchronize and update the data. A service worker typically runs in a separate JavaScript thread and can also run in the background when the application is not active. This allows the service worker to receive and process push notifications even when the application is not open. Registration is done using SW API and can interact with "fetch" or "push" events. In doing so, the SW waits for one of the commands to respond as previously programmed. [172][28]

In addition to the SW, there is the IndexedDB API. This is a NoSQL database in the browser that makes it possible to store and manage larger amounts of data permanently. Especially for offline functionality, this API is important because it enables the processing of the data, which should then be synchronized when the connection is active. [191][218]

The CacheStorage API is used for storing large resources. It can cache all data that is transferred via request or response over HTTP. According to the information, this data can be managed up to several hundred GB this way. Of course, this memory is bound to the limits of the hardware. This memory is used for caching purposes, so repeated access to web pages is faster. [173] The SW can help one access data from the server via CacheStorage or IndexedDB even without internet access. However, if one has edited data locally, which should then be loaded to the server as soon as the internet connection is established, one needs a Background Sync API. [9] In order for the PWA to send a push notification to the user even if the application is not open, a Push API is used. [186] The interaction of all the above interfaces allows the creation of a PWA with the strengths and weaknesses mentioned above. There are many ways to use it. In the work [218] the accessibility was examined. Moreover, a more complete overview of the strengths and weaknesses of PWAs can be found there.

2.7 Tripple-A (AAA) Technologies

"Many books were written on the topics of security and cryptography, bringing the dark and difficult secrets of fields such as public key cryptography to a public that typically was far less mathematically savvy than the original inventors. Many protocols and procedures were designed to realize infrastructures such as PKIs to bring these difficult concepts to life. Still, cryptographic algorithms or security protocols such as IPsec are not enough alone to operate a network that needs to generate services and revenues or to protect its constituency. Access to the network needs to be controlled. Users and devices need to be authorized for a variety of services and functions and often must pay for their usage. This is where the AAA protocols came in. In its simpler form a AAA protocol such as a base RADIUS protocol only provides authentication-based access control." [193, S. xvii f]
The Authentication, Authorization and Accounting (AAA) (pronounced triple-A) from the quote represents the tasks of authentication, authorization, and accounting. In this context, AAA is a framework for performing these tasks of the acronym in a network. Along with other security protocols and systems, it is an integral part of networks. The goal of the protocol is to control and monitor access to network resources. In addition, a billing system for any payments can be integrated.
The first A, *Authentication*, refers to the process of verifying the identity of a user or device to ensure that it is authorized to access the network. This is an essential part of AAA systems. And a significant one for IT security. Because of this, it is ensured that only authorized participants are admitted to the networks. In addition to authentication, the possibility of secure communication is also established. The authenticated entity would thus be able to exchange data securely with the server. Some methods of authentication are described in 2.3 Authentication. Most of the authentication methods presented describe unilateral authentication. This means that the client authenticates with the server, but not vice versa. This is said to be a vulnerability MITM attacks [64]. In many use cases, such unilateral authentication is sufficient, since the user can often trust the network (e.g., the work computer logs into the corporate network via cable). However, there is always discussion about whether mutual authentication is necessary for secure communication. Especially in wireless or mobile networks, MITM attacks should be easier to carry out [195]. Even with digital identities in SSI concepts, there are discussions about the need for mutual authentication. For example, in several cases, a successful MITM attack could be carried out due to a lack of mutual authentication [262]. For communication within authentication, two models are described in AAA. The model of Two-Party Authentication (2PA) and the model of Three-Party Authentication (3PA). The first model occurs in smaller networks or is needed for direct applications, such as the Internet Key Exchange (IKE). In larger networks, possibly in so-called multi-domain networks, the performance of the servers can easily be overloaded. In such cases, smaller intermediate hardware is used to intercept the queries and increase scalability. Such systems are called Network Access Server (NAS) and are essential in 3PA models for client communication. In 2.2 the architecture can be seen. In the diagram, two communication channels become clear. One is the channel from the user to the NAS, and the other is from the NAS to the server. We will see the 3PA model later in the architecture of RADIUS systems. But also in the integration possibilities of blockchain technologies in the RADIUS authentication process. [195]
The second 'A' describes the *Authorization*. Here, the respective authorizations for network resources are granted to an already authenticated person, or device. The authorization can take place on different levels, for example, at the file folder, or network level. The AAA server is integrated into the network architecture, so that authorizations can be queried (there are interfaces to LDAP, AD, etc.) or managed (own database for access management). [195]

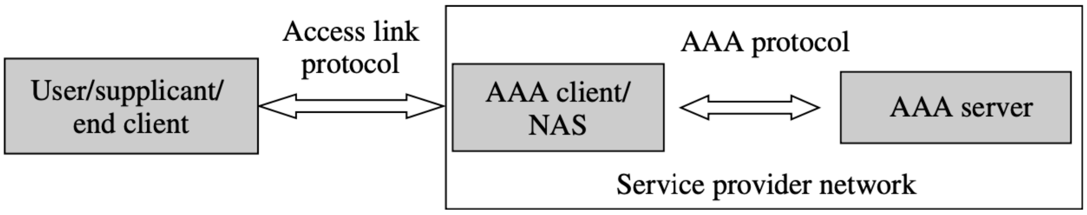


Figure 2.2: Three-party authentication model deploying an AAA infrastructure [195]

Accounting refers to monitoring and recording network activity of users or devices, including usage time and resources used. It can be used to track network consumption and account for network services. Three main tasks can be performed: auditing, cost allocation, and forecasting. An exciting feature here is the desired option for inter-domain billing. Within a company (intra domain), billing should be relatively easy to monitor. If one would like to accomplish now company-spreading (inter domain) an account, then a safe communication of the AAA servers must take place. In addition, it must be ensured that the account balances of the respective resources and users remain the same on both servers. This presents an enormous challenge, since communication and the integrity of the data must be secured. In addition, there must be trust in the network operator involved. The operator of a network is responsible for the correctness of the billing within his network. However, any errors could lead to problems elsewhere in the overall settlement. An example can be considered with mobile devices. If the user moves quickly with his cell phone, the radio network may have to be adjusted regularly. Information on the duration of calls must be recorded, and continued across operators. All of this is done without requiring the user to disconnect the line. The total cost of a call is the sum of all the partial costs of the radio network operators.
In accounting, collected data can be company secrets. Other network operators or even parties remote from the system should not be allowed to look into the actual accounting systems. Nevertheless, user policies (account policies) and actual accounting data are necessary in the exchange. This is ensured by employing cryptographic systems. [195]

AAA systems have an important function and are mainly found in cross-network systems. For example, in mobile networks or in WLAN accesses, i.e., larger networks with many accesses. Widely used implementations of AAA systems are TACACS+ (Terminal Access Controller Access Control Server Plus), RADIUS, and its successor, Diameter. All implementations have their advantages and disadvantages. TACACS+ is said to have a focus on device administration, while RADIUS was designed primarily for user authentication. Other differences include the communication protocols used. In TACACS+, the TCP protocol is used. In RADIUS, UDP is used. The TCP protocol is said to have higher reliability but lower performance than UDP. The fact that in RADIUS only the passwords are encrypted in the communication concludes a lower security in the RADIUS protocol compared to TACACS+. [121][209] In comparing RADIUS to its successor, Diameter, similar comparisons can be made. In Diameter, the TCP is also used for communication. In addition, Diameter scales better on busy networks. Meanwhile, however, RADIUS does better in scaling an authentication. In larger-scale networks, Diameter seems to be preferred over the RADIUS protocol. For example, in LTE mobile, [95]. A detailed description of Diameter can be found in the corresponding RFC document [96]. A performance comparison can be found in the paper [241]. However, this was written in 2008 and thus does not include many improvements to both protocols. With RadSec, the RADIUS protocol is provided with another protocol that enables a connection via TCP and TLS. This would eliminate this vulnerability. Due to the large number of implementation options, a general AAA architecture is useful to agree on goals to be implemented in such protocols. Such a general architecture can be found in RFC 2903 [126].

2.7.1 RADIUS Protocol

The Remote Authentication Dial-In User Service (RADIUS) is a network protocol that implements the tasks of AAA. As such, it serves the tasks of authentication, authorization, and accounting for users in a network and is implemented as a centralized client-server model. In this process, the user usually sends a username and password combination to the RADIUS server via a client. This server checks the available data and authorizes the user to access the network accordingly. If needed, additional information is collected and managed by the server for accounting purposes. An important function of RADIUS is the cross-network usage of the methods, which is called roaming. The exact performance and implementation of roaming are explained later in this chapter. To implement the functions, RADIUS has a centralized structure. In a single network, with the client-server model, and also in the implementation of a roaming service.

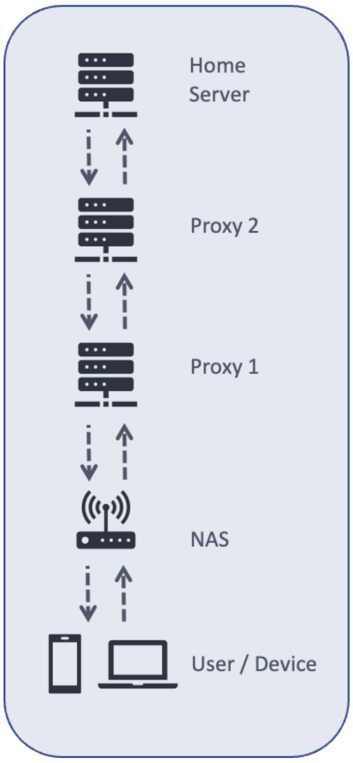


Figure 2.3: Representation of RADIUS roles in the roaming approach.

By implementing the AAA functions, RADIUS can be of enormous importance to network operators. First, it implements predefined user access to the network. Only authorized users with the respective defined rights are allowed into the network. This gives the application a great deal of responsibility for a secure implementation with integrity. However, such a system is also essential for any billing that may be required, for the successful handling of user billing with integrity, and thus potentially for ensuring the network operator's accounting.

At this point, it must be mentioned that there are deviations from the general understanding of terms when it comes to roles in the RADIUS protocol. A client is generally understood as RADIUS client or NAS. These are the devices with which the user tries to dial into the network with his (mobile) device. Such devices can be routers, switches, wireless access points, DSLAMs or VPN terminators [199]. The user, which is called client in many representations, is mostly called user or device in the RADIUS context. In this thesis, the usage of RADIUS client is tried, usually to make the context clear. Nevertheless, the reader has to look at the exact context to make it clear which client is meant. Usually, the user has a laptop, PC, or cell phone. Depending on the network, this can also differ. Besides the client and user, there is the RADIUS server. This receives the authentication request from the RADIUS client (NAS) and processes it. The answer is given back to the user via the client. For the implementation, access from the RADIUS server to a database is necessary. Various interfaces have been provided for this purpose. So access to AD and LDAP directories, an ordinary SQL database, or also a Kerberos service server can be set up [199].

For implementations of the RADIUS protocol, much standardization has already been created as Request for Comments (RFC) documents. In chapter 2.3 Authentication, a list of authentication mechanisms supported by RADIUS can be found (generally Password Authentication Protocol (PAP), Challenge-Handshake Authentication Protocol (CHAP) and Extensible Authentication Protocol (EAP) variations). Corresponding RFC standards are listed there. Other RFC standards describe integration, security, or even the general structure of AAA processes of RADIUS or even accounting [215]. A clear list of RFC documents concerning the RADIUS protocol can be found on the Wikipedia page [258].

In the section 2.7 Tripple-A (AAA) Technologies, reference was made to various AAA implementations, and the differences to RADIUS were discussed. It was found that the UDP transport protocol was preferred over TCP. The reasons for the UDP protocol were the higher flexibility to send the transaction quickly to another server when one server is busy and the simpler implementation of the multithreaded implementation at

that time. Today, however, RADIUS can also be used with TCP if needed. [220] Current discussions are concerned with how to implement communication in untrusted and insecure networks. IPsec and TLS are proposed for this purpose [76].
For communication, there are several other aspects to consider besides the transport protocol. For example, there is a request-response mechanism that describes and defines an initial authentication process. As reported at the beginning, the RADIUS protocol is centralized. Accordingly, the RADIUS client must connect to a RADIUS server and request access to the network. In the case of cross-network authentication (roaming), the RADIUS servers must also enter into a request-response mechanism with each other. The response is returned to the client after the servers have been checked. The packets are each designed as attribute-value pairs (e.g., JSON). Important fields are the username, user password, NAS IP address, and service type, depending on the communication type. A complete representation of packet types, structure, and general communication within RADIUS components can be found in [199, 220, 194].
One of RADIUS's strengths is what is known as roaming. This allows users from one location to access a network managed at another location. This is useful for frequently changing access locations. Such as employees with distributed duty locations or students with distributed campuses. Their access to the network can be efficiently managed by roaming with RADIUS. For the organization of the networks, this increases security, since access data is organized by the respective site administration and not left to the user. The user is thus provided with seamless and secure access to the network, which increases convenience in two ways. First, there is no need to manually select access points and enter data. Second, the user does not need to create new credentials, which reduces password management. [6]
To implement a roaming service, the RADIUS servers must be configured to communicate with each other. The server is then called a proxy server. In such a case, the RADIUS server first accessed by the user acts as NAS (client) and forwards the request to the next proxy or to the RADIUS server of the home network. In the figure 2.3 Representation of RADIUS roles in the roaming approach., one can trace such a concatenation with two proxy servers. [194, ch. 6.4.1]
A proxy performs the following six functions according to [6]:

1. **Authentication Forwarding:** An authentication and authorization request is forwarded as described above, and as shown in Figure 2.3 Representation of RADIUS roles in the roaming approach.. In this way, cross-network authentication and authorization are possible. Many NAS systems do not support proxy settings themselves. Therefore, RADIUS servers with proxy settings are required to implement forwarding. To ensure a smooth transaction, proxies try to maintain a connection once it has been established. However, a dropped connection can be compensated by redirection.

2. **Scalability Improvement:** The proxy chaining method described in RFC 2607 [6] establishes a roaming path hierarchically between the necessary participants. The hierarchical path favors scaling effects, especially in larger consortium networks, since no communication is required from all consortium participants, but only from participating participants. This reduces the necessary key generation and key exchange.

3. **Capabilities Adjustment:** In larger consortia, different features of the authentication and authorization processes can occur during roaming. Different data is supplied or required by the devices. The authentication methods could also differ in their implementation. The proxy servers should ensure that these differences are compensated for and adapted. In the process, formats must be translated, but in some cases, data must also be concealed from the home server for security reasons.

4. **Policy Implementation:** In a roaming situation, the proxy server may have predefined policies (e.g., access is programmed only for certain times). Such policies can be programmed into a proxy server. In this case, it can be considered a kind of Man-in-the-Middle (MITM).

5. **Accounting Reliability Improvement:** Proxy chaining in RADIUS secures the traffic of data packets sent within the proxy servers, clients, and the home server. Packet loss occurs due to the default use of the UDP protocol. RADIUS can accomplish this due to accounting reliability improvements.

6. **Atomic operation:** The RADIUS protocol implements so-called atomic processes. This ensures consistency among all parties. These operations are an essential part of the realization of reliability in accounting. To enable atomic operations, all parties must receive accounting data and confirm its receipt. Only then can the accounting be performed confidently and completely.

The scheme defined in [6] was only informal and thus not mandatory implemented in all RADIUS implementations. With the RFC 8559 [77], the most recent reference document was created. This is no longer informal, but a proposed standard. Accordingly, RADIUS implementations follow it.

The RADIUS protocol was initially designed to perform AAA tasks for intra-domain networks. In such a network, all RADIUS roles are represented in a network organized by an operator. As a result, this operator has complete control over all devices as viewed from the infrastructure. Further development, such as the aforementioned proxy chaining, made inter-domain networks possible. This allowed networks of different operators to communicate with each other and forward AAA requests. As such, as an inter-domain application, proxy servers cannot necessarily be trusted. The vulnerabilities mentioned in [6] due to the cross-domain feature, are:

- **Message Editing:** Since a proxy server is implemented in roaming like a MITM, it also has the possibility to act as an attacker. In doing so, it is possible to edit messages. For example, an access-accept can be changed to an access-reject, which could result in incorrect billing.

- **Attribute Editing:** If an authentication request arrives, a malicious proxy server operator could modify it to the extent that attributes are removed or added so that an easier authentication scheme to compromise is used.

- **Theft of Passwords:** Passwords are transmitted with a hash function, and a salt. However, the hash function used is not secure. Transmissions of passwords could therefore be at risk of detection.

- **Theft and Modification of Accounting Data:** As in the two cases of manipulating the transaction messages or the transmitted attributes, information on billing can also be manipulated. Since the billing data must be transmitted via the proxies, this would be easy to implement. It would have immediate consequences for a settlement.

- **Replay Attacks:** An authentication that has already been performed would be attempted again by an attacker here. This would allow the attacker access to the same resources as the original user. This could be implemented, for example, with a created token that is not marked as invalid after use. A proxy server would be well able to do this, since the tokens used are transported through it.

- **Connection Hijacking:** In hijacking, the fraudulent proxy server attempts to inject its own data into an authentication process to enable its own authentication.

- **Fraudulent Accounting:** In this type of attack, the fraudulent proxy server sends fake billing packets or session records, thus falsely collecting fees.

All of the above vulnerabilities can be mitigated in their damage potential with appropriate encryption and data transport protocols such as IPsec, SSL, and TCP. Furthermore, the latest RFC to RADIUS, the RFC 8559 [77], describes methods for secure proxy authentications that reduce the vulnerabilities' potential for damage. It also specifies that roaming should switch from the transport protocol UDP to TCP.

The vulnerabilities presented for roaming in RADIUS networks can also be used to derive vulnerabilities in general. The authentication method Password Authentication Protocol (PAP) in particular is at issue here, since the password is not encrypted even before it is sent. Since SSL allows the transport itself to be encrypted, as it must be in a roaming setup, the risk of disclosure is reduced.
Due to the centralistic structure of the architecture in RADIUS systems, a so-called single point of failure is quickly formed. This means that one can disrupt the availability of the AAA system's functions by selectively turning off one point of the RADIUS system. Disabling such a system is quite possible, for example, by means of DoS attacks. A general description of these attacks is given in [25]. Moreover, what measures can be taken to reduce such attacks in damage potential. The impact of DoS attacks can be seen from the large number of papers. Those discuss this problem in the context of RADIUS, and, in some cases, in more remote contexts such as Mobile Ad Hoc Networks (MANET) or more generally Wi-Fi network architectures. While most measures aim at detection of an attack and thus want to make the attack more difficult, there may also be ways to reduce the damage potential by (partially) distributing the tasks. This could make such an attack unprofitable. Such decentralization could be implemented by resilient dPKIs or also with DLTs. For information on resilient PKI, see 2.4 Public Key Infrastructure and 6.4.2 Public Key Infrastructure vs. decentralized Public Key Infrastructure. For more information on the approach with DLTs, see 4 Methodology and Conception. [128, 70, 220, 26]
Another vulnerability whose potential impact was described above is the Man-in-the-Middle (MITM) attack. Induced by the hierarchical structure and favored by the desire to keep a roaming line constant and not change it, such an attack is inherent. Again, mentions in many scientific and journalistic articles are evidence of the potential danger. These papers are about avoiding the danger from MITM attacks, for example, in improving the EAP authentication methods used [182]. But also PAP based methods [268] could be improved in the past. In the paper [140], a vulnerability in Microsoft Challenge-Handshake Authentication Protocol, Version 2 (MS-CHAPv2) was exploited to eavesdrop on messages within a VPN connection between client and server. A much more recent publication also highlights the timeliness of the issue. In [71], it is described that a spoofed packet injection attack provides for a secure AAA implementation through improved security modules in the authentication PolicyFlow. In the work [45], a vulnerability was exploited in the RADIUS implementation of eduroam to steal an identity as MITM. A total of five measures are presented as countermeasures. Three measures are client-level configurations, and two are infrastructure-level configurations. Not all measures sing equally effective in preventing a MITM attack. However, the proposal is to build a more distributed PKI that is responsible for only one network at a time.

The RADIUS protocol has evolved into a modular system over the many years of development. This has made it possible to extend and customize the system in a simple and efficient way. These extension modules are independent software components that can be integrated into the RADIUS server. This independence allows other teams to separately extend the range of functions. Moreover, the maintenance of these components can be done independently of the further development of RADIUS. The administrator of a network has the possibility of setting up the required modules. A list of available modules is provided by the manufacturer of the RADIUS client. For example, with freeRADIUS each of the modules described in chapter 2.3 Authentication is customizable as a module. Other modules allow for additional authentication methods or database systems. Even whole network authentication systems like Kerberos can already be integrated as a module into a network implemented by RADIUS. It can also be envisioned that scalability and security can be increased as modules are developed and integrated for more efficient authentication and communication. Moreover, an update of an extension module has no direct impact or consequences on the RADIUS server, which can continue to run with other modules.
Thus, in summary, the advantages of modularity allow for great *adaptability* by enabling the implementation of specific requirements of the network. A high *scalability*, through possible integration of more efficient algorithms. In addition, easy *maintenance* is enabled by modular updates. And probably the most

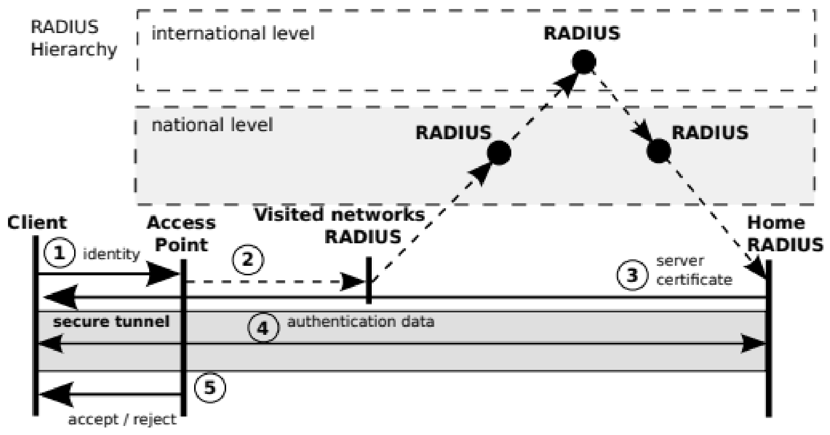


Figure 2.4: Authentication Process in eduroam [45]

obvious advantage is the *extensibility*. In addition to the advantages, the disadvantages must also be considered. For example, modularity increases the *complexity* of system integration and management. The administrators of a network must keep track of and manage various modules that may interact with each other. The increased complexity can lead to *security risks*, e.g., due to misconfigured modules. Furthermore, some modules may negatively affect each other by being incompatible. This could degrade the performance of the RADIUS system. The high complexity also demands higher maintenance and support efforts.
A list of possible modules for freeRADIUS can be found on the website [107]. Among them, you can find, e.g., DHCP, YubiKey, or the already mentioned Kerberos module.

Application Example - eduroam

The eduroam project (short for "education roaming") is an international, secure, and easy-to-use roaming service for the research and education communities. It allows users of participating institutions to automatically connect to the eduroam WLAN using their home institution credentials when they visit other participating institutions. This eliminates the need for them to request temporary credentials or create separate accounts. The eduroam network was originally developed by the TERENA Task Force on Mobility, which later became the GÉANT Association, and launched in Europe in 2003 [101]. GÉANT is the pan-European research and education network that connects higher education networks at the national level. Since then, eduroam has spread worldwide and is available in over 100 countries. It says it performs over 500,000 international authentications every day. The system is based on WPA2 enterprise security technology and uses a hierarchical structure of Remote Authentication Dial-In User Service (RADIUS) servers to authenticate and authorize users. In Figure 2.4 Authentication Process in eduroam, the hierarchy can be traced.
The hierarchy in RADIUS networks such as eduroam arises when roaming is implemented. Users can be identified by their access data. A network is divided into so-called "realms." In addition, there is the so-called Top-Level-Domain (TLD). The access then has the format <User>@$realm.tld$. The TLD tells the RADIUS server to the top-level servers, which forwards one to the respective home university servers. The concept can be seen in the 2.4 figure in the "RADIUS Hierarchy" description. Based on this connection, a transfer of authentication from the remote home network to the visited remote network is enabled. [45, 116]

Chapter 3

Current Trends and Development

3.1 Authentication Methods

In chapter 2.3 Authentication on page 32, we clarified the basics of authentication. In this section, we briefly introduce various authentication methods. Essential to understanding the thesis will be the methods used in the Remote Authentication Dial-In User Service (RADIUS) protocol. However, since newer methods also provide good ideas for improvement, these should be considered and included in your development.
Since the basic technology is always related to cryptography, terms from this area are used here. Among them, for example, are public and secret keys or cryptographic hash functions. You can read about these in chapter 2.1 Cryptography on page 5.
The authentication methods supported in RADIUS can be roughly divided into three variations. There is the relatively simple Password Authentication Protocol (PAP). An extension of this is the Challenge-Handshake Authentication Protocol (CHAP). In addition, there is the class of Extensible Authentication Protocol (EAP)methods [238]. An overview created with short descriptions [200]:

1. **Password Authentication Protocol (PAP)** [239]: This method transmits the username and password in clear text. Therefore, it is not recommended to use it for security reasons. Attackers could intercept or manipulate the data.

2. **Challenge-Handshake Authentication Protocol (CHAP)** [156]: This is a more secure method than PAP because it uses a challenge-response loop method. The server sends a random string to the client. The client uses its password and a hash function to create a response. By repeating the procedure, authentication can be maintained.

3. **Microsoft Challenge-Handshake Authentication Protocol, Version 1 (MS-CHAPv1)**: This method is older than CHAP, but works similarly. However, a fixed password is used here, which allows an attacker to find it out by trying repeatedly. This would compromise the password. For this reason, it is better to use MS-CHAPv2.

4. **Microsoft Challenge-Handshake Authentication Protocol, Version 2 (MS-CHAPv2)**: A more secure version of MS-CHAPv1 which uses stronger encryption. It is often used with EAP and is a common authentication method for VPN connections.

5. **Protected Extensible Authentication Protocol (PEAP)** [5]: This protocol is an EAP-based protocol. It is thus widely used in wireless networks and VPN connections. The protocol allows increasing security by using Transport Layer Security (TLS) or Secure Sockets Layer (SSL) for authentication communication.

6. **EAP-Tunneled Transport Layer Security (EAP-TTLS)** [111]: This method is also an EAP-based authentication. This makes it suitable for wireless networks and VPN connections, just like PEAP and other EAP methods. The main difference to PEAP is the security architecture. With EAP-TTLS, in addition to a secure TLS connection, a tunnel is also established for authentication. This further secures the data. Nevertheless, this method is less widespread than PEAP.

7. **EAP Generic Token Card (EAP-GTC)** [40]: With this authentication method, a token is created for authentication. This token is passed to the user. This method is often used in RADIUS servers.

8. **EAP Message Digest 5 (EAP-MD5):** This EAP-based method uses a cryptographic hash function MD5 to protect the user's password. Here, the method is structured as a challenge-respond. The server sends a random value to the client. The client then calculates the hash value of this number with the password.

Besides the RADIUS protocol, which was developed as an implementation of an Authentication, Authorization and Accounting (AAA) system for wireless networks and VPN connections, many other authentication methods exist. Each of these methods has its own strengths and is thus used in other application areas. The following is intended to provide a non-exhaustive listing with various methods:

- **HTTP Authentication:** This Request for Comments (RFC) provides a standardized method for authenticating HTTP clients. Two main methods are defined: Basic and digest authentication. In client authentication, username and password are transmitted in clear text. In the digest variant, the password is encrypted in advance. Thus, client authentication should only be used if at least the transport is encrypted (HTTPS).[105][98]

- **The Transport Layer Security (TLS) Protocol**: This document defines a procedure that secures data traffic on the Internet. Eavesdropping, tampering, or message forgery are to be avoided. Both communication partners must authenticate themselves to each other. Common cryptographic systems are used for this purpose, such as Rivest–Shamir–Adleman (RSA), Elliptic Curve Digital Signature Algorithm (ECDSA) or pre-shared key procedures.

- **Generic Security Service Application Program Interface (GSS-API)**: This RFC defines and describes a channel binding to increase communication security. It is used, for example, in Kerberos systems[10] or in SSH connections[147]. Overall, one can find an implementation of this reference in a variety of network protocols and applications. [260]

- **Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)**: This RFC describes an architecture of authentication in so-called Global System for Mobile Communications (GSM) networks. GSM is the second-generation standard for telecommunication networks. Besides authentication, the key communication is described, which just describes the vulnerable key exchange. In addition, the protocol also describes an AAAimplementation. [222]

- **Kerberos:** Kerberos is a network protocol for authenticating users in (distributed) networks. It is considered a more secure, but also more complex, protocol than RADIUS. It uses a ticket authentication method, where the user receives a ticket from a ticket-granting server (TGS) to access other resources on the network. [201]

- **WebAuthn (Web Authentication):** WebAuthn is an open web standard that provides a user-friendly and secure method for authenticating users in web applications. It is designed to replace password authentication. This is because password authentication is vulnerable to phishing attacks and other security threats. Instead, WebAuthn allows users to be authenticated using various authentication methods, including fingerprint scanners, facial recognition, security keys, and other biometric factors. It uses a signature method in which the user creates a private key. With this, the user can answer a challenge from the server and confirm their authenticity.

- **Open Authorization (OAuth):** This protocol enables cross-website login without having to enter further user data. Well-known examples are the log-in functions of Twitter, Google, Facebook, and others. It is sufficient to log in to the respective services (also called Identity Provider (IdP)) and then refer the new service to the IdP. Successful authentication is done by requesting the IdP from the user. [130]

- **SAML (Security Assertion Markup Language):** SAML is an XML-based protocol for the secure transfer of authentication information between different systems. It allows users to log in to an IdP once and then gain access to multiple services and applications provided by different Service Provider (SP) platforms. To implement it, various protocols and standards are used, including XML Signature and XML Encryption, to ensure the security of authentication information. It is also capable of supporting various authentication methods, such as username/password, two-factor authentication, and Single Sign-On (SSO). [57]

The latter methods (WebAuthn, OAuth and SAML) are typically used in web applications. Therefore, the architecture of these methods differs from network-level authentication systems. Since the RADIUS protocol can also be used in roaming, the boundaries of both authentication architectures disappear here. In addition, RADIUS can be easily extended to include other authentication systems. More information about RADIUS and its extension potentials can be found in chapter 2.7.1 RADIUS Protocol.

3.2 Digitalization and Blockchains for Public Administration

The perceived increasing digitization of the economy and society is leading to a demand for digitization from public administrations. Efficiency and simplification should be the main advantages. This digitalization should initially be considered independently of blockchain technology.
Digitization in the Federal Republic of Germany is currently not well advanced. Governments are often criticized for this. Especially in comparison to other countries, Germany is regularly in a bad position. [97, 82] The focus in this overview will be on German administrations due to the research cooperation with the Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD).
The Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD) is a central IT service provider for the state administration of Hesse, Germany. They provide various services, such as software development, infrastructure and network provision, data security, data protection, and IT consulting. It also facilitates the implementation of digital transformation within the realm of public administration.
The government is taking various measures to improve the situation. One identified reason for poor digitization is the federal structure of the Federal Republic of Germany. Each federal state works on its solutions. This increases complexity and makes cross-state processes more difficult. The "once-only" principle attempts to reduce the administrative effort for citizens and companies and increase the efficiency of public administration. This is achieved by requiring citizens and businesses to provide information to the public administration only once, and then allowing this data to be reused within the administration for different services without the citizen or business having to provide it again. Germany, like other EU member states, has sought to implement this principle as part of digitization strategies to improve services for citizens and increase efficiency in public administration. However, this principle raises data protection concerns that need to be addressed during implementation.

In addition to the once-only principle, there is also the Online Access Act (German: Onlinezugangsgesetz) (OZG). This law is intended to digitize Germany's administrative services. The law was originally scheduled to expire at the end of 2022, but was initially extended for an as yet undetermined period. The OZG provides for the federal, state and local governments to offer their administrative services via a joint portal in order to simplify access to these services for citizens and companies. These include, in particular, services such as applying for identification documents, registering a trade, or applying for subsidies. The OZG is part of the broader digitization strategy of the Federal Republic of Germany and is in line with EU initiatives to digitize public administration. It can be seen as an instrument that promotes principles such as the "once-only" principle in practical implementation. [85]
All digitization efforts require authentication. If a German citizen wants to apply for a new ID card, it must be ensured that the applicant is actually the person making the application. For this authentication, the new ID card was introduced in Germany, which enables authentication on the Internet via a corresponding card reader. In addition, there is the electronic IDentification, Authentication and trust Services (eIDAS) regulation at the European level. This regulation provides a legal framework for electronic authentication and trust services for electronic transactions in the single European market.
The actual reasons for the slow or nonexistent progress of digitization are manifold. The aforementioned federal structure is just one reason. However, other federal countries, such as Denmark, are further along in the process. The main reasons are the administrative culture and the administrative structure. Germany, for example, has shown a strong focus on legal certainty at the expense of user-friendliness. And the aforementioned federal structure could not be efficiently circumvented either. This could be implemented through appropriate governance. In well-implemented governance, decentralized participants from all federal states or municipalities would take part in decision-making processes involving reusable processes and products. This could be setting open standards for data transfers or determining regulations. Other areas would be developed and determined by the respective states and municipalities. [112, 85]
Even though digitization can bring advantages, it also harbors dangers. It is already difficult to find good personnel for IT and other areas. Digitizing processes will increase the need for IT specialists. The more efficient performance of simple tasks will thus be complicated by digitization in terms of maintenance and further development. A lack of further development can lead to not revising existing processes further. Should the technology evolve, this could lead to an outgrowth of skilled personnel and put a brake on further development. [158]

Blockchain technology is said to have the potential to significantly influence and improve the digitization of administrative processes in Germany. The integration of blockchain can lead to an increase in transparency, efficiency, and security due to its characteristics. As a part of digitization, blockchain can be considered with similar considerations as shown earlier. Its use promises many benefits, but on the other hand, it also brings dangers. As discussed in the chapter 2.2.6 Problems, Disadvantages, and Concerns of Blockchain Technologies, the blockchain itself poses some problems that can impact the digitization process. Digitization is necessary for the use of a blockchain. This is because the blockchain itself involves the digital processing of data. However, the processes would not need to be digitized in advance. The digitization of a process can be done from conception with blockchain technology. Considering this, there were considerations to use the technology as profitably as possible within the framework of the OZG.
The decentralized nature of the technology requires distributed users of the network to get together and discuss the architecture. This is done in an attempt to map certain processes. Govdigital has been formed in Germany for this process. This is a cooperative of municipalities, counties, states, and other public institutions in Germany. It was founded to promote digitization in public administration, considering the special requirements and challenges of the public sector. The aim is to create cooperative ventures among the participants and counteract potential competition. In this cooperative, a blockchain infrastructure was also formed that could be used for projects. In the sense of a nationwide architecture. However, due to failed

blockchain projects at the federal level in Germany, there was a reduction in funding for further projects. The work was not completely shut down, but was integrated into other areas of work. [123, 131]
But at this point, a separate warning about the dangers must be issued. Since critical and personal data Personally Identifiable Information (PII) are processed in numerous instances in the processes of public administrations, it is necessary to protect them. It must not be possible for the data to be siphoned off by maladjustment. Therefore, there is a high need for protection. However, if the possible processes are dependent on the blockchain architecture, which in turn is dependent on the projects and the existing data, this creates a mutually influencing dependency that complicates the complexity of finding and securing the architecture. This circumstance makes it difficult for public institutions to stay ahead of developments in the market.

3.2.1 Blockchain Developments for Public Administrations

In the Federal Republic of Germany, as in other countries, pilot projects have been carried out to test the use of a blockchain for public administrations. This section presents some prominent projects.
In the asylum process for asylum seekers, a pilot project using blockchain took place. The goal of this project was to both speed up and simplify the processes that run across different authorities. To make this possible, a blockchain was to be used. This should ensure the self-sufficiency of the respective authorities and make the transfer of documents audit-proof. The respective authorities are the Federal Office for Migration and Refugees (German: Bundesamt für Migration und Flüchtlinge) (BAMF), the Central Authority for Foreign Nationals (German: Zentrale Ausländerbehörde) (ZAB)[1] and the Federal Police. All agencies managed their peers and formed a blockchain. The system itself was built based on the Self-Sovereign Identity (SSI) principle. According to this, asylum seekers have a mobile device with their PII on it, and an attestation is provided with the blockchain. These attestations are implemented with cryptographic processes so that signatures and Zero-Knowledge Proofs (ZKPs) ensure their integrity and correctness. In this way, a digital identity is made possible, which is in the administration of the user and conforms to data protection rules. The problems of transparency and the Oracle problem are to be mitigated by this. A verifying body can now obtain the necessary information from the asylum seeker and verify it with the blockchain. This SSI principle is used in many pilot projects, as we will see. In general, a SSI infrastructure does not necessarily need a blockchain. A Public Key Infrastructure (PKI) could equally be used for verification.
The pilot project demonstrated that an asylum process could be carried out quickly and verifiably in an exemplary manner. The legal safeguarding of digital signatures and interoperability beyond the state of Saxony were still problems. However, these issues also exist in other areas of digitization. [261, 14]
The next feasibility project can be found in the tax administration. Here, SSI blockchain-based digital identities are to form a digital tax ID. Tax authorities would thus be able to efficiently provide online retailers and marketplaces with an ID, which in turn can be used for their tax returns. It is said that this serves to prevent tax leakage. The current implementation of the process relies on paper documents, resulting in high complexity and the possibility of easier fraud. The study demonstrated the technical feasibility and also described and examined the economic and regulatory implementation. [167]

In addition to the aforementioned projects, each of which was developed and studied in controlled and small laboratory environments, there were also projects that could be tested by the German company. As a first project, a SSI with blockchain infrastructure should map a digital driver's license. The holder of the driver's license should thus be able to have a digital copy on his cell phone. Police and other inspection agencies can then use the blockchain to ensure the integrity and accuracy of this data. The whole process should be easy to use for the respective users, who scan QR codes with their devices and execute the respective actions by calling buttons. However, the developed version of ID Wallet showed that some configurations were not

[1]Depending on the federal state, this authority can change. The authority used in the prototype was the ZAB.

implemented well until then. For example, Domain Name System (DNS) entries were publicly visible, or the entity that was supposed to check one's data did not have to identify itself. These and other bugs quickly caused the app to disappear from the public app stores. [23, 42, 29]
Also, going with a SSI infrastructure should be the solution presented by the Bundesdruckerei for the implementation of certificates. Thereby, the certificates are not to be limited only to those of the school. Much more infrastructure is to be created to ensure testimony and examination. But also here, serious weak points were not closed. Thus, open Application Programming Interface (API) interfaces or serious XSS vulnerabilities could also be found and exploited. This made it possible to create certificates that looked like the original certificates issued. The suspicion was that false data could also be written to the blockchain. In addition, the only access to the blockchain data was through a service provided by the Bundesdruckerei, which mitigated the property of decentralization. [164, 127]

The failed projects lead one to conclude that blockchains may not be suitable for public administrations. But if you look at the reasons for the failure, you will see that it was often due to the wrong conception and implementation, which were detached from the blockchain technology itself. Other projects would also have failed under the same conditions. Thus, no immediate impact of the technology on public administrations can currently be concluded. Nevertheless, the projects give us insights and knowledge to consider for future developments. Whether public administrations should move forward so strongly in exploring technology is one of the insights to be debated.

3.3 AAA-Technologies on Blockchains - Related Work

With the rise of cloud computing, cross-enterprise collaboration is also becoming more prominent. Along with this independence from certain system requirements, reliable authentication and authorization of the participants are essential. An optional billing system could be integrated here to ensure usage-based billing.
In this section, I will give an overview of already-done AAA and RADIUS implementations on and with the blockchain. We will see that there are different implementation options. I will distinguish not only the blockchain technologies themselves, but also the level of implementation of the AAA system. In chapters 4.1 Blockchain and RADIUS Interface Possibilities and 4.2 Schemes for Authentication, the different options are discussed.
Currently, used systems, such as RADIUS, are characterized by a strict hierarchical structure that raises security concerns. Using man-in-the-middle attacks, the central authorization unit could be compromised or disabled by Denial of Service (DoS) attacks. Blockchain technology, described in 2.2 Blockchain Technologies on page 10, could offer a decentralized alternative here. Initial approaches using Ethereum (ETH) and its ability to support smart contracts have already been presented ([180] and [152]). However, these approaches are based solely on blockchain technology, which could replace the conventional AAA service. A possible technology switch to another blockchain platform was not considered, leading to an increased dependency on Ethereum. The models presented in this paper have incorporated these insights into their development.
So-called mesh networks are often implemented using blockchains. Mesh networks are good examples of applications for AAA protocols. In such a network, computing nodes are connected to each other to pass data between them. The network may or may not be fully connected. A widely used example application is WLAN mesh networks, which provide access for devices to the WLAN infrastructure. This can provide access when there is a high load at one location, but also over greater distances and possibly in institutions. Through such a mesh network, the eduroam project enables students and staff from participating universities to access the Internet at any other of these locations with their respective information. In doing so, they do not need to actively re-authenticate, as the process runs automatically in the background [116].

A custom blockchain-based authentication, to extend mesh networks for WLANs, can be found in [153]. Here, a RADIUS server is used for AAA tasks on the network. Described is an authentication from a foreign device (like a cell phone or laptop) and an Access Point (AP) to the system. In this case, the request is first searched internally. In the event of missing information, an exchange with the blockchain will take place, so that cross-institutional and decentralized access data can be provided in the future. The blockchain technology is not described in detail here, and therefore the architecture is rather general. It should be noted here that the SSI technology was not yet widely known, and therefore a separate authentication was developed. The described system resembles a SSI implementation in such a way that mutual authentication is described, which can then be accessed. In SSI context, the procedure is enabled by Verifiable Credentials (VC). Even if not explicitly mentioned, the character of a Web of Trust (WoT) becomes clear, since the described system extends and defines the hierarchical architecture of authentication systems in mesh networks with a decentralized solution. In the described results, RADIUS and RadSec are examined separately. According to the company's own information, however, RADIUS is now equipped with RadSec, so from today's perspective, a separate evaluation does not make sense. However, the result that RadSec is supposed to perform worse in larger distributed systems compared to systems with blockchains remains exciting to assess. Unfortunately, little is explained about the blockchain technology used, making it difficult to reproduce the results and leaving many questions unanswered, such as whether consensus is enabled.

In the work of Helebrandt, Ries and Khilenko [135], a system is described that can take over the control and monitoring of AP. Device configuration updates are stored encrypted in the blockchain and made available again for APs. Administrators have to authenticate themselves by means of digital certificates and can then have the configuration updates applied, depending on their authorizations. The blockchain technology used is Hyperledger Composer, which is said to have better scaling compared to a private instance of Ethereum. In addition, Rest APIs are automatically provided, which can be accessed from the server using TFTP. The server is a Cisco router and thus interfaces with the network. This system does not require an AAA implementation such as RADIUS or DIAMETER. The access requirements for users who want to connect to the network via the router are thus encrypted in the configuration on the blockchain. This is initially contrary to common practice, as the convention in identity projects in particular is that no personal data should be stored on the blockchain. The server now can access credentials in a decentralized manner, instead of from the local disk. [135]

Another project that describes the interface of an AP with a blockchain is [47]. The goal here is to develop a system that uses a Proof-of-Work (PoW) blockchain so that an overload of the AP by too many requests is counteracted. This will use consensus as a preventative, but also develop a new authentication scheme that can handle fewer message exchanges than is common in the RADIUS network. The use of a blockchain is also intended to potentially increase privacy, through possible anonymous authentication.

Trustroam [175] is a project that directly addresses the vulnerabilities created by centralization in eduroam (such as Man-in-the-Middle (MITM), single point of failure, and privacy breaches) and implements a decentralized solution using a private Ethereum blockchain. At the blockchain level, a dedicated authentication system is implemented using Solidity. The architecture specified by eduroam will be extended so that access requests in the RADIUS server network will no longer be handled in communication with other RADIUS servers, but via the blockchain using the custom-developed smart contract. The proposed system is tested in a test environment and compared with the RADIUS protocol. An expected result was the higher number of packets in registration processes compared to a RADIUS system. Surprisingly, however, the Trustroam system based on the Delegated Proof-of-Stake (DPoS) consensus method showed faster performance and reduced latency in the lab. In contrast, the PoW-based system performed significantly worse because the blocks are generated with a probability distribution.

In [181], an AAA solution is implemented to grant access to software systems. In doing so, the entire business logic specified by AAA is implemented using smart contracts on a blockchain. This work illustrates that other application domains such as SSO can be implemented in addition to wireless network access. From the

results, it is clear that (partially) public blockchain systems may have transparency issues due to their public access. Here, we describe an authentication method that uses a hash function to reduce the risk of strangers seeing the data. Continuously new passwords make it difficult to link different requests from a user.
In [235], a project is presented that aims to fully implement an AAA system on a blockchain. The focus is on the abstract and general definitions of heterogeneous networks (HetNet). The architecture of this network is to be extended by a decentralized element that enables the application of AAA-systems on blockchains. Cellular networks could serve as an example of such a HetNet. Within a radio tower, not only can cell phones with different protocols log on to use services such as phone calls, SMS, or Internet access, but other devices, such as landline phones or fax machines, can connect or at least transmit data.
[151] presents a solution for Internet of Things (IoT) devices. Using Hyperledger Fabric, a blockchain solution is developed with side-chains. These side-chains represent the infrastructure of a particular domain. The presented solution allows registering IoT-devices within a certain domain. While a billing functionality was not explicitly addressed in the paper, it could be easily integrated through the use of a RADIUS server. The blockchain solution is implemented mainly to avoid single points of failures and to prevent unwanted DoS attacks by requests from IoT devices. Especially, the second aspect contributes to the fact that the proposed blockchain solution scales better compared to traditional centralized systems, especially when the authentication requests increase exponentially due to the increasing number of devices. Another advantage of the proposed solution is the reduced number of communication steps in the authorization process. Instead of forwarding multiple requests to centralized systems, the required information can be accessed directly in the node in the proposed solution.
In Software-Defined Networking (SDN), the architecture of network management models changes by separating the data plane from the control plane. To ensure secure communication, a concept was presented in [137] that proposes an implementation of AAA on the blockchain. An AAA system serves as the primary backup. However, as has been noted many times, these systems also have vulnerabilities that the blockchain is designed to replace. The proposed solution is divided into different modules with respective tasks. For authentication, a Certification Authority (CA) is to be implemented via the blockchain. Communication is done via REST interfaces upstream of the modules. Data for authentication is stored directly on the blockchain. In addition, there are Access Control List (ACL)s that control and monitor the respective accesses.
For the realization of an Industrial Internet of Things (IIoT) system, many protocols, and interface descriptions are needed for a smooth process. Modbus is a protocol for these tasks. With it, tasks such as robotics, artificial intelligence, and intelligent decentralized manufacturing are applied. With the work [99], an authorization and authentication solution is technically implemented on the blockchain. This should allow the two AA tasks to be provided to IIoT devices. The system was implemented using Hyperledger Fabric with the RAFT consensus process. The AA system was built according to the SSI principle.
Also for IoT systems, an approach with so-called Non-Fungible Tokens (NFTs)[2] in the work [231]. This will be based on Ethereum (ETH). The NFTs here are to implement various access control mechanisms. Role-based, attribute-based, as well as various other access controls are supported. Accounting is implemented in this solution by means of a separate token. This is an additionally provided smart contract.

[2]see chapter 2.2.3 Ethereum and Blockchains 2.0

Chapter 4

Methodology and Conception

To address the research inquiries posed in the introduction, a Design Science Research (DSR) approach is employed as the research methodology. In information systems research and related disciplines, the DSR is a research method used to develop innovative and practical solutions to real-world problems while advancing theoretical knowledge. The DSR approach is often seen as a bridge between theory and practice, as it aims to both develop practical solutions and expand theoretical knowledge. [78]
Through the close cooperation with the research partner Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD), with the aim of realizing an in-house development with interfaces to HZD systems, results in an intensive linking of theory and practice in the research project. As this is explicitly supported and required in DSR, this research methodology aligns with the subsequent process of this work.
The approach adapted to the thesis consists of the following steps:

- Problem identification: The research questions posed in the introduction are spread over several aspects. *Can the RADIUS protocol be used on blockchain?* and *Evaluation of blockchain technologies for public administrations*. For the latter, the state of development will be presented and potential strengths and weaknesses identified.

- Definition of the objectives: First, the current state of developments will be reviewed. The focus is on the advancement of Remote Authentication Dial-In User Service (RADIUS) protocols, as well as the advancement of blockchains and Distributed Ledger Technologies (DLTs). Existing combination research will also be elaborated on. The second objective is to design and implement a framework that can realize the RADIUS protocol on blockchain. The outcomes of the research study will be compared to those of the advancement of the RADIUS protocol and DLTs identified and debated. Thereby, the questions regarding the application of blockchain for public administration will be answered.

- Design and development: To achieve the defined objectives, a systematic literature review is conducted first. Various sources are used for this purpose. The list includes the Google Scholar, IEEE, Springer, ACME, and Elicit databases. The latter is an AI-assisted service that helps users find appropriate scientific literature. Blockchain for eduroam, Blockchain for radius, and Blockchain in public administration were some search queries used. Furthermore, a selection of literary works was made for the respective technological advancements. Sources were used that were related to the author's own research and met the criteria for good papers. Those criteria include the number of foreign references, the time of publication, and the number of conference reviews.

  The classification is followed by the design and implementation of the RADIUS protocol on the blockchain. This design and implementation is called AAA-me. The conceptualization is described in 4.7.3 Architecture of AAA-me. The implementation is shown and described in 5 Implementation. The measurement of AAA-me is then compared in a laboratory experiment using a RADIUS system. The concept of the experiment is described in 5.2 Experiments.

- DEMONSTRATION: The implementation of AAA-me and the performance of the AAA-me experiment can be found in chapter 5.1.1 Architektur.

- EVALUATION: To conclude, chapter 6 Results and Discussion compiles all the results.

There are two other aspects of DSR. The first additional aspect is communication; research results are communicated and discussed with experts and stakeholders. Results from these discussions are incorporated into the development process. Iterative discussions with the HZD and a research network of research institutions worldwide called Bloxberg took place continuously during development. The latter was used to facilitate university research exchange and to obtain additional perspectives. The second additional aspect of DSR is keeping track of the research results. This is done by publishing this dissertation.
At this point, we will recall the terminology regarding blockchain programs mentioned in the introduction. The term “smart contract” is extensively utilized in the literature. But in this work, the term "blockchain program" is often used to describe smart contracts. To avoid any confusion, it will be carefully chosen which of the two terms will be applied, depending on the context.

4.1 Blockchain and RADIUS Interface Possibilities

Several options exist for implementing RADIUS on a blockchain, which are discussed in greater detail in this section. Choosing the ideal interface for a particular application involves a multitude of considerations. These factors include the architecture of the blockchain, the structure of internal authentication, the number of users and transactions involved, and the distribution of proprietary systems.
It should be noted that, regarding RADIUS, a client is the device allowing access to the network. It is not the device used by end users. To avoid misunderstandings, clients used in RADIUS protocols are called RADIUS clients. These are typically modems, (wireless) Access Points (APs), switches, or generally Network Access Server (NAS). More information about RADIUS in chapter 2.7 Tripple-A (AAA) Technologies.
Multiple options for RADIUS blockchain interfaces exist, listed in the following.

a The RADIUS functionalities are fully implemented within the blockchain. In this case, smart contracts would be able to perform the respective authentication and authorization steps. The data would also be available on the blockchain in some encrypted form. The RADIUS client would send the request directly to the blockchain and receive the response directly. Key information and scheme designs would be stored on the blockchain (for basic information about PKI, see 2.4 Public Key Infrastructure).

b The blockchain is used as a complementary technology. In this case, the blockchain is used to send requests to the RADIUS servers. Furthermore, authentication would be performed through the blockchain. The range of functions of the RADIUS servers could therefore be expanded to include additional authentication methods.

c The blockchain is used exclusively for communication between the RADIUS servers. The RADIUS implementation specifies the protocol of messages, and the blockchain would only exchange information, like authentication request forwarding to another RADIUS server.

This listing indicates varying degrees of integration with blockchain technology. The first proposal (a) uses blockchain the most. Networks with no existing RADIUS system and where many clients have different types of hardware are good for this type of system (CPU, memory, and hard drive). Devices like this could be present in an Internet of Things (IoT) environment. Moreover, IoT devices might not be manually controlled by a human. They would be controlled by programmed-in routines. Clients would be able to connect to an Authentication, Authorization and Accounting (AAA) system without having to build an AAA architecture like RADIUS. However, if one has already implemented a RADIUS, this system would be congruent with the blockchain solution and result in double expenses, bringing economic efficiency into question. This implementation has the biggest negative effects. Scaling is the most serious limitation for blockchains and must be considered in the design of such an AAA system. If, for instance, one realizes the privileges granted by a token generated by a blockchain program (e.g., smart contract on Ethereum (ETH)), which is then used in the authentication procedure to generate another access token or further refresh tokens, this demands numerous and frequent interactions with the blockchain. This temporary access token would be deleted after successful authentication. A token called Refresh will be used to keep logging in automatically. The token is being deleted to avoid replay attacks. All token interactions described here would lead to a transaction with the blockchain. If the blockchain can't handle scaling well now, this design would worsen it. Beyond scaling, the transparency problem is a significant issue that needs to be addressed. All transactions are publicly accessible, except for the participants in the blockchain (for private blockchains). It is not enough to have simple encryption in this regard. More on this problem in the section 2.2.6 The Transparency Problem.

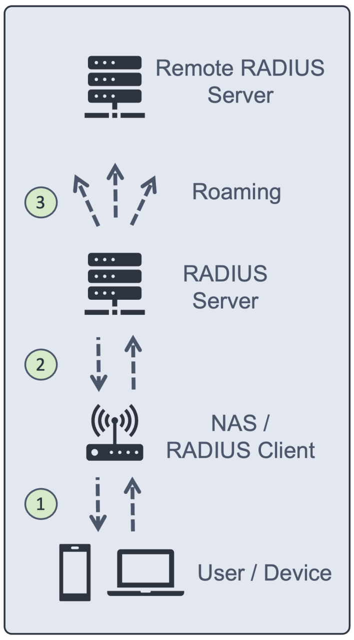
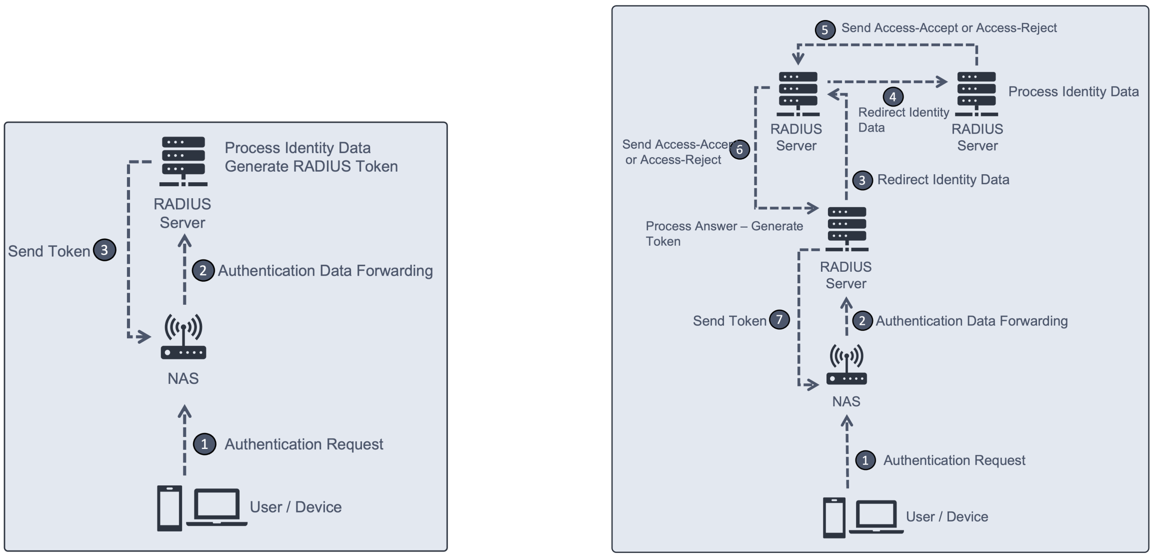


Figure 4.1: Interface points in the RADIUS scheme for blockchain integrations.

The desire for greater resilience to glitches and to an erratic course in blockchain technology may motivate a more cautious approach to blockchain integration. In this case, it would be best to move to the second option (b) mentioned for the RADIUS blockchain interface. This is where someone has an existing RADIUS system and would like to extend it with the decentralized functionalities of a blockchain. The RADIUS servers would still be able to optionally communicate with each other, for example, in proxy mode. Furthermore, a connection to the blockchain will be established, enabling the RADIUS server to communicate with it. A request for authentication could be made on the blockchain or in the traditional system. However, this introduces a new problem with diverging authentication or authorization results. One would have to make a personal determination in the specific application case. The actual additional costs of such a dual system are difficult to quantify, as they depend heavily on the architecture of the blockchain, the implementation of the integration, and the necessary decentralization. This hybrid variant has the potential to exhibit varying degrees of authentication on blockchains. For example, authentication will be extended to blockchain, but authorization and accounting not. Other mixed processes are also possible.

In the last proposal (c), blockchain technology is only used to communicate between the respective RADIUS servers. This design could be motivated by non-hierarchical authentication chains. A hierarchical system is used to route authentications in RADIUS systems like eduroam. Blockchain has the potential to establish a system similar to Web of Trust (WoT) for authentication. These messages might be about the authentication requests mentioned before. Furthermore, digital identities, for instance, in the form of certificates, could be transmitted as well. Transparency and scaling are still issues with this implementation. Every interaction would trigger a transaction, thereby enlarging the blockchain. Furthermore, all information would have to be encrypted in such a secret manner that any subsequent decryption remains unlikely. Current methods for achieving this can be found in chapter 2.1 Cryptography.

It is possible to implement the described interfaces in different ways. Figure 4.1 represents three communication paths. Each of the communication paths provides different capabilities for RADIUS interfaces a, b and c. The initial communication is established between the user device with a NAS or RADIUS client.



(a) RADIUS Local Setup (b) RADIUS Roaming Setup

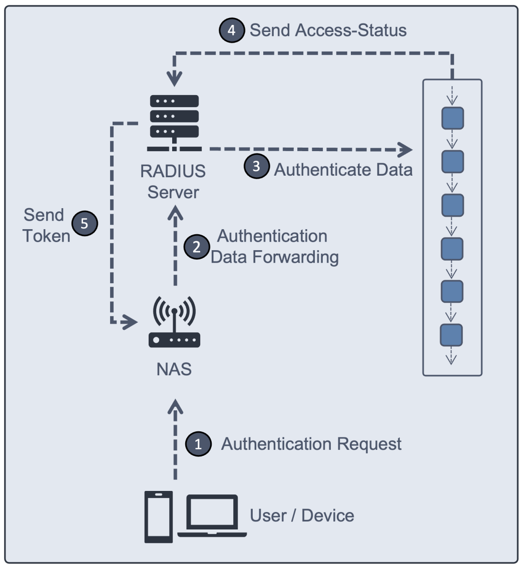
Figure 4.2: Authentication Schemes with pure RADIUS protocol

This communication only provides space for a full RADIUS implementation on the blockchain. Authentication would be requested to the blockchain, and if positive, the result would be passed to the RADIUS client. This could be done, for example, through a token, as is done with identity providers such as Trusted Third Party (TTP). The identity providers tasks would be taken over by the blockchain.

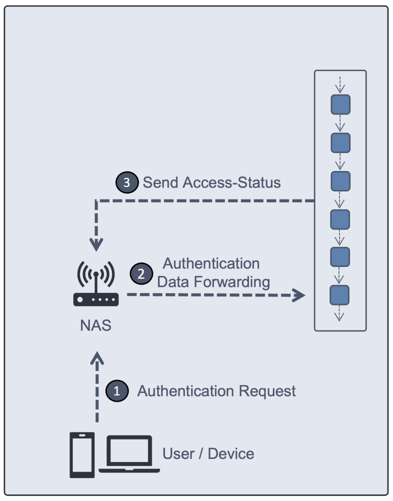
The second communication path is between the RADIUS client and the RADIUS server. The server will typically forward the authentication request and execute it. The response is sent back to the user via the RADIUS client. This communication provides scope for both a complete and hybrid interface implementation a and b. The NAS would then forward the request to the blockchain in case (a). The difference between this situation and the one before is that the NAS sends the request to the blockchain instead of the end user. In a hybrid implementation, a check is made to see if identification data is available locally and, if it's not, it's sent to the blockchain or another server (proxy mode). The hybrid approach is the most flexible from a technical perspective and is therefore preferred later for the development of the framework.

The third communication is between RADIUS servers of different organizations. This setup forwards a request to produce an Authentication, Authorization and Accounting (AAA) in foreign networks. A pure blockchain implementation is no longer possible, since a check has already taken place on the local server. Otherwise, the necessity for establishing a connection between the RADIUS client and the server would not arise. A hybrid implementation, as well as a connection among the different servers as described in options b and c, can be integrated at this point.

Each of the interfaces presented has its own advantages and limitations. It is becoming apparent that a universal solution is not feasible. To make an informed decision, the use case requires a careful analysis of the circumstances.If a RADIUS system exists already, a hybrid variant will be preferred. This would be the more secure and conservative variant. Because if there are any issues with the blockchain implementation, the old functioning system will continue to exist so that there are no failures. Other considerations to consider when designing an architecture include the blockchain technology with its consensus process. This determines the scalability of the system. This reveals the possible steps that could be transferred to the blockchain.



(a) A RADIUS Blockchan Hybrid System



(b) A just with Blockchain RADIUS System

Figure 4.3: Authentication Schemes by AAA-me with Blockchain

4.2 Schemes for Authentication

This section explains the authentication schemes performed by the respective roles in the protocol. A restriction is made to make it easier to understand. To keep the communication steps from being too confusing, not all of them are listed separately. Particularly when it comes to authentication methods that employ a challenge-response approach, only specific generalization steps are included. This greatly increases the overview if repetitive interchanging requests do not have to be mapped.
Later, we will discuss the complexity of the respective blockchain transactions. Transactions that are more complicated allow for more complicated programming, which is called smart contract capability in the literature. Depending on the complexity, the corresponding authentication methods described in 3.1 Authentication Methods are specified. Transaction-based actions have a limited scope of applications. However, the increased complexity introduced by smart contracts can lead to vulnerabilities.
If one intends to enable authentication through transactions, it is more challenging to implement the process to revoking a certificate. This is more feasible when utilizing programmable blockchains. Especially when multiple approaches are planned in the future, programmable blockchains offer greater adaptability and accelerate implementation. But such blockchains are also much pricier and have a higher chance of being hacked.
In the figure 4.2 Authentication Schemes with pure RADIUS protocol (a) you can see a scheme for authentication in the conventional RADIUS protocol. Here, an authentication request comes from the user via a device (cell phone, laptop, etc.) to a NAS. This forwards the request to the configured RADIUS server. The server then performs a check of the delivered data in the local storage. The positive scenario involves the creation of a temporary RADIUS token, and the user is notified of their acceptance of access. The connection can now be maintained using the RADIUS tokens. The shown illustration shows a local authentication without roaming. Here, only a check by locally available data systems takes place.
A strength of the RADIUS protocol is the possibility of authentication by roaming in foreign networks. Users can authenticate themselves with their data from the local network in other networks. The scheme can be seen in the figure 4.2 Authentication Schemes with pure RADIUS protocol. A foreign network has the same beginning of an authentication request as a local network. From step three, the procedure changes,

as the information must be forwarded to the remote network containing the users' authentication data. This is demonstrated in steps three and four. The fact that a so-called realm has been defined makes such a forwarding recognizable for the Radius server. The structure of this realm is similar to an email address. Indexed with an "@" symbol and terminated with a top-level domain, so that the corresponding server can be quickly found. The server then processes the data and sends it back to the NAS that first got the request. These are steps 5, 6, and 7. To maintain a connection, the RADIUS server must first process the response and, if necessary, create a RADIUS token. This is used to keep the connection, just like in the local authentication method.

The authentication schemes for an implementation with blockchains using AAA-me is shown in 4.3 (a) and (b). The blockchain is shown here in its data structure, not as a group of nodes. First, a user logs in to the NAS with an authentication request, as described in the conventional RADIUS protocol. In a hybrid implementation of the protocol with blockchain features, the request from the NAS is now sent to the server. The server uses the blockchain methods available to it to verify the authentication data. The server is aware that it requires authentication via the blockchain, as the ID has been designated by the user as a blockchain-compatible realm.
The second case describes a full implementation of the RADIUS protocol on the blockchain (figure 4.3 (b)). The main difference between the conventional and hybrid implementations is that the NAS is already capable of transmitting the authentication request to the blockchain. There, it is processed and returned to the NAS.

4.3 Programming on Ledgers

Programming on blockchains and DLTs introduce novel and essential programming paradigms. The immutability of a record and its transparency force programmers to reconsider their approach to development. For instance, the immutability of software updates can lead to complications. If I have successfully installed and instantiated a contract, it is not feasible to simply substitute it with a new one. Bringing a new contract with the renewed functions onto the blockchain would be like creating a new instance, and all users would have to react to this change independently. Furthermore, all stored data and states would not be able to be automatically taken along. These would have to be transferred to the new contract manually, which would be expensive.
Depending on the consensus methods used or distributed data storage, new, unpredictable security issues may arise. This happened on the ETH blockchain in the "reentrency problem", which caused a hard fork. (See chapter 2.2.3 Ethereum and Blockchains 2.0). A case that was not considered from a classical security perspective led to the transfer of Ether to a foreign account over several transactions. This error required a hard fork, which resulted in the split of parts of the community. Ethereum Classic has been around since then.
The poor scalability of blockchain systems leads to necessary performance considerations in the design and implementation of practices. And that before the first install in development. The aforementioned immutability complicates the issue. The complexity of performing updates ensures a need for best performance from the beginning.
Consensus is an important factor to consider in development, not only when considering security considerations. The intervals within the blockchain can vary, or the prediction of a blocks' generation can be erratic, depending on the consensus. This has a direct impact on a contract or the execution of transactions, through so-called race conditions.[1].

[1]Race conditions are events in a system that can lead to inconsistencies, malfunctions, or even security vulnerabilities in programming. A common example is when many threads or processes access a memory area and affect each other.

Design patterns have been established to address the issues mentioned above. Data segregation is one of the recurring patterns. In this pattern, the data is separated from the business logic. This means that there are two smart contracts in use instead of one. If a method with access to the data is executed, it will access the data smart contract known to it. It is likely that the user will only notice this access by increasing transaction costs. With an update, it is possible that the data does not have to be expensively ported onto a new smart contract. In this particular instance, it is sufficient to establish a connection between the two smart contracts. This variant still has a problem. Even if I modify my approach, the user still needs to be informed about the new instance. And therefore have to adapt the corresponding tools by itself. This scenario is extremely error-prone. The person might not have seen the announcement, or they need to ask for a copy of the acknowledgement. This further complicates the updating procedure. To address the issue at hand, in addition to segregating the data in a distinct contract, a proxy contract is proposed that also segregates the business logic. This proxy contract manages the addresses of the data and business logic contracts. The user controls the proxy with his methods and is forwarded the methods and data according to the addresses. This design pattern is shown in the figure 4.4 Representation of Data Segregation. Easy to see are the three stated smart contracts. An implementation of this concept can be seen in 5.1.3 Blockchain Programms for AAA-me.

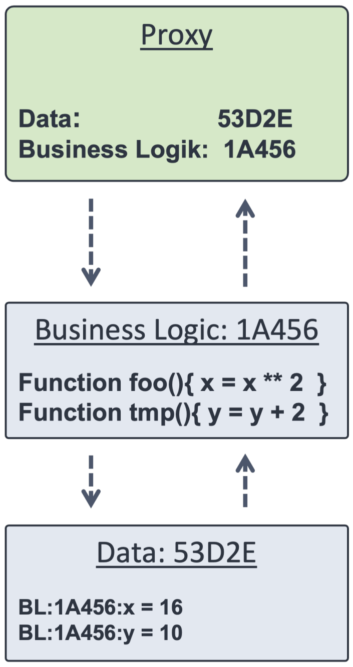


Figure 4.4: Representation of Data Segregation

The advantages of a system modified by data segregation are easily identified. It comes to the so-called Separation of Concerns (SoC), where the different aspects are organized in different contracts. The SoC is known from software development and aims to divide different aspects into modules. This leads to improved readability, improved maintainability, and increased scalability for programs designed in this way. Partitioning can be beneficial for large-scale smart contracts in that it positively affects cost and performance.
Nevertheless, there are also drawbacks. The dependence on a working proxy contract is evident. If the corresponding cryptographic secret key is compromised or the contract has a vulnerability in the code, the associated contracts are at risk. An attacker could, for example, link the business logic through a new contract and thus change the functions as they see fit. Now, not only one smart contract but at least three contracts have to be organized, checked, and maintained, which increases the management complexity significantly. This also includes the access rights, since the methods and data can't be shared with everyone, but they can be shared across multiple contracts. This requires appropriate lifecycle management.
In conclusion, this paradigm can be applied to all smart contracts and all blockchain technologies. Particularly necessary for those with high security requirements. It's possible to think about coordinating multiple projects through a proxy contract. Nonetheless, the power and performance tests in the implementation are not significantly affected by this.

4.3.1 Erasing Data on Blockchains

Blockchain technology is known for its immutability of data. This characteristic is caused by the chaining of hash pointers. A data block's hash value is formed and used as input for the next data block. If I want to alter or even remove a historic date, all the hash values associated with it change. With all the blocks that follow. Thus, a modification of the data is possible, but it requires a fresh calculation on all nodes.
The first requirement for deleting data is to increase privacy. Blockchain technology is not anonymous, but pseudonymous. It is therefore possible to identify the identities behind the respective addresses of the wallets

with sufficient effort. So, it doesn't apply to privacy laws. Furthermore, in programmable blockchains, it is possible that Personally Identifiable Informations (PIIs) may have been stored in the blockchain programs. It is also possible with blockchains that are not programmable, but the need for data for specific programs makes it more likely on programmable blockchains. It's easy to understand that you need to delete this information later. If the encryption process has been classified as broken, it may still be necessary to delete the PIIs.

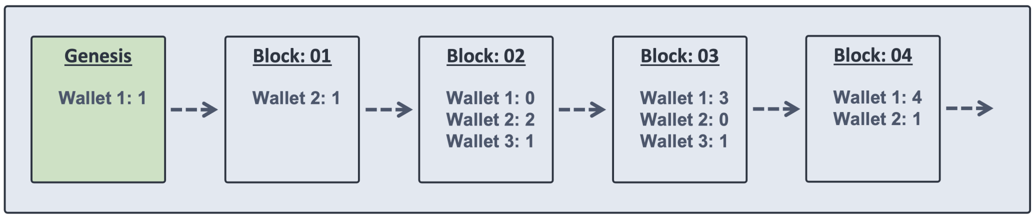
Data protection may initially be a desired objective for an individual with high self-imposed protection objectives. However, it could also be required by legal requirements, such as the European General Data Protection Regulation. For instance, it is possible that information may be legally compelled to be deleted within a certain timeframe, a task that is rendered more challenging on an immutable blockchain. A transaction may also be in violation of applicable law. For example, quite early in the Bitcoin (BTC) blockchain, websites were linked that referenced child pornography material. It's an extreme case that has been regulated the same, or at least similar, in many countries internationally. But a blockchain that works internationally, like BTC or ETH, has the problem of different regulations. Depending on the country, a transaction may be legal or illegal. The deletion requirement for this particular transaction would also only be incurred in one country.

Additionally, a user may need to delete information for other security reasons. Particularly if the user believes that this could compromise the security of the transactions he or she has initiated if the information is not erased. If data is used not only for the transfer of values, the level of security will have an impact on the realization implemented outside. These must be protected, especially through connections between information on the blockchain.

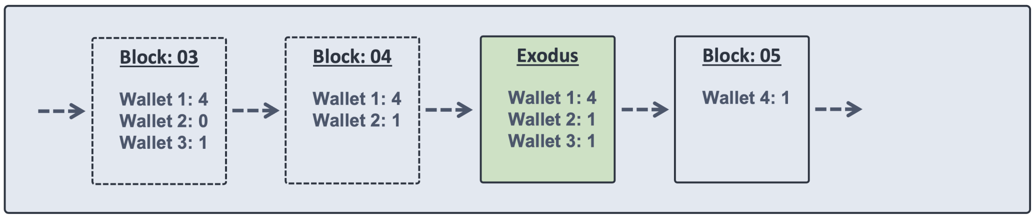
If blockchains are stored permanently with transactions, the demand for storage space can grow significantly. According to [30], a current node needs 491.5 GB. On its own established BTC Node, 601.3 GB was reported. On the Ethereum blockchain, however, it is even larger, at 962.67 GB in full synchronization [90]. The varying specifications of the memory consumption for BTC result from the options available to the node operator to configure the node. Besides a complete synchronization, you can also delete old or no longer needed transactions. This case keeps the block header with its concatenation and the root hash of all transactions. One can remove the transactions themselves, as well as the intermediate levels of the Merkle tree. This can lower the amount of memory needed. The idea is that not every transaction today needs to be tracked for five years. Only the chaining correctness must be verified. To set up a pruned node, it is still necessary to connect to a full or archive node. The transactions are loaded, checked, and verified to ensure that they are in compliance with the current blockchain state. Once this number is reached, the pruning node deletes all older or no longer used transactions. From this point on, there is no difference between the Full and the Pruned nodes. This method has the advantage of reducing the memory requirement of a block to a few KB. However, to check the blockchain state, it is still necessary to connect to full nodes. This means that old information is still saved in the system.

The problem of permanent storage of PIIs is often encountered in projects for the realization of digital identities with blockchain and Distributed Ledger- (DL-) technologies. Only a verified hash value of this data is stored on the ledger. The data itself resides on an external, accessible, and secured data source. In addition, the hash value on the ledger contains a link to this data source. This procedure is reminiscent of the implementation of Non-Fungible Tokens (NFTs) on the ETH blockchain. Data can be deleted, and only the link with an integrity-checking hash value remains on the ledger. However, what remains is the transaction itself. This means that the keys are still in the owners' control, and they can still be gathered even if the link to the PIIs has been removed. Thus, the storage requirements of the distributed ledger cannot be lowered. Despite greater use, larger amounts of data can be preserved on the blockchain, and thus the need can remain limited. Since large amounts of data remain excluded.

The Oracle problem is often discussed in this implementation. External information is provided through this particular interface, which represents a strong dependency. This raises the risk of a Single Point of Failure



(a) The creation of a blockchain based on the Genesis Block.



(b) Exodus Block integration to continue the blockchain.

Figure 4.5: Realization of the Exodus blocks. - These figures illustrate exemplary and simplified block creation mechanisms and how an Exodus block can be integrated as a checkpoint. In this example, transaction fees are not considered, but one token is created per block. If a wallet's status changes, it is added to the current block. The structure of a blockchain chain can be seen in figure (a), beginning with the initial block called Genesis. Figure (b) shows that an Exodus block is created that contains all wallet states greater than 0 from the past.

(SPOF) and calls into question the use of a DLT.
The linking of external data could have been realized differently. Either they are present in the data input of a transaction as a link. If the DLT supports it, blockchain programs (smart contracts) are created that can access external data. The blockchain programs could be used as Oracle contracts to connect external data sources to the blockchain. Data added to the blockchain in this way cannot be removed.
The implementation of sharding is currently in progress in the development of ETH. The current chained blockchain will be realized by several interlocking blockchains. Individual blockchains are referred to as shards and are interconnected by a so-called beacon chain. Furthermore, it is imperative that transactions across the shards be made feasible. This approach ensures that each node does not have to synchronize all the network data but only the necessary ones. The memory requirements are reduced by this. However, the deletion of information is not achieved in this way. This could become much harder because the chaining of shards could make transactions last longer to keep the overall state safe.

Another way to reduce the total storage requirement is by setting up a new blockchain with the current state. It is therefore possible to delete old transactions. To achieve this, a new genesis block would be created by premining. Premining can be understood as pre-configuring the distribution of coins or tokens. It will be used to transfer the old state to the new blockchain. The beginning of a blockchain is referred to as a genesis block. In it, parameters regarding the blockchain are defined in advance, including potential premining. Premining is understood as a premature allocation of states. Numerous token projects have already allocated credits for their tokens to the addresses of the developers before the start of the first block. This is intended to be a development and startup aid for the projects and is highly debated. One could use this preallocation to store all the current active blockchain states in a new Genesis block and then restart from there. The block could be referred to as an Exodus block, following the Bible's example of the Genesis block. This procedure is comparable to checkpoints. This checkpoint stores the current state of variables. The transactions that led to these states and were created and verified before the checkpoint can be subsequently deleted.
The described procedure can be understood from the figure 4.5. The structure of a blockchain is shown

starting from a genesis block in figure 4.5 (a). Up to block five, the state changes are shown in a simplified way. Figure 4.5 (b) shows the integration of an exodus block into the blockchain. The blocks preceding it are now dashed, which indicates a possible removal. Since the Exodus block, older transactions are no longer needed to check the ledgers' integrity.
This method is promising when it comes to avoiding constantly growing storage needs. Nonetheless, it is not beneficial if one intends to modify or eliminate data. Furthermore, it is difficult to implement and can lead to problems. Although forks are a common occurrence in the blockchain and can be effectively managed through the established consensus protocols, the significance of an exodus block to persistence is significantly greater, rendering it a potential target for potential attacks. This requires further investigation so that any implementation doesn't pose a hazard to the entire system.
In conclusion, there is no single solution to the deletion requirements of an immutable blockchain. Many approaches attempt to alleviate or mitigate the issue. Implementing the various suggested solutions will be the most effective approach to minimizing the issue.

4.3.2 Blockchain Connection

The perspective of a software developer and system administrator reveals a variety of prerequisites and methods for establishing a connection to a distributed ledger. The developer needs to be able to find a simulated blockchain on the computer, so there aren't any costs for testing. It is imperative for the administrator to establish a connection with a functioning blockchain instance. But there are some security issues that need to be thought about. A single point of failure is still possible if only a single node is set as access. Furthermore, the effort increases if there are many nodes set as access.
One question that arises is whether to set up and manage multiple blockchain nodes yourself, or to connect to several different nodes. The second scenario, however, involves additional considerations by the administrator:

1. Data consistency (Consensus)
2. Latency and timeout management
3. Load balancing
4. Security requirements (authenticity and integrity)
5. Error treatment (Consensus)
6. Scalability

It is possible that the nodes of a blockchain have different views of the data. Particularly during the propagation of blocks, there may be discrepancies in the data received from the various nodes. Verifying the information and resolving any discrepancies is crucial. To accomplish this, existing consensus protocols can be utilized to ensure uniformity.
Connecting to multiple nodes can result in increased latency. This potential latency is increased if they are distributed across different locations. To ensure seamless interaction with the blockchain, timeouts, and incompatibilities must be identified and addressed promptly. Asynchronous methods can be used to run processes at the same time without waiting for them to finish.
When working with multiple nodes, it's important to distribute the load evenly among the various connections. Use load-balancing algorithms or set priorities for nodes to achieve this. A round-robin approach could also be employed, in which nodes are surveyed one by one randomly. To maintain security and trust in the data in the process, for example, so-called Proof-of-Authority (PoA) procedures exist that have also been implemented on round-robin systems. The consensus is created by examining the information one at a time

and assigning writing permissions randomly. More about this is in chapter 2.2.2 Consensus Procedures. Should a node encounter security issues arising from connection issues or an attempted attack, it is possible to detect and respond to them retrospectively. Furthermore, the erroneous node should be penalized and, if necessary, removed from the randomized grouping.
The choice between running your node or connecting to a service provider has little impact on the actual operation of AAA-me. The decision must be made separately, and only the connection information must be entered in the configuration. This implies that this decision does not require further consideration within the program. However, it's strongly suggested that you establish your node. This approach has the potential to enhance the security level of the blockchain and the verification of the data's authenticity.

4.4 Blockchain Transactions and Programs for Triple-A functions

A high-level illustration of how to implement AAA functions on a blockchain is given in this chapter. External factors, like the oracle, scaling, or transparency problem (section 2.2.6 Criticism of Blockchain Systems) are not considered at first. These are listed at the end.
An AAA implementation on the blockchain can be categorized by the implementation with transactions or blockchain programs. Moreover, the issue with the name from the introduction should be mentioned regarding blockchain programs, which are usually called smart contracts in the literature. The term blockchain program is employed in this section. It is essential to consider that a blockchain program will deliver significantly more functional scope, which will entail significantly higher costs. The cost of blockchain technology is impacted by the issue of having calculations performed simultaneously on all nodes. These are typically quantified in transaction fees, to which the calculations have been mapped. However, data from a program can be retrieved faster on the blockchain, especially when there is a lot of data on an entity. This could lead to significant speed increases. Nonetheless, by utilizing the data inputs from transactions, one could enhance the reach of blockchains and DLTs, thereby eliminating the necessity for customization. This means that you can use the Bitcoin (BTC) blockchain for your applications both in a private place and in a public place.
Different authentication methods described in the section 3.1 Authentication Methods are used to further categorize the different methods. The techniques described in this section are based on those employed by freeRADIUS. Depending on how the DLT is used, other authentication techniques can be implemented, and the outcome can be passed on to the RADIUS server. Whether a transaction-based or program-based implementation is aimed for depends on the method chosen. Therefore, this decision also has a direct impact on the determination of which DLT should be employed.
Until now, only the authentication of a user has been considered. For further discussion of the cases of authorization and accounting, please refer to the section 4.5 Accounting and Autorisation. Since authorization usually means only one additional piece of information in the data set, it is not a significant limitation for the general description. In accounting, there are many other things to think about that can affect the choice of DLT.

4.4.1 Transaction-Based Implementation

If you want to use blockchain technology for transaction-based systems, it's easier to choose one that has been designed specifically for these systems. According to chapter 2.2.1 Bitcoin Blockchain, BTC is such a technology. Therefore, the explanation will focus mainly on it. An implementation of alternative technologies, such as ETH or Hyperledger Indy, is undoubtedly straightforward and therefore will not be further discussed.
Normally, transactions don't have a lot of space for data input, so you have to reduce the needed storage space. Consequently, important information should be stored in a compressed form. One potential consideration could be the implementation of a hash value for all the information provided by a user seeking to

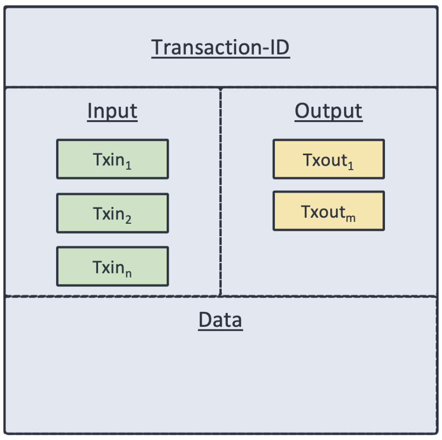


Figure 4.6: Representation of a transaction in transaction-based blockchain.

authenticate into a network. In addition, you can use the security features in the wallet and node software of the blockchain technology. For example, BTC, ETH, and other similar technologies already have ways to make cryptographic key pairs, manage them, and create signatures. Challenge-response procedures for the verification of certificates can be easily implemented. Another consideration is the use of compression algorithms, which we already use. The data required for authentication would therefore be reduced to the necessary size. However, there is no cryptographic protection here, as in the first consideration. This option is not recommended and will not be implemented for AAA-me.

Knowing how a transaction works helps you envision how a potential certificate might be put into action. A detailed description of transactions can be found in the section 2.2.1 Transactions. However, other DLTs are similar in structure. In the figure 4.6, one can schematically see the structure of a transaction-based blockchain transaction. In a conventional transaction for the transfer of values, the sum of the inputs would be greater than or equal to the value that is being transferred. The right side of the figure lists the public wallet addresses that should get the desired state changes. The left side provides inputs to state that you are allowed to transfer. Every "Txin and Txout" is a transaction ID, so they can be seen like the transaction in the figure. If you are using a certificate for authentication, this could be implemented by using a transaction input to interpret a confirmation from an issuer. The addresses of the people or entities who receive a certificate can be found on the right side of the transaction. Such an implementation would result in a certificate being issued. Furthermore, it is possible to create multiple certificates for multiple users at the same time in one transaction. This could be used to authenticate and authorize access to several work devices for one employee. If a user who has been certified in this way wants to connect to a network, the information is sent to the Network Access Server (NAS). This checks the legitimacy of the person who initiated the transaction. It also checks whether the user is actually in possession of the address mentioned in the output. A challenge-response procedure, or a Zero-Knowledge Proof (ZKP) can be used to accomplish this check.

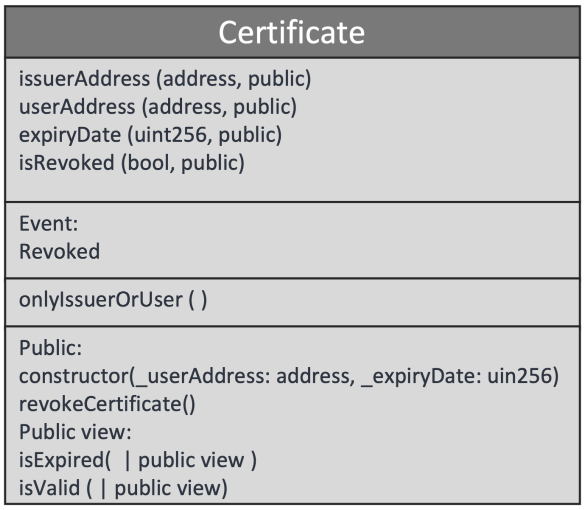
The data field on a certificate can have more information needed to follow the X.509 standard or system rules. Furthermore, additional cryptographic elements could be included here to implement further procedures in addition to the challenge-response procedure already described. This realization lacks the simple possibility of revocation by the certificate creator, despite the many possibilities. The person who owns the certificate, on the other hand, has the power to revoke it via a blockchain-based transaction. Additionally, a time limit can be established to ensure that a certificate is automatically revoked after a specified time frame specified by the user or issuer. However, this cannot replace the issuer's decision to cancel.

4.4.2 Program-Based Implementation

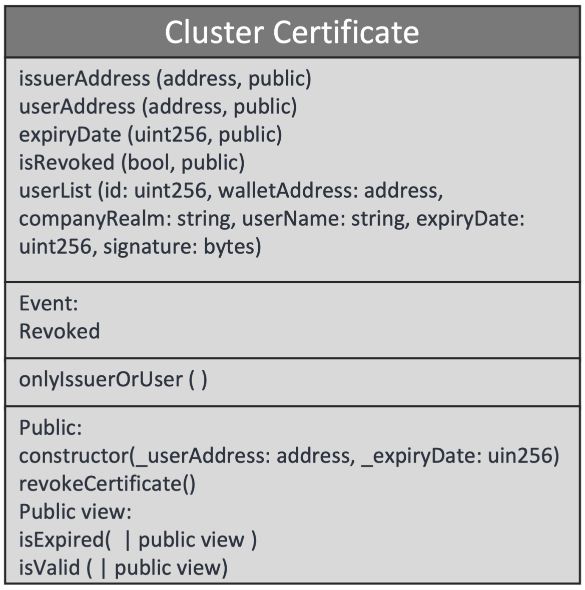
The absence of a suitable mechanism for revocation of the certificate issuer in the transaction-based implementation makes the creation of a certificate for program-based authentications extremely intriguing. Even though the procedure is similar. The complexity of programming on a transaction-based blockchain is significantly higher than on programmable blockchains. On programmable blockchains, transaction-based systems cannot be used. Therefore, an account-based system is needed. Essentially, the difference is in the structure of a transaction and its processing. More on this in section 2.2.1 Transactions.
To remind you, a blockchain application is installed by transferring the byte code (or other representation understood by the blockchain technology) as a transaction. These transactions are similar to the one presented in the figure 2.2.1. The program is stored in the data field.
Few limitations are given in the conception and implementation of AAA functionalities due to Turing-complete programmability. It's important to keep the idea of the transparency problem in mind, so that no PIIs is saved. Moreover, high complexity can increase transaction fees a lot. Nonetheless, if the possibility of high fees is not a concern due to the utilization of a private blockchain instance or the governance model allowing for it, all authentication methods can be implemented.

At ETH one could see a development in programming templates. These are called token standards. These are specifically designed for the most diverse requirements. The most famous template is the so-called Ethereum Request for Comments (ERC) 20 standard. It is possible to create an own token based on ETH. However, this standard is only of limited use for AAA tasks. To implement accounting, a token by this standard could be implemented. But the timestamping of the Ethereum blockchain already makes it possible to prove any settlement, so the added value can be questionable. The presented token standard is also known as a fungible token in the literature. These are tokens where each unit is equivalent and interchangeable. Accordingly, it is irrelevant to the user which five tokens he has. The primary factor is that he possesses five tokens.
Besides the fungible tokens, there are the "Non-Fungible Token (NFT)". There is an identity assignment to the token, which makes them unique. The intention was to create a verifiable scarcity in the digital world. This meant that a digital receipt on the blockchain could be made for a physical object, which would show who owned it. The hype about the ETH blockchain with NFTs has shown that it's difficult to set up a system like this. [219] However, there is potential for digital certificates to be included in this standard, since they must also be distinguishable. But a NFT can be passed on, something that is not desired with certificates. Therefore, the so-called "Souldbound tokens" could become exciting. These correspond to a NFT, but cannot be passed on.
Turing completeness brings responsibility and complexity with it. This renders it more challenging for the developer to create a secure program that can be converted into a certificate. Even if token standards already exist, there is still a risk, especially if adaptations to these standards become necessary.

In the figure 4.7 , you can see two class diagrams for how a certificate is made and how a bundle of certificates is made.These two class diagrams illustrate the implementation that will be discussed later in 5 Implementation, more precisely in chapter 5.1.3 Blockchain Programms for AAA-me.
The diagram (a) illustrates a simple certificate. The necessary variables include the current state of the certificate, as well as the issuer and the user. Both of these fields are filled in with the public wallet address, not a name. The address of the issuer is obtained from the transaction that generates the smart contract, and it is imperative that the issuer performs this task. The trust in the certificate comes from him, and accordingly the authorization for the user. Two additional events have been implemented that can be used for an external reaction. Thus, a node that interacts with the associated NAS in communication with the user can send necessary information directly to the NAS or to the user himself when an event is detected.
A so-called modifier method is described in the listed method onlyIssuerOrUser. This modifier can be used


Certificate
issuerAddress (address, public)
userAddress (address, public)
expiryDate (uint256, public)
isRevoked (bool, public)
Event:
Revoked
onlyIssuerOrUser ( )
Public:
constructor(_userAddress: address, _expiryDate: uin256)
revokeCertificate()
Public view:
isExpired(| public view)
isValid (| public view)

(a) Class Diagram of Smart Contract Representation of a Certificate.


Cluster Certificate
issuerAddress (address, public)
userAddress (address, public)
expiryDate (uint256, public)
isRevoked (bool, public)
userList (id: uint256, walletAddress: address, companyRealm: string, userName: string, expiryDate: uint256, signature: bytes)
Event:
Revoked
onlyIssuerOrUser ( )
Public:
constructor(_userAddress: address, _expiryDate: uin256)
revokeCertificate()
Public view:
isExpired(| public view)
isValid (| public view)

(b) Class Diagram of Smart Contract Representation of a Cluster of Certificates.

Figure 4.7: Class Diagram of Certificat Representations

repeatedly in the smart contract and requires that only the issuer or owner of the smart contract be able to change its state.
Further methods include a getter for reading the states and a revocation function, which allows both the issuer and the user to revoke the certificate. Thus, the possibility, which is absent in the implementation based on transactions, is realized. It should be noted that the getter methods do not require a blockchain transaction and can therefore also be retrieved in a well-scalable manner. However, revocation requires a transaction, which in turn requires fees to be paid.
In the class diagram (b), there is a similar implementation as in the class diagram (a). The primary distinction pertains to a list. This list can help users and access points find permissions for many users more easily. Also, a new certificate for a user can be made more easily by using this method, since not a complete program must be uploaded, but only a part of it must be changed.

4.4.3 General Implementation Details

We observed that transactions, as well as blockchain programs, can be used to authenticate an entity. The extension of authorization can be accomplished by incorporating a variable set for the corresponding accesses. You can do this using either of the methods mentioned. If one desires to include an accounting, it is recommended to refer to the section 4.5 Accounting and Autorisation. There, the two situations of authorization and accounting are explained in detail.

Until now, the potential for implementing AAA functions on a blockchain has been extensively explored. The question of the added value of such an implementation, as well as the clarification of obvious issues with blockchains, is still unanswered. Part of this question is answered by the implementation, specifically about SPOF resistance, in chapter 5.2 Experiments. The general issues that a blockchain brings with it remain to be highlighted, specifically in relation to AAA. Those general issues include the scaling problem, transparency problem, and Oracle problem. All of these problems can be addressed in the same way, with reference to transaction-based or program-based systems.

- *Scalability:* The scaling problem is caused by the fact that few transactions per second can be validated. Depending on the design of the authentication methods and the frequency of access to the blockchain, both read and write, this can potentially exhaust and overload the limitations. Therefore, it is essential to establish a clear strategy for the transfer of operations to the blockchain and the selection of the appropriate blockchain platform for other projects. The decentrally managed blockchain typically does not encounter any difficulties when performing read operations. Any other node can be selected for reading if it's not reachable. Write processes alone are subject to scaling issues because they must be approved by the consensus. If the write procedures also have plenty of arithmetic operations, scaling problems also arise.

  The most appropriate measure would be to incorporate as few process steps as feasible onto the blockchain. This can be achieved through longer-running certificates or other digital identities. Additionally, when designing a blockchain infrastructure, it should be considered which operations take place on the ledger and which do not. If there are many calculations running on the ledger, the nodes must be supplied with sufficient hardware. Second-layer strategies are also important for outsourcing subprocesses and thereby boosting scalability.

- *Oracle:* Adding information from other sources to the blockchain is a difficult task that needs to be considered individually. In general, it is detrimental to the decentralization principle of the DLT if one relies on a provider to store the desired data in the blockchain, as it leads to the loss of an intended property of the act. Furthermore, one loses functionality and thus the need for Turing complete programmability if external data cannot be brought in.

  This leads us to conclude that digital data derived from bilateral transactions is more manageable on the blockchain. An requirement is that this information is required for subsequent processes. The additional costs due to distribution would not be justified. Examples of such virtual assets include monetary assets or certificates. Not all projects can be reduced to virtual information. Therefore, the necessity for oracles is existent. If one requires external data, it remains to be clarified where they could be stored. How can they be accessed? Is the management and access compliant with data protection regulations? What is the potential impact of the provider of this data on the operations occurring behind the blockchain in external networks?

  The last question can have big effects on a RADIUS system. By allowing roaming, an incorrect local registration can facilitate access to a plethora of networks. However, in addition to authentication, RADIUS systems also allow authorization and accounting. The consequences would consequently be fatal if a defect were to infiltrate the advancement ofAAA functions on the blockchain.

  To avoid these issues, it is recommended that interactions be limited to virtual information. This is where there is the most potential. Generally, certificates can be used to make external data sources obsolete. To boost data security and prevent possible manipulation, consensus procedures and multiple data sources should be established and utilized.

- *Transparency:* Any operation carried out within the blockchain can be viewed by any individual. Private blockchains are limited by the group that has access to them. The fact that even encrypted data could be tapped today and decrypted later makes this a threat. In the literature, this issue is also referred to as Harvest Now, Decrypt Later (HNDL). Therefore, it is crucial to preserve information free from identifying characteristics. Furthermore, mechanisms must be put in place to avoid the correlation of data. Otherwise, experts might be capable of tying individuals to the digital keys, thereby interpreting subsequent interactions.

  Measures against correlation analysis can be taken from current open blockchain projects such as BTC or ETH. Applications are already being tested to further obscure identities. One such measure is the use of constantly new wallets to keep affiliations as minimal as possible. For AAA tasks, new keys can be made for each process. If one has received a certificate with permissions, one could create a new wallet and give it the same permissions as the address in the certificate.

Furthermore, technologies such as hash values or ZKP should be used to further obscure the information on the ledger. Proof of the information would then be provided and verified through bilateral interactions.

Additional functionalities can be easily incorporated into a blockchain system for AAA. Particularly when it comes to ensuring compliance with a standard, such as X.509 for certifications issued by a Public Key Infrastructure (PKI). If you need another data field, this is included in the template and marked as a requirement if needed. The only valid transactions for creating a certificate would then be those that contain the newly implemented data fields. Possible data extensions to a certificate could include:

- Network information: The information about the network configuration could be integrated. This could be used to identify hardware, obtain authorization, or establish a trust network in a PKI context.

- Personal Information: Information about an identity could be added. This information may help with additional restitution or authorization processes. An illustration could be that a user must be of legal age to access a network. Because of the already discussed transparency issue, this information must be handled securely.

- Client Information: The information pertaining to the user's device can be beneficial, in a manner similar to that of the network. It is necessary to handle these data with caution because they may be of PII.

To summarize, despite the existence of numerous implementation alternatives for AAA functionalities on the blockchain, the utilization of a certificate coupled with the concealment of PII through hash functions or even ZKP holds the greatest potential to effectively fulfill the desired properties securely.

4.5 Accounting and Autorisation

In the previous sections, we examined the authentication implementation on a blockchain. However, within the AAA systems, there exists a total of three significant tasks. Furthermore, there are also aspects of authorization and accounting. The procedure for adding this functionality seems complex, but it can be treated the same way for all blockchain approaches described. Hence, these two functions are regarded in a distinct manner.
An authorization defines the permissions of an authenticated entity within a system. Thus, authorization can be given to either physical or digital environments. But also task areas and permissions for the transfer of rights can be set up. With reference to the example application of an eduroam, there are usually no additional permissions. Either one is authorized to access the network facilities or not. However, in certain situations it may make sense to have guests in isolated network areas, or to define other restrictions and areas.
To achieve authorization, a group assignment is stored in the RADIUS system. This assignment is saved locally and requires the dialing party to be authenticated before it can be saved. Therefore, authorizations must be set up and agreed on in advance. In the RADIUS use case, eduroam, students dial into the Internet infrastructures of other participating universities and are allowed to access the Internet. Therefore, authorization has been granted in advance that this access is available to any authenticated user.

To implement authorization on the blockchain, nothing needs to be changed in the described protocols, whether implemented via the RADIUS protocol or by AAA-me. The important changes happen on the network in predefined ways that automatically register things. Through AAA-me, only the optional blockchain-level integration is feasible. Further information is added to the program on the blockchain, for example as a variable in a list of identities. This information is used to determine the respective and individual authorizations. For illustration, one can look at the class diagram from the figure 4.7 Class Diagram of Certificat Representations (b) on page 70. You can see a list of things like ID, walletAddress, companyRealm, userName, expiryDate, and signature. The associated authorization could be mapped with the variable "autorisationGroup: int". Regarding the transparency issue, this could be stored as ZKP and thus comply with security standards.

As external work steps are necessary here, authorization can be easily integrated into existing RADIUS systems. The process of executing the accounting is, however, extensive and incredibly complex. Especially if the transparency problem is included.
Accounting is another integral part of a AAA protocol. freeRADIUS already provides the implementation of a billing as an implementation of AAA protocols. Especially when it comes to roaming services, this is a challenge due to dependencies and trust in the correct measurements of other providers. Moreover, there are security concerns because of SPOF considerations. So, it might be a good idea to use a blockchain for some or all of the billing, as we will see later in this document.

Before examining the various realizations of settlement on a blockchain, a schematic explanation of settlement in the RADIUS protocol is provided. For this purpose, two processes are important: the login and logout of a user to a RADIUS server, and the transfer of this information for an overall settlement.
In order to use its services, the user must first authenticate himself to the network. If the usage is billed, a billing procedure is initiated in addition to the authentication, which can take up to three steps. First, a *start record* is created by the user submitting a message to the RADIUS server. The contents of this message include details about the individual's name, their Internet protocol address, their unique session ID, and the time the connection was established. This message could be signed to prevent later manipulation. The process could also include a request for interim records, which would permit prompt identification of a connection's termination. In this particular scenario, messages are periodically exchanged between the user and the server. If the connection is disrupted, the billing process may be halted. The contents of these messages include the most recent details about the current session, such as the amount of data gathered so far and the time remaining. If the connection doesn't end suddenly, the user can end it with a *Stop Record*. The message required for this will contain the final details of the session, including the total duration and the total amount of data used.
The procedure described occurs between users and a network. To use roaming, an agreement must have been reached between the various network operators, in order to coordinate the hardware and define the billing management of the services, especially if spontaneous abortions could occur. The process, as previously described, proceeds. The user has ended the session and transferred all information to the RADIUS server. This now creates a *Call Detail Record (CDR)* with all the information transferred by the user and its own measured information.
These CDRs are sent from the server to the users' home network or to a clearing house. Depending on the agreement, the connection can then be billed for the connection.

We should now look into the possibility of settling on the blockchain. The cases of a hybrid or entire implementation of the AAA protocol on and with a blockchain are very similar. It is possible to start an authentication and authorization process either with just a NAS and a blockchain behind it, or via a NAS with an AC RADIUS server. Subsequently, the procedures and considerations remain unchanged.

The user now writes a message for their start record. The blockchain also allows for simultaneous transmission of these messages, either from the user, NAS, or the RADIUS server. In the same way, the Interim and Stop Records are made. Finally, a CDR is created, and it's obligatory to post it to the blockchain. This data suggests that it is no longer possible to erroneously revoke accounting information.
One noteworthy attribute is the capability to integrate events with blockchain programs (smart contracts). An event can be comprehended as a console output. One can define a method in the program such that an event is triggered immediately after an authentication has been successfully performed. This event can be used as a start record for building a settlement. If a logout function is also defined with an event in the program, then this can also be used as an event-controlled stop record. This allows for remote measurement and the generation of a CDR for billing.

Regardless of how the implementation is decided. With a blockchain, the problems of transparency, scaling, and Oracle still exist. Particularly when it comes to the scaling issue, one must consider whether it makes sense to make all record transactions blockchain-compatible if there is a substantial ongoing workload, or if a CDR is not sufficient. If roaming has been used, the Oracle issue is less important because accounting data is created in multiple places. and is important for accounting multiple networks. Accounting procedures don't require any external data. In terms of openness, it's possible to say that the relevant data should always be obscured, and not just encrypted. Here ZKPs and cryptographic hash functions may help.

4.6 Governance in Blockchains

Throughout the chapter, a great deal has been shown on the realization of AAA functionalities in various forms on a blockchain. Also, important for an implementation of AAA-me are arrangements, processes, and agreements that enable several participants to interact with each other in a trusting manner and to respond to events on the blockchain. This refers to standards that define how to interact with others. Actual interaction requires decentralization and, therefore, a blockchain. By extending the RADIUS functionalities, which have already implemented such agreements in applications such as eduroam, additional essential agreements have been incorporated. Blockchain management, implementation, and execution are some examples. This section talks about these agreements, which can be thought of as "Governance".
Below is an overview of the necessary steps for agreements, which are necessary for a functioning roaming system with the RADIUS protocol. Should the technology of the components required for the protocol change, further steps may be required.

Configuration: When establishing the individual components, it is important that security features, such as shared secrets, are also exchanged. The appropriate authentication methods must also be determined, configured, and implemented, if necessary. These agreements may also include the selection of suitable and unsuitable hardware.

Security: As mentioned in the configuration, security is an essential point where agreements must be made. The procedure for cryptographic key replacement requires proper planning and documentation. If keys fail after a certain period of time, procedures must be in place to ensure that they can be replaced. In addition, arrangements must be made for monitoring the overall system and the individual components, patch management, and other security factors. Neglecting this point can not only endanger a single subnetwork, but the entire system.

User management: For the AAA protocol, it is necessary to define how new users can be created, how they can be logged out, and how temporary conditions can be set. This also includes password policies and which authentication methods and which billing methods are used.

Performance: To guarantee top-notch performance, evaluations are required, and hardware and software must be adapted to meet the requirements. These tests include both performance and system configuration tests.

Support: Who can be contacted by users and service providers if they encounter mistakes? This must be defined, and the corresponding resources must be provided. Supporting users who encounter difficulties with authentication or other network elements is included in this.

Documentation: Documentation is required for all agreed-upon terms and requisite actions. Training should also be offered if needed, so that new participants (users and network operators) can be added. Furthermore, it is essential to provide instruction to all users and administrators involved in the network.

Compliance: It is important that the RADIUS network meet all relevant regulations and standards. Performing regular compliance checks and audits may be required for that.

The above points are not unique to the RADIUS protocol on blockchains, but may become necessary in many decentralized applications. However, there are other aspects to blockchains that we will now examine. We will explore how governance structures can be developed and implemented, as well as the challenges and risks involved. In essence, the objective of governance is to establish the regulations, procedures, and mechanisms that regulate interactions among participants and decision-making within the network. It ensures that the system operates effectively, securely, and in the best interests of all stakeholders. Once again, these arrangements are not in conflict with decentralization. It is important to note that they only clarify standard interactions and types of functions. The respective interactions are executed in a decentralized manner.

Consensus: A significant and prominent component of the blockchain is the consensus mechanism employed to align the data state on all participating nodes. Some consensus methods are presented and described in the section 2.2.2 Consensus Procedures. The Proof-of-Work (PoW) used in the act blockchain is the most popular consensus. Other projects use the as well, but it is in BTC that the most significance takes place. The selection has a direct impact on the scaling, decentralization, and security aspects, thus defining the selection of blockchain technology itself.

Decentralized programs: Blockchain programs (smart contracts), by themselves, require a governance model. This involves ensuring that the programs themselves function properly, that updates are feasible and executed, and that there are procedures in place to resolve issues. Those programs are not always required. Thus, the need must be defined and documented among the participants.

Transparency: One of the strengths and weaknesses of blockchains is their transparency. This leads to a problem of transparency, especially regarding PII and data protection issues. In order to comply with data protection, the procedures must be defined and conceptualized in such a way that PII are protected and will not be compromised later on. Furthermore, it is imperative to discuss the methods employed to make transparency usable, not only in terms of data security, but also in general.

Security: The expression is not unique, but it commands considerable attention. With reference to the CAP theorem (see figure 2.1 on page 11), it means the consistency of the system and thus of the data states on all participating nodes. Often, security is also interpreted as resilience against SPOF. Due to the extensive distribution, partial failures of nodes are not a concern. Neither accidental nor intentional partial failures. Furthermore, the term security is understood to mean the use of suitable cryptography, which is actually used in blockchain ecosystems. Regardless of the interpretation, the agreements on the use and maintenance of security are important and essential for a functioning blockchain system. The hardware requirements for running the cryptographic systems safely are not to be overlooked. Which private keys are located, how are the signatures created, and how can keys and signatures be exchanged securely?

Compliance and Regulations:In this particular section, it is imperative to clarify the regulations under which a decentralized system is to operate. Moreover, these rules must be compared with the regulations of all participating countries and adapted. The different regulations in place in the countries make this undertaking complex from an international perspective, some of which may contradict each other.

Participants: Is there a difference between read and write access? How can new participants be integrated into the overall system? These and other questions are addressed in this category, which ensures the management and implementation of a blockchain system. Here, the keywords are private and public with reference to read access, and permissioned and permissionless with reference to write access.

Fork Management: A fork possesses multiple meanings, as elucidated in chapter 2.2.2 Updates and Forks. The first meaning of forks is the natural forks that occur when new blocks are found at the same time. These forks appear only in probabilistic consensus methods. Random forks are omitted if one chooses a different consensus that does not involve or simulate randomness. Therefore, this option should be considered subject to consensus governance. There are still update forks and state-contradiction forks. Update forks happen when an update is being developed and released. The decision to implement this update lies with each participating node. If it decides not to, its state may be permanently different from the nodes that deployed the update. There's a distinction between hard and soft forks. You can read more about forks in chapter 2.2.2 Updates and Forks. The significance of update forks makes it necessary to include agreements for the implementation of updates in these categories. What's the process for coordinating updates ahead of time? What can be done in the event of a disagreement? What processes are required to achieve a successful update fork? The remaining forks are those with state-contradiction. These occur when a participant does not accept a block and creates one of its own in its place. For the next participant, if both blocks contain only legitimate transactions, one must now decide between the two states. This situation is problematic when there are different understandings of legitimate transactions. If both blocks contain legitimate transactions for all participants, the contradiction can be attributed to a temporary error, which is subsequently and independently clarified by the blockchain system. In other words, the underlying rules that govern a valid transaction are outlined here. Furthermore, processes are defined that regulate the updates and how to proceed in case of contradiction.

Tokenomics: The governance structure must also consider aspects like the issuance of fresh tokens, the control over the money supply, and the distribution of tokens among the project's participants if it puts a token on the blockchain.

Fees: A blockchain requires validators in order to ensure the consensus of a ledger. In public blockchain projects, fees are distributed with a reward to create incentives. These incentives are provided in the form of a token created by the system. The fees also serve as an inherent safeguard against spam attacks, which have the potential to rapidly escalate into a blockchain internal Denial of Service (DoS) attack via the Gossip Protocol. Therefore, it stands to reason that the fees cannot be set to 0. However, if the fees are too high, regular and automated use can be prevented, since a distribution of one's tokens or coins must first be made and adjusted.

The points outlined are crucial elements that must be incorporated into distributed, decentralized structures and programs. The complexity created by decentralization must be reduced through agreements. The

explicit implementation and representation of a transaction can have a significant impact on the overall systems efficiency and security. Furthermore, interactions, such as those between participant arrangements on the consensus and the associated blockchain technology, could overturn a previous preselection and require a new evaluation.
The possibility that such governance regulations could also be organized via a blockchain has not yet been considered. A so-called Decentralized Autonomous Organization (DAO) has this as its goal and could be used to make the votes happen quickly and transparently on a blockchain program. However, previous DAOs have not shown acceptance or even major security vulnerabilities.
The governing principles described must be adapted to the blockchain instance. To give an example, deleting data can cause problems with the system if no adequate solutions are found.4.3.1 Erasing Data on Blockchains, the issue of deletion is described in more detail.
Furthermore, participants may have to respond quickly to future problems, such as a ledger suddenly running full of transactions or the arrival of quantum computers. Regulative measures and provisions for upcoming events, similar to those referred to as Black Swan events, pose a challenge in establishing a comprehensive set of regulations. However, this is a general problem and not blockchain or RADIUS specific.

Governance can be considered a supporting necessity that can be difficult to create due to the large number of participants. Nonetheless, ignoring these guidelines can result in impulsive decisions that could have a negative impact on future developments. Thus, the DAO hack and the reaction to it with a fork by the ETH foundation are still viewed critically today. As a result, the Foundation is frequently accused of being an integral part of ETH. Therefore, Ethereum would not be a decentralized project.
To summarize, governance is the set of rules, policies, and procedures that cover technical, security, and data protection issues, as well as legal agreements. Once these rules and agreements have been implemented, the project is recognized and accepted across different systems and jurisdictions. [155]

4.7 AAA-me

In the previous sections, we discussed the general implementation possibilities of the RADIUS protocol on a blockchain, including its strengths and weaknesses. The following provides a more thorough explanation of custom development. There is a framework called AAA-me that can act with multiple DLTs and implement the functionality of a RADIUS system.

4.7.1 Assignments of AAA-me

The objective of AAA-me is to facilitate the integration of existing RADIUS systems, the development of RADIUS systems, and the connection to blockchain systems. Thus, it will be capable of accommodating both hybrid and full implementation. Furthermore, it is expected that blockchain applications, commonly referred to as smart contracts, will be capable of being monitored and controlled. An extension to this could be the incorporation of a Blockchain program IDE to create blockchain programs. For the actual implementation of these goals, please refer to the chapter 5 Implementation. The frameworks and libraries used are discussed there. Here, I am remaining conceptual and presenting the anticipated implementation objectives.
In order to identify and examine the tasks of AAA-me, we included experiences from previous projects, regardless of whether they were successful. An overview of projects related to public administration can be found in section 3.2 Digitalization and Blockchains for Public Administration.

At least one blockchain infrastructure must be present for each of the three interface implementations. Therefore, the framework provides the possibility of connecting to a blockchain instance. The prerequisites for such a connection are specified by the relevant blockchain technology and instance. To ensure that

AAA-me's utility is as broad as possible, the system utilizes a variety of blockchain technologies. Initially, these are ETH, BTC, and Hyperledger Indy. Other technologies are currently being evaluated and may be added.

Other tasks related to blockchain interfaces include monitoring the current status of the network, providing interfaces for interaction with the respective blockchain instance, and obtaining data from different nodes and comparing them with each other. The last step is to make sure that there isn't a cut-off from the network or a Man-in-the-Middle (MITM) attack. To accomplish this, it is necessary to create and maintain a constantly updated list of nodes. There is a pre-defined number of nodes connected to. The information queried from the blockchain is compared to each other. If all the information is identical, we can utilize it to accomplish additional tasks. If the data is inconsistent, it must be investigated where and how this occurred. This task can be accomplished manually after a warning message sent to the user or administrator. One alternative is to establish a threshold value, which will determine the number of inconsistencies that can be automatically processed. It's important to note here that the consensus procedures should be used to match the nodes for consistency. However, an attacker could attempt to isolate the connected node and manipulate the input of the data. This would be comparable to a MITM attack when roaming within RADIUS systems, albeit with a more elaborate approach.

It would be too expensive to develop a separate node for different blockchain projects within AAA-me. The process of creating and managing a network element is extremely involved, and the requisite security guidelines render it impractical. Therefore, only interfaces to node instances come into question for AAA-me. In order to accomplish this, it is attempted to implement as many interfaces as feasible. This implies that a diverse range of blockchain technologies are enabled for the realization and integration of AAA functionalities. This calls for building modular structures.

In addition to having control over blockchain instances, we also need to have control over blockchain programs (smart contracts). Like the blockchain instances, this includes the monitoring, provision of interfaces and control of these programs. AAA-me provides a selection of predefined authentication schemes as templates, so these can also be used with the framework on the respective blockchain instance.

Summary of AAA-me assignments:

1. Blockchain instances
    - Monitoring
    - Interfaces deployment (documentation, adjustments, ...)
    - Node list administration
2. Blockchain Programs (Smart Contracts)
    - Monitoring
    - Control and Interaction
    - Interface description
    - Blockchain programs across different blockchain instances and technologies.
    - Deploying and Maintaining
3. Authentication
    - Methods Maintaining
    - Documentation
4. RADIUS Systems

- Monitoring
- Control and Interaction
- Interface description

In conclusion, AAA-me will be designed and developed as a configurator. After configuring your setup, it is expected that you will no longer require it. Different monitoring options are provided in order to extend the range of functions. Thus, the software gains considerable significance with minimal additional work on its part.

4.7.2 Frameworks

The use of existing frameworks, libraries, and Application Programming Interfaces (APIs) is essential and crucial in today's software development. This enables developers to focus on one part of the development and adapt the results of other developments. Using this method saves both time and money. Moreover, using frameworks can improve the structure of a program. By using standards, we can achieve higher quality and higher quality applications. With APIs, methods and data can be made available to third parties or integrated into one's own system. Furthermore, frameworks provide the prospect of external assistance and the advancement of specific programming components. Thus, developers can be assigned a focus that is specialized for them, which will significantly accelerate development.
Using external libraries and frameworks, on the other hand, also creates dependencies that can lead to problematic outcomes. Many of the available libraries are maintained by leisure and hobby developers in their spare time, thereby limiting their capacity to satisfy essential security standards. One is also dependent on external providers. The adjustment and maintenance of the own application can be difficult if the support is discontinued or the API is changed. The degree of complexity of the own program can rise, as interdependencies can have a negative impact on each other. This can lead to conflicts if external libraries use other libraries or frameworks.
In conclusion, frameworks and APIs offer developers many benefits that improve the efficiency and quality of software applications. Although there are some potential concerns, the benefits usually outweigh the risks, as long as developers are aware of potential risks and take appropriate steps to manage them. The following steps should be taken by developers to maximize the benefits of frameworks and services:

1. Careful Selection: It is imperative to conduct thorough research in order to identify the most suitable frameworks and libraries. Consider factors such as popularity, community support, documentation, and security measures.

2. Monitoring and Updates: It is imperative to verify the current status of all frameworks utilized for assessing the current state of the system and assessing potential harm and vulnerabilities. These factors have the potential to persist and have an impact on in-house advancement. You should be able to perform quick updates in case of damage to avoid failure or own damage.

3. Security: The points in Monitoring and Updates should be extended to security aspects. For instance, a library designed for a cryptographic system may persist in operating securely and exhibiting excellent performance, despite the possibility of a vulnerability being discovered by the mathematics. It is important to observe and adhere to security best practices.

4. Performance: In order to achieve optimal performance, developers should ensure that their applications are regularly tested and optimized. This can be done by refactoring the code, doing load tests, and using caching techniques.

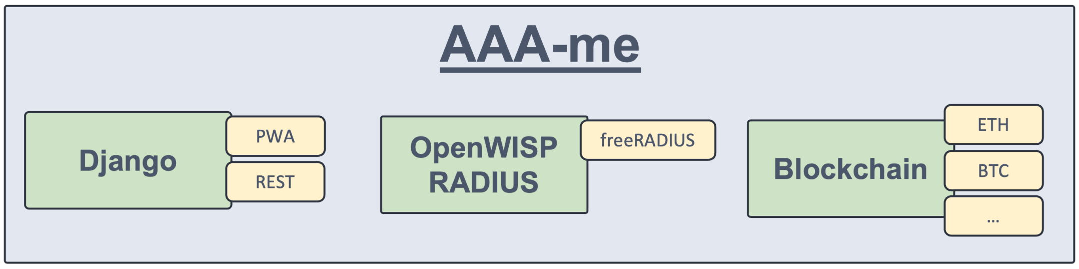


Figure 4.8: High-Level overview of used Frameworks

The four points mentioned above were taken into consideration when selecting frameworks for internal development. A high-level consideration of the frameworks are:

1. **Django:** The Django framework is a framework for creating websites using Python. The system implements a Model-Template-View architecture system, which is comparable to the more renowned Model-View-Controller architecture system. There are objects, methods, and procedures that are responsible for the respective application areas. The model manages database connections and administrations, the template prepares the websites' data, and the template implements the representations. The natural implementation of the Python language made this framework preferred over other frameworks. This framework is the best choice for this thesis because it has extra security and performance features. [84]

   Progressive Web App (PWA): For a broader application, AAA-me is also designed as PWA. Basic functions and important data are to be made available to the user as an executable service. The only necessary tool the user requires is a web browser. The most important wallet functions for creating a private and public key pair, signing a message and creating transactions are examples of those basic functions. For the realization of a PWA JavaScript is integrated over the templates of a Django App. For full functionality, other frameworks and libraries such as Ajax or jQuery are used.

   Django REST Framework (DRF): An open source project for creating APIs in Django website projects. It allows developers to create RESTful APIs (Representational State Transfer) that can be used by web and mobile applications, IoT devices, or other systems that rely on web APIs. This framework is essential in AAA-me in two aspects. To construct PWAs efficiently, APIs are an indispensable resource. So, the internal methods are available for external use.use. The second necessity is the integration of other modules, such as OpenWISP RADIUS. There, AAA methods and observation and management functions are provided per DRF as API. Therefore, it was decided to stay with DRF and not move to newer API frameworks, such as Django Ninja. Despite the promising new features that the latter offers. [139, 232]

2. **OpenWISP-RADIUS:**

   OpenWISP is an open-source project that aims to manage and monitor network infrastructures. This software is mainly written in Python and has a lot of modules to help you reach your goals. First of all, the OpenWISP RADIUS module is essential for AAA-me. But other modules, like OpenWISP Monitoring, are exciting additions. OpenWISP is particularly useful for handling wireless networks utilizing OpenWrt-based devices, but it's also capable of being used with other network gadgets.

   The OpenWISP RADIUS project describes itself as follows: "OpenWISP RADIUS provides a web interface to a freeradius database, a rich REST HTTP API and features like user self registration, SMS verification, import of users from CSV files, generation of new users for events, social login, and much more." [204] The integration possibilities for Django projects to already existing RADIUS systems are given and explained.Thus, this framework is specifically used to achieve RADIUS integration for AAA-me.

FreeRADIUS: freeRADIUS is the software used to implement a RADIUS system. It's an open-source project that implements Internet Engineering Task Force (IETF) reference documents for AAA tasks. More information about AAA in chapter 2.7 Tripple-A (AAA) Technologies and on RADIUS in chapter 2.7.1 RADIUS Protocol. It supports the authentication techniques mentioned in the preceding sections while remaining stable and adaptable. Furthermore, it is modular, which makes it easier for a developer to integrate additional functions. Besides, the fact that freeRADIUS is accountable for one third of all authentication procedures on the Internet, as per its own data, speaks in favor of freeRADIUS [108]. The high degree of adaptation makes it more secure to continue working on the project, further develop it, and document it in the future than in other ones.

3. **Blockchain and DLT:** This module contains a variety of libraries for interacting with various blockchains and DL- technologies. The different configurations of these blockchains make it more difficult to integrate them into AAA-me. There is a general blockchain module that contains the functions needed for AAA-me to connect with a blockchain in order to realize this promise. The blockchain projects provide libraries in other parts of their software that are adapted for the blockchain module. These libraries are used to make the blockchains work with and through AAA-me. Using the principles of object-oriented development, new modules for new network technologies can be easily crafted through interface descriptions. A small list of already integrated networks, are:

   Hyperledger Indy: Hyperledger Indy is a project by the Hyperledger Foundation that is open-sourced and focused on digital identities. Hyperledger is a project that is hosted by the Linux Foundation and works on various blockchain and DLT technology projects for enterprises. The Hyperledger Indy project is conceived as a digital identity endeavor in terms of Self-Sovereign Identity (SSI) execution. Other Hyperledger projects include Besu (a client for ETH) and Fabric (a general blockchain framework for making your own network). [145, 249, 7]

   Two additional projects are significant for Hyperledger Indy. Some other projects may utilize them. These are called Hyperledger Aries and Hyperledger Ursa. Aries is responsible for direct user communication. This includes establishing a safe means of communication, handling information in local and cloud-based storage, key management, and other features. Aries, in addition to ensuring safe communication, can also be viewed as a wallet in the sense of the BTC network. [7, 117, 146]

   Ursa is a library for the implementation of cryptographic algorithms. Cryptography is an intricate discipline, and its implementation is equally challenging. To disperse fewer resources across all projects of the Hyperledger Foundation, Ursa was employed to consolidate previously acquired resources. All projects have the opportunity to use the results from Ursa. Being a free software endeavor, the strategies can be applied to other areas as well. [117]

   ETH: Ethereum is a second generation blockchain, which means it comes with Turing-complete programming. Being the first of its kind, it attracts numerous programmers. Furthermore, it establishes many prerequisites for further blockchain projects with Turin-complete programming. New projects will not be able to circumvent the provision of Ethereum support or any other form of dealing with this technology. The web3.py library is used to establish a connection with either the private or the public Ethereum network. This library is a Python implementation that was initially influenced by the web3.js library, but has since shifted its focus towards catering to the requirements, ease-of-use, and guidelines of Python developers. It can connect to all Ethereum-based blockchain systems. The package includes a link to the distributed ledger, administration of Ethereum accounts and wallets, interaction and the execution of smart contracts. Additionally, data from the blockchain can be accessed. Since a Web3 library is not a node, it must connect to one. In the attachment D Private Ethereum

Network a possible setting of the node can be seen, so that the library Web3.py can execute the desired functions. [149]

BTC: Bitcoin is the first blockchain that has been developed. The software provides support for a transaction-based scheme and possesses a limited set of functionalities within its own programming. The minimalist approach is expected to enhance the degree of protection. Bitcoin has a large community that continues to develop not only the technology itself, but also further solutions to address known blockchain-related problems. Similar to the libraries for ETH, there are libraries for BTC. The Python BitcoinRPC module is used for AAA-me. The range of capabilities for interacting with a node is smaller than for ETH, as there are no full wallet capabilities available. One authenticates to a node where any Remote Procedure Call (RPC) commands are provided. A file is used for this authentication. This improves the level of security while maintaining the same level of convenience as common username and password combinations. [115]

4.7.3 Architecture of AAA-me

An abstract high-level plan for the realization of the project is provided in this section. A more detailed view can be found later. The figure 4.8 can be used to illustrate the structure of the architecture, as the mentioned frameworks are used in AAA-me. Furthermore, it is described here how AAA-me can be integrated.
Chapter 4.7.1 Assignments of AAA-me was used to describe the tasks of AAA. These were essentially the configuration settings for RADIUS and blockchain and monitoring systems. It can be deduced that a modular design for already existing network infrastructures is more target-oriented and promising. In such a configuration, it is possible to utilize the already implemented features and components of FreeRADIUS and execute AAA tasks within the network. It's irrelevant whether a system has already been incorporated. If a RADIUS system is already in place, there is less to configure on the RADIUS server.
Modular integration can be accomplished in two distinct ways. In FreeRADIUS, it is possible to write modules, which can be activated by the administrator. The activation is usually done via configuration files, which can be found in the corresponding folders of FreeRADIUS. Therefore, a blockchain module could be developed that enables the administrator of the RADIUS system to initiate and oversee a connection to a blockchain. This approach has the advantage of being easy to integrate into existing FreeRADIUS instances. Furthermore, there is also the possibility of writing a program that implements corresponding interfaces with FreeRADIUS. The library OpenWISP-FreeRADIUS is intended for this purpose. It is possible for AAA-me to directly control the permissions and accounting of FreeRADIUS databases without negatively affecting the existing system, provided that write processes do not destroy the necessary table structures in the database.
The interfaces play a crucial role in the communication between AAA-me and other components. Many of these are implemented by APIs in the nodes. The corresponding blockchain nodes are accessed by AAA-me through APIs. Furthermore, AAA-me extends its capabilities to other services via APIs. Frameworks are used to ensure that the API is functioning properly. By supplying AAA-me's services as an API, it's straightforward to set them up and use them as a configuration tool and for monitoring. The interfaces also enable the implementation of PWA services, which makes it possible to implement AAA-me functionality on all devices that can run a web browser. Provided that the necessary security modules are implemented, the custody and management of secret keys can be managed within an alpha version.

Governance

Governance is an essential component for the integration of new features. Particularly, when the participants are distributed, such as in RADIUS or blockchain systems. But the vast majority of agreements and rules are negotiated on a bilateral or multilateral basis. Furthermore, these are essentials before the initiation of the

initial emergence and implementation. As a result, governance cannot be regarded as such within AAA-me itself. In the actual implementation, as a result of governance agreements, one finds it. The necessity and factors of governance are considered and described in chapter 4.6.
Since AAA-me has few intervention possibilities, there is no explicit representation for an intervention or configuration here. It is, rather, distributed in different configurations due to the agreements. Since even the smallest individual conditions will have a large impact on the decisions, no general template is defined.

Blockchain Programs for AAA-me

The possibilities and necessities of blockchain programs, better known as smart contracts, were discussed in great detail in chapter 4.3 Programming on Ledgers and chapter 4.4 Blockchain Transactions and Programs for Triple-A functions. An overview of the templates already available in AAA-me is given in this section. Various programming methodologies and examples were examined. This is a general overview of these programs. The low-level consideration, with lines of code, can be found accordingly in chapter 5 Implementation.

- A preliminary template facilitates the employment of a Proxy Contract, thereby enabling the administration of programs following the Data Segregation Design Pattern to be feasible. For setting a name to an address, setter, and getter methods are implemented in this template. This feature can be utilized to gain access to methods of other applications. Therefore, updates are also feasible on Blockchains.
- A certificate according to the X.509 standard is used as an ERC-721 program. An ERC-721 contract is also referred to as a NFT. These contracts have an owner who has the ability to transfer ownership of the contract. In addition, these contracts are clearly not fungible, which is where they got their name from.
- The two certificates shown in the class diagrams 4.7 Class Diagram of Certificat Representations (a) and (b) are implemented as templates. These protocols enable the deployment of certificates over a blockchain through a hierarchical system or a Web of Trust (WoT) method.
- There is a contract template that implements the Password Authentication Protocol (PAP) authentication. The username and password are stored in the contract for this authentication method. This contract was created for research purposes and should not be found in production systems. There is a second version of the implementation that addresses the transparency issue and does not store PII on the ledger, but can retain the functionality. It is done with Zero-Knowledge Proof (ZKP).

These program templates can be configured individually for deployment and then implemented on the ledger. In addition to the deployment process, it is also possible to manually enter access information in AAA-me, such as the public address of a program instance and the Application Binary Interface (ABI), in order to utilize the monitoring capabilities. In addition to this, there is a possibility of improved protection of potential PIIs by using the ZKP. Nonetheless, this necessitates the implementation of a fresh version of the template, as the interactions with it undergo modifications as well.

4.8 Application Examples for AAA-me

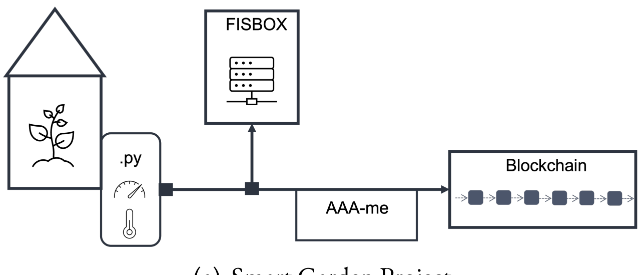
The RADIUS protocol can be shown and explained using application examples like eduroam and mobile networks. In this chapter, an extension of these examples is attempted with AAA-me. What are the potential motivations, and what advantages does integrating an existing RADIUS protocol with AAA-me bring? The examples are not considered separately since the exact implementation, whether to run RADIUS systems hybrid with AAA-me or replace it entirely, does not affect the example applications.

The first instance of an AAA-me-based application can be found on both mobile and eduroam networks. Both of these applications are the primary uses of the RADIUS protocol, and as such, the protocol provides a suitable implementation for AAA-me. The objective of implementing AAA-me and thus blockchain technologies is to enhance resilience to SPOF. This is because these vulnerabilities are built on individual elements of the system, which could cause the system to collapse. The failure of this component will have a significant impact on the entire system, not just the local one. In the case of the mobile network, this could be due to a radio tower failure. Customers can't use this tower to connect to the network anymore. The eduroam example would be comparable. Furthermore, roaming might also cease to function if there is an interruption in the proxy chaining of the requests forwarding.

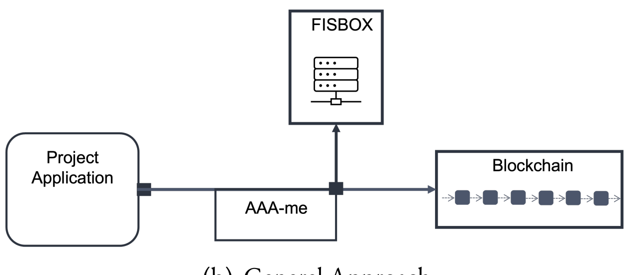
Such a failure could be caused by an accident. However, it would also be possible to cause deliberate failures caused by external attacks. Blockchains and DLTs may be suitable for avoidance, as these systems have been specifically designed to be resilient against SPOFs. Authorization requests from users could thus be routed via alternative NAS, and billing via alternative radio towers would be possible in the mobile sector.

In another instance, we intend to leverage the potential of distributed blockchain to facilitate a Single Sign-On (SSO) system. In SSO, the user authenticates exactly once and has access to multiple independent applications, such as software. The potential of SSO has already been recognized by most organizations and is already being utilized by numerous ones. It is much easier for users to navigate networks when the authenticity check is centralized. This prevents numerous errors. However, the user is dependent on the service's functioning. An unsuccessful attempt would have devastating consequences. As a result, resilient systems are essential, and the use of a blockchain can be discussed. Thus, AAA-me could be employed to handle a digital identity and transport it from its own network to other ones. This roaming feature thus extends SSO functionality across different networks and is more failure-resistant than existing systems. It is possible to implement a SSO at the physical level using the network extension remotely. This would enable the employee to be authenticated upon their entry into the premises, thereby facilitating further access to their designated workstation. In addition, a security system would already be used to provide access to the computer and phone at work. A system of this nature would significantly facilitate the users' access. However, there exist additional factors that have the potential to pose a threat to the entire network. It's not a good idea to rely on the security potential of the blockchain, so a separate investigation is needed.

Another example of a potential application would be the possibility of adding additional authentication methods without having to modify the RADIUS client or server. In this instance, the blockchain would act as an authentication translator and modify the authentication data supplied by the user to enable its comprehension and processing by the RADIUS server. An example of such a method extension could be a passwordless authentication method using certificates. The user would send their certificate for authentication, and the blockchain would respond to the RADIUS server with the answer. The process would be described in greater detail as follows: If the user now attempts to authenticate using a certificate located on the blockchain or a PKI, AAA-me will take the necessary authentication steps with challenge-response procedures and, if successful, will generate a 'radius token,' which will be deposited in the respective databases of the RADIUS, with which the user will be able to move around the network. It would not be necessary to customize the RADIUS system. Even if the RADIUS server already has methods, new methods could be developed that make an update necessary or desirable. Depending on the software used in the RADIUS implementation, there are several options available. The FreeRADIUS implementation employed exhibits a significant degree of modularity. It should be possible to update easily. However, in this variation, customization is done in one place. If you have more RADIUS developers but fewer blockchain developers, customization might be more purposeful and less expensive at this point. But if you want to add a method in different places at the same time, the Blockchain implementation might be the best choice.



(a) Smart Garden Project



(b) Generel Approach

Figure 4.9: Project Architecture of FISBOX with AAA-me

4.8.1 Adoption in Hessian Central Office for Data Processing (HZD)

As a participant of the research cooperation between Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD) and Frankfurt Institute for Advanced Studies (FIAS), HZD has expressed the desire to explore the possibilities of integration into its own systems. By implementing this approach, the integration possibilities need not be restricted to the RADIUS protocol. A comprehensive exploration of the blockchain and DLT can be delivered.
During the duration of this doctoral program, a digital WLAN infrastructure was established in Hesse, with the aim of facilitating access for every Hessian through a WLAN router. The project for this started before the PhD in September 2018[106] and is currently still ongoing [109]. The proximity to the eduroam project, which allows students to connect to networks at universities besides their own, makes it sensible to apply the same functionality to the Hesse WLAN via AAA-me. Since the project isn't under the direction of HZD, these considerations have been pondered and theoretically explored based on its technical viability.
The specifications of the Hesse WLAN were taken for this purpose. In these, the requirements and processes given by Hesse WLAN were identified. Subsequently, these requirements were compared with the possibilities of RADIUS and AAA-me.

A "Smart Garden" should be created as part of a research project of the HZD. During this project, the feasibility, and problems of developing a gardening, water, forestry, and agriculture supported by hardware and software should be investigated. The Smart Garden projects focus on automatic ventilation, watering, and irradiation with UV lamps, as well as plant monitoring and environmental enhancement. Only greenhouses were investigated in the projects. Future discussions will include forests and fields that could be monitored and analyzed by drones.
So, greenhouses were thought of and built, where strawberry plants were to be grown all year long. Sensors monitored the temperatures, moisture levels of the soil and air, as well as the conditions of the plants. According to the sensor data, the windows were opened via motors for ventilation or the water pumps for watering, respectively. There were no real plants used in the lab environment. Software generated simulations of these.
In the figure 4.9 (a) the Smart Garden project with an integration of AAA-me is shown. The greenhouse is visible on the left. Sensors are affixed to it, enabling the simulation of air and soil moisture, as well as temperature measurements. This information is used to execute procedures that have been pre-determined and programmed. This was simulated by a corresponding output on the screen. The data generated in the Proof of Concept (PoC) is initially stored in a company-owned database (Fisbox) without blockchain. With reference to blockchains, one can recognize the representation in the figure on the right and also find AAA-me. The AAA-me software was employed to set up the interface to the Blockchain and to further monitor the associated Blockchain programs (Smart Contracts). This data allowed the blockchain to trigger events, which the greenhouses control software was able to trigger without my involvement in order to respond to

them. The objective of this endeavor was largely practical with a focus on a Smart Garden, but concepts of a not deniable log integration were also examined with keen interest. The objective was to generate data for a legally secure setting that could substantiate any interaction. A private ETH instance was used for this purpose. With that configuration, it is possible to establish automated procedures for any process within the Smart Garden, incorporating other greenhouses or system-relevant institutions.

Another consideration for a larger adoption opportunity was made possible by the FISBOX® platform developed in-house.
"Mit der FISBOX® als moderne, virtualisierte Betriebsplattform bietet die HZD eine standardisierte, flexible Software-as-a-Service-Lösung (SaaS) für Fachinformationssysteme an – eines der zentralen Werkzeuge zur Umsetzung des Onlinezugangsgesetzes (OZG) und ein wichtiges Mosaik für effiziente Verwaltungsabläufe in der Digitalisierungsstrategie des Landes Hessen.[2]"[225]
FISBOX® is a modern, virtualized operating platform provided by the HZD. It serves as a standardized and adaptable software-as-a-service (SaaS) solution for specialized information systems. It is a key part of the Online Access Act (Onlinezugangsgesetz, OZG) and an important part of the digitization strategy of the state of Hesse.
The state of Hesse and local authorities utilize 50 specialized information systems that cover a diverse range of topics, including mining, genetic engineering, budget management, and funding procedures. New applications or transfers from outdated IT systems are added almost monthly. The FISBOX® is always adapted to meet the requisites and can be deployed quickly.
Using the FISBOX® system, you can optimize internal administrative processes in many ways and map both simple, standardized business processes as well as highly individualized, complex business processes. Even parallel processes that take place during an application process, such as communication, control, and information procurement, are fully supported by FISBOX®.
[225]
The already high availability of the FISBOX® within the HZD in numerous projects makes it possible to provide an interface to the blockchain through a thoughtful and secure implementation and design. Thus, all projects that were previously running on the FISBOX® can be ported to the blockchain. Furthermore, the FISBOX® can be further extended by the Blockchain through further implementations, which would significantly increase its functional scope. The high acceptance rate permits a lower development effort with greater integration potential.
Figure 4.9 (b) shows a potential high-level architecture that routes a data stream through AAA-me beforehand and stores it in the FISBOX® and blockchain according to the configuration.

[2]With FISBOX® as a modern, virtualized operating platform, HZD offers a standardized, flexible software-as-a-service (SaaS) solution for specialized information systems - one of the central tools for implementing the Online Access Act (German: Onlinezugangsgesetz) (OZG) and an important mosaic for efficient administrative processes in the digitization strategy of the state of Hesse.

Chapter 5

Implementation

This chapter presents the actual implementation of the concept from the chapter 4 Methodology and Conception is presented. Thus, its explanation is in complete alignment with the actual code. Here, the goals for AAA-me are the focus. In later chapters, ideas for a further development are described.
The structure of this chapter is as follows: Initially, the architecture of AAA-me is discussed in detail. This includes the respective module implementations, as shown in the figure 4.8 in chapter 4.7.2 Frameworks on page 79. My findings during programming are supplemented by results from the literature, as described in chapter 3.3 AAA-Technologies on Blockchains - Related Work. Additionally, the results of developments in public administrations are included.
The chapter continues with an experiment investigating resilience to Single Point of Failure (SPOF) with respect to Denial of Service (DoS). During these tests, different setups are presented and implemented. The experiments are arranged in a variety of settings, ranging from local environments to remote connections over the Internet.
The final section of this chapter discusses the possibility of adopting AAA-me for public administrations, mentioning Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD) as an illustration.Another research project on a smart garden is used here to identify any interfaces and to identify strengths and weaknesses.

5.1 AAA-me

As a reminder at this juncture, it is noteworthy that AAA-me is intended as configuration assistance. This is derived from the notion that the decentralization should not be compromised by a central interface. For further profitable application possibilities, monitoring, Progressive Web App (PWA) and further characteristics are to be made available.
To provide an overview, here are the possible applications of AAA-me:

1. Configuration of a Remote Authentication Dial-In User Service (RADIUS) system for Authentication, Authorization and Accounting (AAA) tasks.

2. The configuration of an interface to a blockchain for the execution of AAA tasks. Optionally, this can be integrated into an existing RADIUS system.

3. Set up and implement functions for the PWA module. Important here is to make sure that the functions run in safe execution environments.

4. Interface descriptions for smart contracts on the private chain.

5. Monitoring of RADIUS systems, blockchains, and blockchain programs. This includes physical and virtual parts of each mentioned system.

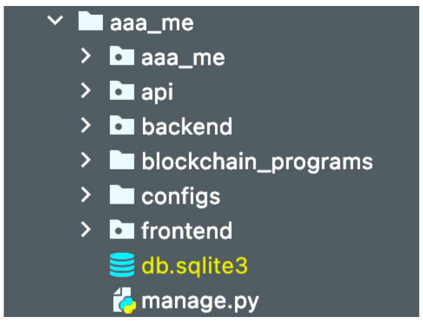
6. The provision of assistance for the management of blockchain systems. The governance itself must be defined independently of the technology. The implementation, however, may be subject to AAA-me. Some examples include the creation and distribution of a token. This may prevent DoS attacks from being started from the system and keep the systems working.

5.1.1 Architektur

A high-level view of the architecture can be found in 4.7.3 Architecture of AAA-me. This section provides detailed instructions for setting up and using the corresponding modules.
The Ethereum (ETH) blockchain technology is used as an example. This has already gained a significant amount of popularity, numerous projects are currently operating on it, and numerous tutorials are available for perusal. Moreover, it is a straightforward process to establish your instance, encompassing monitoring options. With Proof-of-Authority (PoA), the private instances are also good and cheap to manage.
Other blockchain solutions like Bitcoin (BTC) or Indy are similar and will be explained briefly. Due to the modularity structure of AAA-me, the utilized blockchain does not exert a significant influence on itself, with the exception of the blockchains programming with smart contracts.

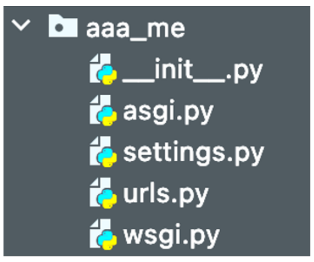
Django Module - Web Application

Django is a Python backend framework that is utilized for developing web applications and Application Programming Interfaces (APIs). In chapter 4.7.3 Architecture of AAA-me, Django is already introduced. This framework is the foundation of AAA-me and must be installed and configured first. OpenWISP is a Django framework that can be used in Django projects. After initializing a Django project, this is done. After this explanation, OpenWISP-RADIUS is described.
The process of starting a new Django project is covered in numerous instructional videos and manuals. At this point, it is important that with every new setup of a project, a fixed folder structure is given. This is, usually, a root folder with the project name, from which only a subfolder with the project name is initially incorporated. In the figure 5.1 (a), this would be the two "aaa_me" folders. The first folder is the root directory. All other folders of the project are located in this one. The second "aaa_me" folder is essential for configurations. In this folder, you can find the general Settings.py file, where the mentioned configurations are set. For OpenWISP this file will have to be adapted.
A well-organized project plan is suggested for the undertaking. Therefore, AAA-me is designed to separate the frontend and backend methodologies. These can be found in the "backend" and "frontend" folders accordingly. The Django REST Framework (DRF) library also suggests its own folder structure for easy integration. The corresponding folders are created in the project using a Django command. This creates database links and other configurations in addition to the folder.



```
aaa_me
  aaa_me
  api
  backend
  blockchain_programs
  configs
  frontend
  db.sqlite3
  manage.py
```

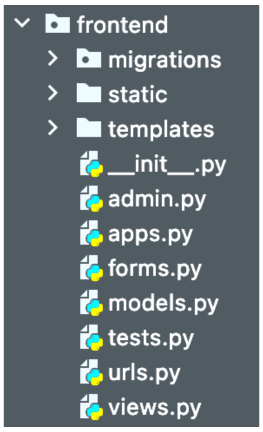
(a) Total Project Structure



```
aaa_me
  __init__.py
  asgi.py
  settings.py
  urls.py
  wsgi.py
```

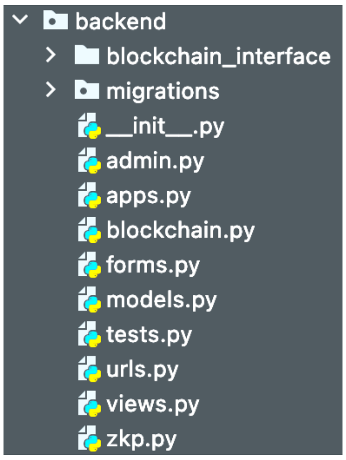
(b) aaa_me Structure

Figure 5.1: AAA-me Django Folder Structure



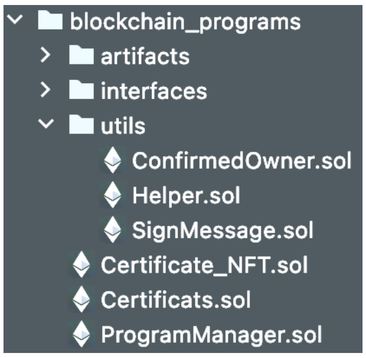
```
frontend
  migrations
  static
  templates
  __init__.py
  admin.py
  apps.py
  forms.py
  models.py
  tests.py
  urls.py
  views.py
```

(a) Frontend Structure



```
backend
  blockchain_interface
  migrations
  __init__.py
  admin.py
  apps.py
  blockchain.py
  forms.py
  models.py
  tests.py
  urls.py
  views.py
  zkp.py
```

(b) Backend Structure



```
blockchain_programs
  artifacts
  interfaces
  utils
    ConfirmedOwner.sol
    Helper.sol
    SignMessage.sol
  Certificate_NFT.sol
  Certificats.sol
  ProgramManager.sol
```

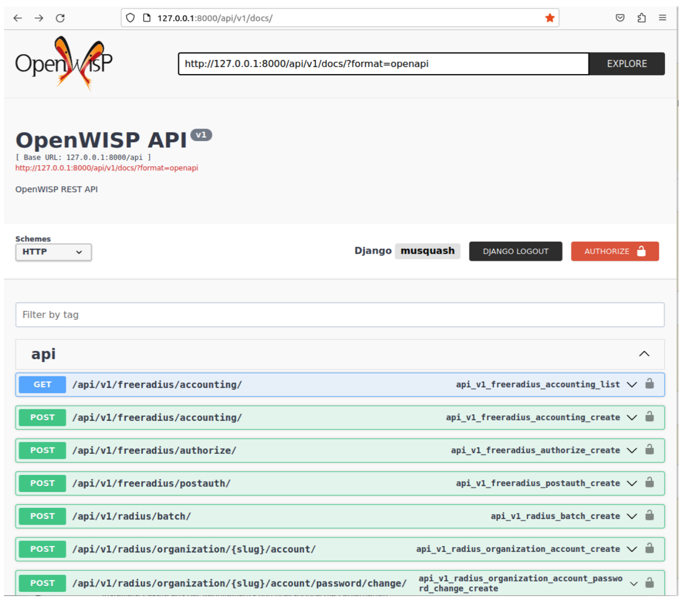
(c) Blockchain Programs Structure

Figure 5.2: Project Folder Structures

There are templates for programs that can be deployed on the blockchain in the "blockchain_programs" folder, which was created manually in the directory.
Next to the folders, you can find the database file and a Python file "manage.py" necessary for the control and administration of the Django web server. With the Python file, set configurations, database adjustments and the start of the web server can be controlled directly. All interaction of the administrator runs through this file.

The project folder structures shown in figures 5.2 (a) to (c) are a visual representation of the folder structures of the backend, front end, and blockchain applications. Django created the folders in (a) and (b) and by default, they contain some files and the migrations folder. This migrations folder has been created to facilitate the necessary migration procedures, which involves the transfer of the appropriate database models to the database. Furthermore, the data that is essential for the rendering of a website is automatically generated within the folders created by Django. These files include apps.py, __init__.py, admin.py, models.py, views.py, and test.py. The files views.py, urls.py, and models.py are significant for further programming of AAA-me. As outlined in section 5.1.1 Architektur, Django possesses a model-view-controller architecture. This architecture is implemented in the three mentioned files. The model.py file encapsulates the database models that transform a dataset into a database schema during the execution of Django. In the vies.py files, functions are written in Python, which can react to database models from the models, but also to potential inputs. Additionally, the processing of this data is determined. The frontend folder also contains a template that is selected to display the processed info to the user. The urls.py files contain the hyperlinks to the corresponding websites.
Two additional folders were created for the frontend, which are necessary for the integration of JavaScript and HTML templates. These folders, which are designated as static for JavaScript code and templates for HTML sites, must be set in the Setting.py file located in the aaa_me folder. More detailed program code will be discussed in the later section. There are templates in forms.py that represent a request for interaction with database models. So that data from the user can be used to create an interaction with the database models.

The folder structure for the backend was created also by Django, which has large overlaps with the frontend. The main difference lies in the necessary programming of the views, Uniform Resource Locators (URLs) and models. The backend describes the mechanics of interacting with blockchains. A modular approach is available as a blockchain-agnostic system. In blockchain.py, general methods are described as interfaces that need



127.0.0.1:8000/api/v1/docs/

http://127.0.0.1:8000/api/v1/docs/?format=openapi EXPLORE

OpenWISP API v1

```
[ Base URL: 127.0.0.1:8000/api ]
```

http://127.0.0.1:8000/api/v1/docs/?format=openapi

OpenWISP REST API

Schemes HTTP — Django musquash — DJANGO LOGOUT — AUTHORIZE

Filter by tag

api

```
GET   /api/v1/freeradius/accounting/                                api_v1_freeradius_accounting_list
POST  /api/v1/freeradius/accounting/                                api_v1_freeradius_accounting_create
POST  /api/v1/freeradius/authorize/                                 api_v1_freeradius_authorize_create
POST  /api/v1/freeradius/postauth/                                  api_v1_freeradius_postauth_create
POST  /api/v1/radius/batch/                                         api_v1_radius_batch_create
POST  /api/v1/radius/organization/{slug}/account/                   api_v1_radius_organization_account_create
POST  /api/v1/radius/organization/{slug}/account/password/change/   api_v1_radius_organization_account_passwo
                                                                    rd_change_create
```

Figure 5.3: API Overview of Provided Functions for AAA

to be mapped to the corresponding blockchain and Distributed Ledger- (DL-) technologies. These mappings are programmed in the blockchain_interfaces folder. The zkp.py file provides methods for the Schnorr-Sigma Zero-Knowledge Proof (ZKP) procedure. These can be used for other procedures in the views, or directly in the corresponding blockchain interfaces.

The folder shown in figure 5.2 (c) contains a collection of ETH Smart Contracts and an artifacts folder for the Application Binary Interface (ABI) files created by the compiler. Given that the Smart Contracts are also modular in design, there is a folder for the interfaces and one for the helper modules in the utils folder.The smart contracts shown were made for research purposes and don't show all the blockchain program templates that are available. These details will be explained later.

The folder structure, for which APIs is not shown in detail in the images, is very similar to the structure for frontend (Figure 5.2 (a)). In this, the URLs are defined for the respective calls. Otherwise, the other methods are defined in OpenWISP-RADIUS and therefore not programmed in the corresponding data. Thus, except for urls.py, the folders are all still in the default state.
If one would like to implement PWA functionalities, it could be that methods must be provided here and then still further links into the urls.py must be written and functions in the vies.py programmed.

OpenWISP-RADIUS

As mentioned earlier in this chapter, OpenWISP is a framework for controlling network components in order to control or implement a RADIUS system. It is a modular framework that provides a diverse range of modules for the realization. Of particular interest to AAA-me is the OpenWISP-RADIUS module, which allows us to build an interface to a FreeRADIUS server. The module can be configured to write to the FreeRADIUS database models. By default, OpenWISP-RADIUS uses the user models of Django and its database. This is because other functions, such as authentication and role models, are already provided for these models. To set up the interface, it is important to add the following data to Installed_Hosts in the Settings.py file in aaa_me/aaa_me/settings.py as you can see in code 5.1.
These settings make Django extend the admin pages with settings for OpenWISP-RADIUS. This is done by adding user models and other features to the authentication. This enables the utilization of authentication

methods obtained through FreeRADIUS to establish a connection to the website provided by AAA-me. Furthermore, it is possible to create users in the admin panel and grant them any permissions they may have. These users can then access and use the APIs according to their permissions. In the figure 5.3 an overview and documentation of available methods to interact with the FreeRADIUS server is given. There, you can also directly call and execute the API. Further necessary settings to use a Django project with Open-WISP-RADIUS can be found in the documentation of the project. [204]
Here, only authentication was discussed. But the group role assignment for users and the “accountings” table in the admin overview hint at the further functionalities. You can see this in figure 5.3 API Overview of Provided Functions for AAA. More about the integration of accounting and authorization in a later section.

```
    # openwisp admin theme
    'openwisp_utils.admin_theme',
    # all-auth
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    # rest framework
    'rest_framework',
    'django_filters',
    # registration
    'rest_framework.authtoken',
    'dj_rest_auth',
    'dj_rest_auth.registration',
    # openwisp radius
    'openwisp_radius',
    'openwisp_users',
    'private_storage',
    'drf_yasg',
```

Code 5.1: Django Configuration with ../aaa_me/settings.py

Blockchain Interfaces

The connection to blockchain nodes can be made in two variants. It is important to comply with the necessary security standards for both variants.

1. Public Nodes: The objective is to establish connections with multiple public nodes. The AAA-me protocol can be configured to establish connections with multiple public networks. In doing so, the retrieved data of the nodes must be reconciled. In addition, one needs a periodic check to see if the nodes are still responsive. A local running consensus has to decide how to proceed in case of conflicting return values. The advantage here is that no blockchain administrators are needed to operate the nodes. The disadvantage is that tasks that the node can already do must be implemented manually. This includes consensus among the data states or managing the connection to several nodes.

2. Local Nodes: A local node is set up that can be trusted. All own interactions are executed via this node. This reduces the effort to implement an additional consensus or to manage the connections to different nodes. The administrative effort of maintenance remains in-house and access to the node should only take place via appropriate security measures, such as a key file and mutlisignature wallets for any transactions.

Neither of the two variants can be named as a clear recommendation for implementation. Rather, it must be looked at individually which resources are available. An example program for managing the connection to multiple nodes will be discussed later.

5.1.2 Frameworks

In this section, particular parts of AAA-me are picked out and described. These parts are essential for the realization and the understanding. A complete presentation of the code would be too extensive and therefore must be reduced and limited to program parts.

Django

As previously stated, Django is a software development platform designed for the implementation of web services. In this section, we will demonstrate the programming of two websites present in AAA-me and demonstrate how Django generates a representation from them.
First of all, we have the blockchain connections. These factors are crucial for the execution of interactions in a decentralized manner. In addition to performing AAA operations with aRADIUS server, other methods are available depending on the type of blockchain. In the figure F.1 (a) from the appendix F, one can see a general overview of the connections to blockchain nodes. It is sufficient to click on the corresponding tile to go to the details of the particular connection. Even though only one node instance is shown as an example in figure (a), multiple tiles are possible here. In order to show the display like this, the following three files have to be programmed in the backend: urls.py, views.py and models.py.
Let us commence with the database by utilizing the models.py file. This defines how objects should be stored in the database. Blockchain node connection information is stored in the class "Blockchain" described in the code5.2. This is where the data is saved so that it is safe to connect to the Node.
The owner of a blockchain model object is the person who created the object. This prevents other people in AAA-me from accessing settings to the Node who do not have administrator rights.

```
class Blockchain(models.Model):
   owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
   name = models.CharField(max_length=255)
   type = models.CharField(max_length=255, blank=True, null=True)
   consensus = models.CharField(max_length=255, blank=True, null=True)
   rpc_user = models.CharField(max_length=255, blank=True, null=True)
   rpc_password = models.CharField(max_length=2048, blank=True, null=True)
   link = models.CharField(max_length=255, blank=True, null=True)
   port = models.CharField(max_length=255, blank=True, null=True)
   genesis_file_name = models.CharField(max_length=255, blank=True, null=True)
```

Code 5.2: Blockchain Model Describtion from models.py

The views.py file contains data objects loaded from the database and rendered for the websites. For this, two of a total of four methods are shown in the code 5.3. In lines 11 to 21, a connection status check function is defined. Here, an excerpt was shown for only blockchain instances of type ETH. Other types are implemented by extending the if-else queries.
The method in lines one through eight takes a request from the user and a private key of the Blockchain object to call. This information is used to load the desired Blockchain object from the database in line 3. The auxiliary function already described for checking the connection status and another auxiliary function for the supplied private key are evaluated. In lines five to eight, the results are turned into the HTML template you want.

```
def blockchain_detail(request, pk):
   config_instance = get_object_or_404(Blockchain, pk=pk)
   blockchain_connection_status = blockchain_connection_details(pk=pk)
   genesis_file_status = check_genesis_files(pk=pk)
   return render(request, 'backend/config_templates/blockchain.html',
              context={'blockchain': config_instance,
                     'blockchain_connection_status': blockchain_connection_status,
                     'genesis_file_status': genesis_file_status, })


def blockchain_connection_details(pk):
   blockchain_data = Blockchain.objects.get(pk=pk)

   if str.casefold(blockchain_data.type) == "eth" or str.casefold(blockchain_data.
       ↪ type) == 'ethereum':
      try:
         blockchain_instance = blockchain_connector(pk)
         status = blockchain_instance.is_connected()
      except:
         status = False

   return status
```

Code 5.3: Blockchain Connection Methods from views.py

We still need to define the URL so that the web server knows which link the user can use to go to the page. These are defined in urls.py. An excerpt of this can be found in the code 5.4. The link assignments all follow the same pattern. The address with which the call should run is first defined. The addition of the angle brackets indicates a variable number. In line four, an integer is described in this manner, which embeds the private key of the corresponding database object.
In lines nine to 12, the URL for the representation of the blockchain objects is defined. For each of the operations of rendering, adding new objects, deleting from an object, and editing an object, a new URL is required.
Lines two to seven define URL, which denotes data objects from the database that represent the blockchain program. Currently, these have not been declared. There are a total of seven links defined, as in addition to the four previously shown for Blockchain objects, three additional links are required in this instance. There is a link defined in line seven that implements a data upload. This link lets you upload the information for smart contracts called ABI data. This helps AAA-me use the functions in it and check how things are going.

```
1 urlpatterns = [
2    path('blockchain_programs/', blockchain_programs_overview, name='
       ↪ blockchain_programs'),
3    path('blockchain_programs/new/', new_blockchain_program, name='
       ↪ new_blockchain_program'),
4    path('blockchain_programs/<int:pk>/', info_blockchain_program, name='
       ↪ info_blockchain_program'),
5    path('blockchain_programs/<int:pk>/edit', edit_blockchain_program, name='
       ↪ edit_blockchain_program'),
6    path('blockchain_programs/<int:pk>/remove', remove_blockchain_program, name='
       ↪ remove_blockchain_program'),
7    path('blockchain_programs/upload', upload_file, name='upload_file'),
8    path('config/', configs, name='config'),
    path('config/blockchain_<int:pk>/', blockchain_detail, name='blockchain_detail
        ↪ '),
    path('config/blockchain/new/', new_blockchain_config, name='
        ↪ new_blockchain_config'),
    path('config/blockchain_<int:pk>/edit', edit_blockchain_config, name='
        ↪ edit_blockchain_config'),
    path('config/blockchain_<int:pk>/remove', remove_blockchain_config, name='
        ↪ remove_blockchain_config'),
]
```

Code 5.4: URL Definition for Blockchain Connections and Blockchain Programms from urls.py

The blockchain programs require separate models and views. The database schema definitions described are designed to support different blockchain technologies. Fortunately, many projects are oriented to the Ethereum Virtual Machine (EVM) in order to be able to port the programs that run on ETH to their projects as well. With the inclusion of logic for programs on the EVM, most blockchain programs are usable. If other technologies deviate and an adaptation of the database models becomes necessary, this can be achieved by adding a new class to the models.py file while including the BlockchainModel class.
The BlockchainModel class is described in the code 5.5. A link to the user model as a foreign key is made possible. This allows users, also created by the RADIUS protocol, to be entered as the owner of a program. However, this does not replace the need for possession of the secret key to interact with the program on the blockchain. Thus, ownership mapping is only for settings in AAA-me.

```
1  class BlockchainPrograms(models.Model):
2      owner = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
3      smart_contract_owner = models.CharField(max_length=255)
4      smart_contract_owner_model = models.ForeignKey(Wallets, on_delete=models.
           ↪ CASCADE, blank=True, null=True)
5      address = models.CharField(max_length=45)
6      name = models.CharField(max_length=255, default='')
7      file_name = models.CharField(max_length=255, default='')
8      contract_class_name = models.CharField(max_length=255, default='')
9      version = models.CharField(max_length=55)
10     blockchain_type = models.CharField(max_length=255)
11     program_path = models.CharField(max_length=255, default='')
12     # Path to local directory, where blockchain programs lie.
13     # Route-Dir: ./project/blockchain_programs/
14     blockchain_instance = models.ForeignKey(Blockchain, on_delete=models.CASCADE)
15     commentary = models.CharField(max_length=1255)
```

Code 5.5: Model for Blockchain Programs

In line two, the ownership assignment is specified. Line four provides another link to a data model. This link is intended for future developments when wallet functionality will be fully provided by AAA-me. Currently, the wallet model only stores information necessary to control an address wallet on a node or third-party wallet software. Lines seven through eleven provide information about the classes in the program and the necessary files for local execution. The smart contracts on an ETH-based instance require, among other things, so-called ABI files. These must be available locally. The path to the data is defined in line eleven. Furthermore, information about the blockchain itself is saved in this model. Therefore, a link to the corresponding data model is also created here in line 13.

In the code excerpt code 5.6, a function is exemplified for the essential methods that regulate the users' interaction from AAA-me in the browser. They are defined in the views.py file. The method shown is used to make web pages where you can change the data of a Program-Data-Models object from the database. This option is allowed and only shown for the user defined in the AAA-me database. It is not possible for other users to view this object on this website.
The "edit_blockchain_program" method receives the request of the call from the user and the primary key contained in it. The helper function in line three checks if there's also an ABI file distributed on the AAA-me server. This helper method is provided via the blockchain.py file and is programmed in the blockchain.py file. Lines five through ten specify a list of all the configured blockchain technologies. For future selection options, this setting is built into the system so that drop-down selections can be used to select the appropriate technology.
Starting at line 12, a check is made to see how the edit site is accessed. If you want to create a new object, the call will be a POST call. The forms.py form templates will be loaded, and the user can fill in the data fields. The validity of this information will be assessed, and if it's found to be valid, a fresh object will be created containing this information. The user will then be redirected to the overview page.
The same form template as described above is used in line 21 to describe the loaded object and then render it with the return command.

```
 1  def edit_blockchain_program(request, pk):
 2      program_instance = get_object_or_404(BlockchainPrograms, pk=pk)
 3      program_code_available = check_blockchain_program_code(pk=pk)
 4  
 5      try:
 6          blockchains = [blockchain for blockchain in
 7                         Blockchain.objects.all()]
 8      except Exception as e:
 9          blockchains = []
10          print(e)
11  
12      if request.method == "POST":
13          form = BlockchainProgramForm(request.POST, instance=program_instance)
14          if form.is_valid():
15              program_instance = form.save(commit=False)
16              program_instance.owner = request.user
17              program_instance.blockchain_instance = blockchains[0]
18              program_instance.save()
19              return redirect('blockchain_programs')
20      else:
21          form = BlockchainProgramForm(instance=program_instance)
22      return render(request, 'backend/blockchain_programs/edit_blockchain_program.html
            ↪ ',
23                    context={'form': form,
24                             'program_code_available': program_code_available})
```

Code 5.6: View Function for Rendering Edit Page of Program

The following code 5.7 shows how a log-in on a website can be extended by the AAA-me component. In the case shown, AAA-me was installed locally on the web server. This also integrated RADIUS functionalities through FreeRADIUS.
However, the integration is more complex, but it is of great interest. In order to provide a more comprehen-

sive description of the integration with the code reference, view function blockchain authentication, and for reading convenience, this excerpt is longer.

```python
def remote_login_request(request):
    try:
        company = CompanyConfiguration.objects.first()
        blockchain_program = BlockchainPrograms.objects.filter(smart_contract_owner_model=company.wallet_main).first()

    except Exception as e:
        company = ""
    if request.method == ’POST’:
        form = RemoteLoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get(’username’)
            try:
                certificate = read_certificate(blockchain_program_model=blockchain_program, user_name=username, user_address=’’)
                if not certificate:
                    messages.error(request, "No certificate found")
                else:
                    user, _ = User.objects.update_or_create(
                        username=username,
                        defaults={
                            "username": certificate[0][4],
                            "company": certificate[0][2],
                            "email": certificate[0][4] + certificate[0][3],
                        },
                    )
                if user is not None:
                    user.backend = ’openwisp_users.backends.UsersAuthenticationBackend’
                    login(request, user)
                    messages.info(request, f"You are now logged in as {username}")
                    return redirect(’/’)
                else:
                    messages.error(request, "Invalid username.")

            except Exception as e:
                messages.error(request, f"Certificate Error: {e}")
        else:
            messages.error(request, "Invalid username.")
    else:
        form = RemoteLoginForm()
    return render(request=request,
                  template_name="frontend/login_remote.html",
                  context={"form": form})
```

Code 5.7: Login Function for Blockchain-Authentication Abilities from Views.py

The first line describes the name of the method that will be defined later in urls.py to retrieve the corresponding URLs. Also, the user's request is passed to this function. An attempt is made to read corresponding configurations from the web server's database at the beginning of the function. The database is described in

Django by model.py and accessed with the corresponding calls (lines two to seven). In these configurations, the corresponding information of the own company or the corresponding blockchain instance is entered, which is to be requested for reading out.
Starting at line eight, the request is taken and examined further. If the request is a “post-request”, the data it contains is loaded by the user into the appropriate form from forms.py. This form is a representation of a form that needs to be filled out for the user. This could be the login window asking for login information, but other forms, such as creating a forum entry, could also be possible.
Based on the provided data, an attempt is made to load the corresponding certificate or, depending on how the programs on the blockchain are developed, to load the blockchain program in which the users’ certificate is managed. Further data about the user is loaded from the blockchain accordingly (lines 17 to 24). If there is no certificate on the blockchain, the process will be terminated here. Otherwise, a potentially existing information record is either updated from the local database or a fresh one is created.
The local authentication process takes place on lines 25 to 32. The information provided is compared with the local backend and authenticated accordingly. For security reasons, one would still need to add a verification layer in order to be able to securely guarantee the ownership of the certificate. This could be a challenge-response strategy, where the server issues a random challenge and requires the user to sign it. The authenticity of the procedure can be guaranteed if the procedure is successful.
Subsequently and in the further `else`-conditions the representations and redirects are defined, which the user should see.
Finally, in lines 39 to 41, the rendering is described in which the data set is passed, and the corresponding template is loaded.

jQuery and AJAX for Progressive Web-Apps

The Python framework Django offers a multitude of functionalities, however, in order to implement a PWA, JavaScript remains a formidable obstacle. Fortunately, Django offers the possibility of integrating JavaScript code. To maximize the potential of a PWA and maximize its use in AAA-me, an interaction of AJAX and jQuery is employed to embed JavaScript into the website and to deploy a PWA.
In order to use JavaScript in your own Django project, the folder (in AAA-me in the frontend subproject) “static” must be set in the Settings.py.

```
STATIC_URL = "static/"

STATICFILES_DIRS = [
   BASE_DIR / "frontend/static",
]
```

A detailed description of how AJAX and jQuery integrate and work in Django projects can be found in the tutorial [217]. This excerpt stays close to the created program code with a basic explanation.

AJAX is an acronym for asynchronous JavaScript and XML programming languages. This technology combines several technologies to enable the transmission and reception of asynchronous data to the web server. This allows the website to be partially updated and does not require a full website reload. A display like a button in a form can therefore be adapted to the input and increases the user experience, since the whole website does not have to be reloaded.
The objective of the jQuery library is to facilitate the process of working with HTML documents. As a JavaScript library, it makes development easier because it allows you to deliver the same results with less code. Furthermore, AJAX code can be simplified and integrated through the use of jQuery. jQuery can include other features, such as result handling or animations, in addition to including AJAX interactions.

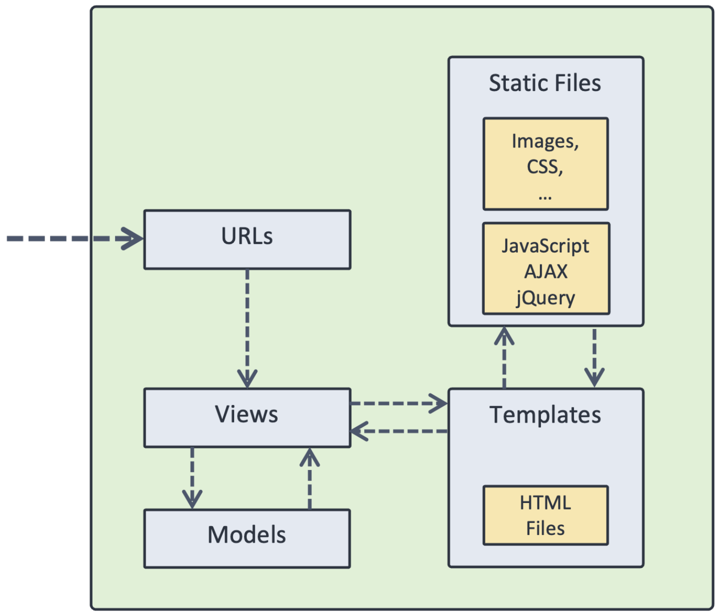


Figure 5.4: Django Architecture with PWA Support

The library encompasses a multitude of aspects of client site development, including but not limited to DOM manipulation, event handling, animations, and Ajax.
DOM is short for "Document Object Model." It is an API for HTML and XML document formats. It outlines the content of a document and allows programming languages to manipulate and interact with it. The structure of a document is represented in a tree-like format known as a DOM tree. Each element in a document, such as an HTML tag, is a node in this tree. DOM manipulation refers to the process of changing these elements or nodes. This can include adding, removing, or changing elements, changing attributes of elements, or changing the content of elements.

In the figure 5.4, the architecture of a Django project is shown, showing how JavaScript can be integrated. The leftmost request comes from a user, which first calls the corresponding view function in Django via URL. Depending on the programming, the view function will pick up database models. Lastly, the view function takes a template and loads the loaded data into it. The HTML template now has JavaScript programs that can refresh parts of the representation through AJAX, which is a fresh development. The templates are extended with jQuery features. One of the advantages is that data from the frontend can be exchanged with the backend in both directions.
So that data can be loaded quickly from the backend to the frontend, APIs are defined in Django, which can call the appropriate view functions. The frontend can then call these and implement interactions with the Django database.

For the actual JavaScript integration into Django HTML templates, they must be included at the appropriate place. The code snippet 5.8 shows what needs to be entered into Django's HTML template to set up the PWA functionality. The import of a separate JavaScript file is represented by each line. You can see the inclusion of jQuery in line one. This version is minimized, so the compressed version can be loaded and executed quickly.
Lines three through six import custom code sections that load the IndexedDB functionality into the Django templates. Using IndexedDB, one can store data in a web browser. In the test application, this was attempted with a secret key that could be loaded and controlled with wallet software.

```
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

   <script src="{% static 'frontend/js/indexedDB.js' %}"></script>
   <script src="{% static 'frontend/js/jquery.cookies.js' %}"></script>
   <script src="{% static 'frontend/js/accounts.js' %}"></script>
   <script src="{% static 'frontend/js/utils.js' %}"></script>
```

Code 5.8: Code Snippet for JavaScript Integration

The code excerpt 5.9, which creates a registration form within an HTML template, which should allow the user to sign up for an account. In line one, the base.html is loaded as the background for the purpose of realization. Only content is created in the inheriting template after all colors, fonts, and possible corporate designs are defined in the base.html. The content is defined in the {% block content %} areas and integrated into the base.html. In lines five to seven, a login form is therefore defined that allows the user to enter input. The form is implemented in Django. The type submits form triggers a POST request, so that the after successfully checked data can be stored.
In the {% block javascript %} indexed blocks JavaScript code is programmed directly into the template, or imported methods are executed. In the excerpt of the template shown, line 14 uses jQuery to execute a function after the template document has finished loading. The document.ready indicates this. In line 16, the sign-up form defined in {% block content %} is caught as soon as it is filled in and sent. Then, in lines 18 to 39, an AJAX request is created and sent. The AJAX method receives the data of the filled form (line 19), the request type (line 20) and the URL (line 21) which, however, was defined in Django notation. The script in this case pulls the corresponding URL defined in the urls.py.
The request is thoroughly examined and, in the event of a positive outcome, an entry is recorded in the IndexedDB database of the browser.

```
1  {% extends "frontend/base.html" %}
2  {% block content %}
3     <div class="container mt-5 w-50">
4        <form id="signupForm" method="POST">
5           {% csrf_token %}
6           {{ form.as_p }}
7           <input type="submit" name="signupSubmit" class="btn btn-success btn-lg"
               ↪ />
8        </form>
9     </div>
10 {% endblock %}
11
12 {% block javascript %}
13    <script>
14       $(document).ready(function () {
15          $('#signupForm').submit(function () {
16             // create an AJAX call
17             $.ajax({
18                data: $(this).serialize(), // get the form data
19                type: $(this).attr('method'), // GET or POST
20                url: "{% url 'validate_username' %}",
21                success: function (response) {
22                   add_to_db(0, response.wallet_address);
23                   if (response.is_taken == true) {
24                      $('#id_username').removeClass('is-valid').addClass('is-
                                ↪ invalid');
                            $('#id_username').after('<div class="invalid-feedback d-
                                ↪ block" id="usernameError">This username is not
                                ↪ available!</div>')
                        }
                        else {
                            $('#id_username').removeClass('is-invalid').addClass('is-
                                ↪ valid');
                            $('#usernameError').remove();
                        }
                    },
                    error: function (response) {
                        console.log(response.responseJSON.errors)
                    }
                });
                return false;
            });
        })
    </script>
{% endblock %}
```

Code 5.9: JavaScript for IndexedDB Application in frontend/templates/overview.html

How to initialize an IndexedDB has not yet been shown. Usually, IndexedDB is executed with JavaScript because this is a web API that only works in browser environments. Nonetheless, given the JavaScript integration options in Django as previously outlined, it is feasible to execute it in Django.
The code snippet 5.10 shows the snippet of a JavaScript code that creates an IndexedDB database and defines functions necessary for Create, Read, Uniform, Delete (CRUD) operations. The brief description of some methods was limited to a brief description of the interface, in order to keep the scope of the task within reach.
For the use of the IndexedDB, the following steps are usually necessary:

1. Open the database with the call indexedDB.open(). If the database does not exist yet, it will be created. The open method is an asynchronous process that returns a request. In line 11 this call can be found as an example. In the lines 12 and 13, the request is caught, and corresponding error actions are defined.

2. Setting up an object store, as a storage area for the database. Lines 17 to 27 show how to set up an object store. If the store does not already exist, a new one is set up. Otherwise, the existing one is updated.

3. Now CRUD operations can be programmed. Lines 42 to 46 and 58 to 66 describe a "create" operation.

```
1  const DB_VERSION = 1;
2  const DB_NAME = "accounts";
3
4  const openDatabase = function() {
5    return new Promise(function(resolve, reject) {
6      // check if IndexedDB is supported in browser
7      if (!self.indexedDB) {
8        reject("IndexedDB not supported");
9      }
10     // open IndexedDB
```

```
    const request = indexedDB.open(DB_NAME, DB_VERSION);
    request.onerror = function(event) {
      reject("Database error: " + event.target.error);
    };

    // add object store to database
    request.onupgradeneeded = function(event) {
      const db = event.target.result;
      const upgradeTransaction = event.target.transaction;
      let requestStore;
      if (!db.objectStoreNames.contains("post_account")) {
        requestStore = db.createObjectStore("post_account",
          { keyPath: "id" }
        );
      } else {
        requestStore = upgradeTransaction.objectStore("post_account");
      }

      // create an index object store with index type (idx_type being type)
      if (!requestStore.indexNames.contains("idx_private_key")) {
        requestStore.createIndex("idx_private_key", "id", { unique: true });
      }
    };

    request.onsuccess = function(event) {
      resolve(event.target.result);
    };
  });
};

// open an object store to add/delete objects
const openObjectStore = function(db, storeName, transactionMode) {
  return db
    .transaction(storeName, transactionMode)
    .objectStore(storeName);
};

// open object store and add object to it
const addToObjectStore = function(storeName, object) {};

// open object store to delete object from it
const deleteFromObjectStore = function(storeName, id) {};

// get requests from object store 'post_account'
const get_data = function (indexName, indexValue) {};

// add objects to IndexedDB and register sync event
function add_to_db(id, private_key) {
    // set up object that should be saved in the db
    const object = {
        id: id,
        private_key: private_key,
    };
    // add object to indexeddb object store
    addToObjectStore("post_account", object);
};
```

Code 5.10: Initialize and Usage of an indexedDB in JavaScript

Blockchain Interfaces (Nodes)

AAA-me connects to blockchain and DL- technologies by using the appropriate node software. These software nodes can be incorporated into AAA-me, as well as on external devices. Many of these Nodes provide secure access with authentication files to make a connection even over the Internet secure.
The decision of whether a node should operate on the same machine as AAA-me is subject to many considerations. Access to a local node is much faster and offers less potential for attacks. However, most current implemented functions and future planned methods only require read access to the node. It is therefore sufficient to describe access to an external node instead of implementing one in AAA-me.

WEB3.PY: Many feasibility studies use the ETH technology because of its great application potential. The connection is established with the library Web3.py [149]. The library is an open-source project that utilizes the methods developed by Geth as a model and implements them as a Python project. Geth is the node software that was used to create a private ETH blockchain. It can be used with any other client software. The naming conventions of the methods have been adapted to conform to the Python standards.
The initial read connection can be accomplished in a single line. The connection to the node and methods necessary for interaction were swapped out to the blockchain-py file and defined as a class in AAA-me.
Essential methods for interaction with the ETH instance are shown in the code 5.11 with python. The node used in the code was self-deployed to make sure it had its own instance. There were three other nodes that were connected to it. Read and write speeds are not significantly affected by the chosen number of nodes. However, it has an estimated impact on SPOF resistance.
Line one illustrates how an HTTP connection can be established with a node. The node must provide the appropriate interfaces in order to accomplish this. The second line describes how a smart contract instance is created based on an ABI file that must be loaded from local storage. With this, one can use the interface description and execute the functions described in the contract. We will see this in more detail in the Python method defined in line 20.
Line three takes a secret key and creates a local wallet, here meant as a key pair, for possible transaction creation. This instance must be used for signing a transaction for every interaction.
Lines five through 18 describe a method to enable an ordinary transaction, i.e., to send one's token. Here, the function is passed a Web3 instance, the account instance created from a secret key as the sender, and a public address as the sender. In lines seven to 14 the transaction is built. Line 14 could be used as input for data. We will encounter this again at a later point. Among the necessary information is this, which avoids attacks, such as a replay attack. Since signing must be done over the entire transaction, a minimal change to this data means that the signature must be recreated. This protection information is hidden in the nonce on line eight, which should not be confused with the nonce from the Proof-of-Work (PoW) in the block formation. The `w3.eth.getTransactionCount` method gets the total number of transactions that have been executed from the given address. The "pending" addition allows the nonce tracker to be incremented even if transactions have not yet been validated. This way, you don't have to wait until the one transaction has been validated before creating a new transaction. Line 16 takes the secret key to create a signature, and line 18 creates the signature and passes the transaction to the node for validation.

```
 1 w3 = Web3(Web3.HTTPProvider('http://185.193.67.50:9001'))
 2 contract = w3.eth.contract(address=bc.CONTRACT_ADDRESS, abi=bc.CONTRACT_ABI)
 3 account = w3.eth.account.from_key(bc.ACCOUNT_PRIVATE_KEY)
 4 
 5 def send_tx(w3, sender, reciever, amount):
 6    account = sender
 7    signed_txn = w3.eth.account.signTransaction(dict(
 8       nonce=w3.eth.getTransactionCount(account.address, 'pending'),
 9       chainId=107,
10       gasPrice=w3.eth.gasPrice,
11       gas=100000,
12       to=reciever,
13       value=amount,
14       data=b'',
15    ),
16       account.privateKey,
17    )
18    w3.eth.sendRawTransaction(signed_txn.rawTransaction)
19 
20 def authenticate_user_pap_contract_call(w3, sender, contract, username: str,
      ↪ password: str, wait_for_receipt=False):
21    password_bytes = password.encode()
22    password_padded = password_bytes.ljust(32, b'\0')
23    account = sender
24    txn = contract.functions.authenticateUserTx(username, password_padded).
         ↪ build_transaction(
25       {
26          'from': account.address,
27          'gasPrice': self.w3.eth.gas_price,
28          'nonce': self.w3.eth.get_transaction_count(account.address, 'pending')
29       }
30    )
31    signed_txn = w3.eth.account.sign_transaction(txn, account._private_key)
32    txn_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
33 
34    if wait_for_receipt==True:
35       txn_receipt = w3.eth.wait_for_transaction_receipt(txn_hash)
36       return txn_hash, txn_receipt
37    else:
38       return txn_hash, False
```

Code 5.11: Ethereum Interactions with Python

A method that executes a function from a smart contract is described in lines 20 to 38 The Smart Contract will be shown in detail later in chapter 5.18. There is a method that takes a password as a bytes32 data type and a username as a string. The name of the function is found in the code shown above at line 24. The required input variables are also listed there. The following parameters are required to create the transaction, including charges, the transaction number, and the senders public address. One can observe resemblances to the structure of a typical transaction. In lines 31 and 32, the transaction is signed and sent.

Depending on the decision to wait for a reply, the blockchain is questioned and verified. Regardless, the method outputs the transaction hash, which can also be understood as the ID of this.

BITCOIN-RPC: For the connection to a BTC node, the Python library bitcoin-rpc is used. Other libraries are also available in various languages. Since no smart contracts are inherently possible on BTC, only the possibility of exploiting the OP_RETURN command is presented.
Using Python to create a transaction on the BTC blockchain is shown in the code 5.12. The rough structure is reminiscent of an ordinary transaction on the ETH blockchain. Before a transaction can be sent to the Node, it must first be compiled and signed.

```python
from bitcoinrpc.authproxy import AuthServiceProxy, JSONRPCException
from binascii import hexlify

rpc_user = "<rpc_user>"
rpc_password = "<rpc_password>"
rpc_host = "<IP-Node>"
rpc_port = "<Port-Node>"

rpc_connection = AuthServiceProxy(f"http://{rpc_user}:{rpc_password}@{rpc_host}:{
    ↪ rpc_port}")

# Chose a UTXO to spend
listunspent = rpc_connection.listunspent()
txid = listunspent[0]['txid']
vout = listunspent[0]['vout']
scriptPubKey = listunspent[0]['scriptPubKey']
redeemScript = listunspent[0]['redeemScript']
amount = listunspent[0]['amount']

# create OP_RETURN
message = "test-Data"
message_hex = hexlify(message.encode()).decode()
op_return_output = {"data": message_hex}

# create change addresses
change_address = rpc_connection.getrawchangeaddress()
change_amount = round(amount - 0.0001, 8) # minus fee
change_output = {change_address: change_amount}

# create raw transaction
inputs = [{"txid": txid, "vout": vout}]
outputs = [op_return_output, change_output]
rawtx = rpc_connection.createrawtransaction(inputs, outputs)

# sign transaction
signed_rawtx = rpc_connection.signrawtransactionwithwallet(rawtx)

# send the transaction
txid = rpc_connection.sendrawtransaction(signed_rawtx['hex'])

print(f"Transaction send; TXID: {txid}")
```

Code 5.12: Bitcoin-RPC Usage with Python

The first two lines import the libraries that are needed. After setting up the node, the RPC information is used to connect to the node. This is done in lines four through nine. Now transactions must be accessed to prove ownership of the token, in this case the BTCs. These proofs of ownership are usually associated with transactions that grant you credit and whose addresses you control by holding the secret key. This happens in lines 12 to 17, where lines 15 and 16 define the bitcoin scripts (see also chapter 2.2.1 Transactions), which define the ownership of the tokens. The scriptPubKey tells the next owner of the UTXO (unspent transaction output) how to prove that he can spend this output. The recipient is usually required to provide a signature with the private key of the recipient's address in a standard bitcoin transaction.
It is specific to P2SH (Pay To Script Hash) transactions. The redeemScript is a program that tells you what needs to happen for the output to happen. The individual who intends to utilize the coins generates the redeemScript, and a hash of it is stored in the transaction that transmits the coins. The person receiving the coins must provide the original redeemScript that matches the hash in the transaction, as well as the information that meets the requirements of the redeemScript, in order to issue them.
Lines 20 to 22 are the parts of the transaction where you can write a message. Since BTC is meant to work with transactions, the `OP_RETURN` command is like a transaction with tokens that can't be spent. It means that the token sent with this transaction can't be used as an input for a new transaction. The sender has the option of adding additional credit to this transaction in addition to the charge. This credit is usually called a burned credit because the access is no longer tangible to the current state of the art.
The `OP_RETURN` command has a storage capacity of 80 bytes. In ASCII, this means that there are 80 characters. In other formats, this might mean fewer letters. Before creating and sending a transaction, it is always important to check if the message fits. Otherwise, you could pay fees for an invalid transaction. Here, we looked at the characteristics and possibilities of the act Blockchain and how a certificate for authentication could look like. This should proceed similarly to the ETH blockchain. In subsequent sections, we shall provide further details regarding the characteristics, limitations, and potentials of the technologies, as well as the possible appearance of a certificate pertaining to them. However, in short, a new certificate is issued by the issuer, which creates a transaction. In this particular transaction, additional information is entered through the `OP_RETURN`-command. It is important to include a time limit for the validity of the certificate and the appropriate authorization rules. If the holder wishes to confirm his certificate, he can do so with a challenge response against his possession of the secret key of the receiver address of the transaction.
In the code block of lines 25 to 27 a change address is created. To this, all credit of the inputs is transferred, so that this would not be paid to the address of the receiver and thus be burned.
Lines 30 to 32 now create the transaction, line 35 signs it, and in line 38 the signed transaction is sent to the linked node. Finally, the transaction ID is output to the console.

5.1.3 Blockchain Programms for AAA-me

The design of programs and transactions to implement AAA functionalities was discussed in great detail in chapter 4.4 Blockchain Transactions and Programs for Triple-A functions. This section will focus on some direct implementations.
You need to install and set up wallet software, or even nodes, to be able to interact with Decentralized Applications (dApps) or even smart contracts on ETH. Furthermore, there are also several essential considerations, such as the kind of data that may and may not be processed, and the manner in which code is initially stored indefinitely. In addition to the usual requirements, there are also specific requirements for blockchain nodes.

The first necessary step is the installation of a proxy contract to control the respective program instances. The data segregation paradigm is explained in chapter 4.3 Programming on Ledgers. As a reminder, there is a proxy contract that contains the addresses of the business logic and data storage instances.
You can find an example of this proxy in the code 5.13. The third line imports a library from OpenZeppelin

that allows for controlling the ownership of a contract [205]. Certain methods ought to be restricted to a predetermined group of collaborators, in order to establish a level of trust in a proxy connection. Starting at line five, the Contract is defined. There, one can also observe the indication of inheritance of the ownable library of OpenZeppelin. In line six, a list is defined that enables the assignment of a name as a string to a smart contract address. This list can be accessed by the functions described below. There is an addition to the "`setAddress` and `deleteAddress`" functions. This addition is a modifier function derived from the inherited Ownable class that permits execution solely for the owner of the smart contract. Furthermore, these two methods constitute a state modification to the list managed by the Contract and necessitate a transaction with charges to be executed. The "texttgetAddress" function, on the other hand, is a view method, and as such, it can be easily accessed without a transaction, resulting in a swift response.
If a user now wishes to access a smart contract, he must first look in the proxy for the instance, and then access its function area.

```
pragma solidity ^0.8.11;

import "@openzeppelin/contracts/access/Ownable.sol";

contract Proxy is Ownable {
   mapping (string => address) addresses;

   function setAddress(string memory _name, address _address) public onlyOwner {
      addresses[_name] = _address;
   }

   function getAddress(string memory _name) public view returns (address) {
      return addresses[_name];
   }

   function deleteAddress(string memory _name) public onlyOwner {
      addresses[_name] = address(0);
   }
}
```

Code 5.13: Implementation of a Proxy-Smart Contract

A smart contract is presented next, capable of delivering a certificate for authentication in Web of Trust (WoT) fashion. In this instance, an arbitrary issuer provides an attestation to a specific holder. A verifier can check this if he's confident in the issuer. This way, the chain of trust will be distributed, and every participant will have to interact to getting the trust in a certificate. It will be necessary to perform a first transaction with exchange data for later verification.
The smart contract in the code 5.14 represents such an implementation. In this instance, the issuer is the entity responsible for resolving the transaction to the blockchain. In order to accomplish this, it necessitates the inclusion of two pieces of information within the code. The wallet address of the individual who intends to utilize the certificate, as well as the expiration date. The date when it expires is written in Unix Time. It is important to note that these times are only ever adjusted in block intervals. Therefore, there may be short-term delays in the updates, depending on the consensus used in the ledger and the configurations.
In lines four to seven, the variables that are needed are defined. The issuer and holder wallet addresses, the expiration date, and the revocation state are some of the details that can be found in the list.
In line nine, an event that is triggered when a revocation occurs is defined. It is possible to compare an event

with a console output. This is carried out in each node and is accessible for examination by them. Lines 11 to 14 describe a modifier function that checks whether the call is from the person who issued it or the person who holds it. This function can be seen as an auxiliary function that can be used in other methods. The use of the modifier function will be found in the function descriptions of the `revokeCertificate` and `closeCert` functions by including the function name `onlyIssuerOrUser`.

```
 1 pragma solidity ^0.8.19;
 2 
 3 contract SimpleCertificate {
 4     address public issuerAddress;
 5     address public userAddress;
 6     uint256 public expiryDate;
 7     bool public isRevoked;
 8 
 9     event Revoked(address revokedBy);
10 
11     modifier onlyIssuerOrUser() {
12         require(msg.sender == issuerAddress || msg.sender == userAddress, "Not authorized");
13         _;
14     }
15 
16     constructor(address _userAddress, uint256 _expiryDate) {
17         issuerAddress = msg.sender;
18         userAddress = _userAddress;
19         expiryDate = _expiryDate;
20         isRevoked = false;
21     }
22 
23     function revokeCertificate() public onlyIssuerOrUser {
24         require(!isRevoked, "Certificate already revoked");
25         isRevoked = true;
26         emit Revoked(msg.sender);
27     }
28 
29     function isExpired() public view returns (bool) {
30         return block.timestamp > expiryDate;
31     }
32 
33     function isValid() public view returns (bool) {
34         return !isRevoked && !isExpired();
35     }
36 
37     function closeCert(address payable _recipient) public onlyIssuerOrUser {
38         selfdestruct(_recipient);
39     }
40 }
```

Code 5.14: Smart Contract of Certificate - Implementation of 4.7 (a) Class Diagram

The initial state that this program should enter when executing is described in lines 16 to 21. The address of the transaction is taken for initialization and stored as the issuer address. The holder's address is specified by the Issuer in the transaction and stored as `userAddress`. The initial transaction also incorporates the expiration date, which is stored in the variable `expiryDate`.
Ultimately, lines 23 to 40 define the procedures for revoking the certificate or to accessing the respective states. After the expiration date, the certificate can be marked as expired. It is possible for the certificate to be revoked by the person who issued it or the person who holds it. Therefore, both states must be questioned to make sure. So, since only one interaction with the blockchain is necessary, the method for this was created in lines 33 to 35, which queries both states.
The last function that is programmed is a so-called self-destruct. It is possible to set all memory utilized by the Contract to zero by utilizing this method. The freeing of the memory is rewarded with gas, so that one can use this then at other place. It is not possible to call or change the instance afterward. Such a method is important in order to reduce the permanently required memory requirements. Especially if a certificate has expired, one should consider such a release. If necessary, this should be done automatically.

The implementation of the class diagram 4.7 (b) differs from the already shown implementation of the class diagram 4.7 (a) by the code 5.14. This program aims to create a list of certificates, similar to a Certification Revocation List (CRL) in a Public Key Infrastructure (PKI) or even an Online Certificate Status Protocol (OCSP).
In the code 5.15, a way is shown how a list of certificates can be represented using solidity code. A trusted entity is responsible for managing the list into which certificates can be created. This reflects the hierarchical structure of a PKI for building a certificate trust chain.
In lines two through nine, the structure of a certificate is defined. This structure was created following the Request for Comments (RFC) 5280 [43] with simplifications. Here one could complete the compliance with the standard or adaptations for individual needs could be made.
Lines 11 to 14 define lists that allow access to certificates by ID, by username, or even by wallet address. Afterward, an event is programmed that outputs the confirmation of the creation of a certificate to the user.

```
1  contract UserCertificates is Ownable {
2      struct Certificate {
3          uint256 id;
4          address walletAddress;
5          string companyName;
6          string companySlug;
7          string userName;
8          uint256 expiryDate;
9      }
10
11     uint256 private certificateCounter;
12     mapping(uint256 => Certificate) public certificates;
13     mapping(address => uint256) walletAddressToCertificateId;
14     mapping(string => uint256[]) userNameToCertificateIds;
15
16     event CertificateIssued(...);
17
18     function issueCertificate(
19         address _walletAddress,
20         string memory _companyName,
21         string memory _companySlug,
        string memory _userName,
        uint256 _validityPeriod
    ) public onlyOwner {
        require(walletAddressToCertificateId[_walletAddress] == 0, "Wallet address already has a
            ↪ certificate.");
        require(_validityPeriod > 0, "Validity period must be greater than 0.");

        certificateCounter++;
        uint256 expiryDate = block.timestamp + _validityPeriod;

        certificates[certificateCounter] = Certificate({
            id: certificateCounter,
            walletAddress: _walletAddress,
            companyName: _companyName,
            companySlug: _companySlug,
            userName: _userName,
            expiryDate: expiryDate

        });

        walletAddressToCertificateId[_walletAddress] = certificateCounter;
        userNameToCertificateIds[_userName].push(certificateCounter);
        emit CertificateIssued(certificateCounter, _walletAddress, _companyName, _companySlug,
            ↪ _userName, expiryDate);
    }

    function revokeCertificate(uint256 _certificateId) public onlyOwner {
        Certificate storage certificate = certificates[_certificateId];
        require(certificate.id != 0, "Certificate not found.");

        delete walletAddressToCertificateId[certificate.walletAddress];
        delete certificates[_certificateId];
    }

    function getCertificateByWalletAddress(address _walletAddress) public view returns (Certificate
        ↪ memory) {
        uint256 certificateId = walletAddressToCertificateId[_walletAddress];
        require(certificateId != 0, "Certificate not found.");

        return certificates[certificateId];
    }

    function getCertificatesByUserName(string memory _userName) public view returns (Certificate[]
        ↪ memory) {...}
}
```

Code 5.15: Solidity Code for Managing List of Certificates

The function in lines 18 to 44 takes the data of a certificate according to the given structure and creates a new entry. Meta information is modified during the process, such as the ID, which serves as a counter for the certificates in the list. The method starts with a check to see if there is a certificate issued for the wallet address. When creating a certificate, it is important to create a new address for each interaction. This improves privacy and data protection by minimizing the use of analytics with correlated information.

In the `revokeCertificate` function, a certificate is removed from the list. This method allows for data-efficient management of certificates, as a certificate can be completely deleted upon revocation or after the validation period has expired. In contrast to the preceding smart contract, wherein each certificate denotes an instance, there may be discrepancies in the size of the transaction. As a result, procedures can end up being less costly or more costly.

The two remaining functions permit the selection of a certificate from the Contract's inventory of certificates. The wallet address or the username are used as search identifiers. The identification number is also feasible. However, it has been removed from this excerpt.

5.2 Experiments

In order to prove that the promises of a blockchain are kept, experiments are conducted. The objective of the study is to examine the resilience of blockchain-based systems against SPOF. One research question is to determine the possibility of resilience against SPOF. The research inquiry leads to a hypothesis: A blockchain-based AAA system is robust enough to handle a significant number of requests.

A demo setup is proposed to examine the tasks of a RADIUS system on each of three different setups to answer this question. There are two setups that are blockchain-backed and configured with AAA-me. The other setup is a freeRADIUS system that has been configured to operate on a roaming basis.

The networks are virtualized on an Apple Mac 14 from 2021, equipped with a M1 Max processor and 64 gigabytes of RAM. Parallels Desktop 18 for Mac, in its Pro Edition version 18.2.0, was utilized as the virtualization tool. Ubuntu Server 22.04.1 was installed as the guest operating system in the ARM64 variant. Each Virtual Machines (VMs) was given two CPU cores and four GB of RAM.

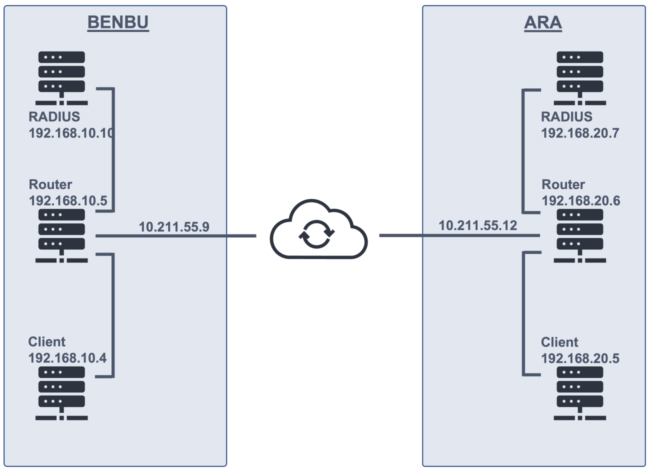


Figure 5.5: Virtual network architecture - On the left is the fictitious BENBU network, on the right is ARA. All IPs have been given by a DHCP Server.

A test network setup can be seen in the attachment G Setting up Virtual Machines with Virtual Networks. Two separate networks are set up, as explained in the appendix. Each network has three VMs with their tasks. The virtual networks are designed to be simpler so that the focus can be on the essential investigations instead. The configuration is identical for all three scenarios mentioned, namely, the scenario without blockchain and the two with blockchain. However, the scenario without blockchain is initially investigated to prevent any interference of settings or installation modules. The setup network can be observed in the graph of the virtual network architecture in figure 5.5. The freeRADIUS, AAA-me and blockchain nodes are each separately installed and provisioned for the network.

5.2.1 Experiment Setup

In these experiments, it is intended to simulate the transmission of client requests to a server using RADIUS. To accomplish this, Python scripts are created that resemble a request to the server. Three configurations are available, which allow one, two, or four clients to interact with the server simultaneously. Maintaining a connection is not examined. As a result, only one authentication request is made for each authentication request. In order to ensure comparability with blockchain setups, and avoid any inequality due to delays over internet connections or the like, a local node is created. The collapse of the RADIUS server is not anticipated in these experiments, as too few authentication attempts are expected to take place simultaneously. However, a delay in the authentication process should be identified in the data.
FreeRADIUS Password Authentication Protocol (PAP) authentications are investigated in the first part. Therefore, a possible negative influence of AAA-me should be proven or disproved. Secondly, an examination will be conducted on the interactions with the blockchain.In order to establish comparability, the categories and configurations remain largely the same.
There exist several limitations arising from the aforementioned test scenarios. Most of the research will be done on a MacBook in a local virtual environment. Over the Internet, execution does not occur. Furthermore, the configurations are not always adapted to real-world situations, but to be able to specifically investigate a behavior. In the case of DoS prevention, this means deliberately generating numerous requests simultaneously, without checking load levels or balancing the system. However, since both blockchain and non-blockchain investigations are structured the same, insights can be gained. These findings can be further verified or disproved through subsequent investigations.

FreeRADIUS Authentication with and without AAA-me

The time it takes to process a certain number of requests is being investigated. There are five categories of legitimate authentication requests: 10, 100, 1000, 10,000, and 100,000. Moreover, there are as many non-legitimate authentication or authorization requests, half of which use an incorrect user and the other half use an incorrect password. These requests relate to the essential (partial) aspects of an AAA system. Registration, authentication, and authorization of a user are these. Each as its own inquiry, with authorization, of course, requiring confirmation of identity. The call is implemented using a Python script that executes a typical RADIUS call via terminal. The authentication call can be seen in the code 5.16 Python File for Authentication Requests. The method receives a username and password combination, as well as an IP address and port. If the IP address and port are not entered, it is assumed that local authentication is taking place. The username and password information is subsequently compiled to the command line in lines 5 and 7, respectively, and executed.

```
1  def user_authenticate(username, password, roaming_ip, roaming_port, shared_secret=
       ↪ "testing123"):
2
3      # Prepare the RADTEST command:
4      if not roaming_ip and roaming_port:
5          radtest_cmd = ["radtest", username, password, "localhost", "0",
               ↪ shared_secret]
6      else:
7          radtest_cmd = ["radtest", username, password, roaming_ip, roaming_port,
               ↪ shared_secret]
8
9      # Setup Clock for time measures and execute radtest command:
10     start_time = time.time()
11     result = subprocess.run(radtest_cmd, capture_output=True, text=True)
12     end_time = time.time()
```

```

    # Calculating Time Consumption
    response_time = end_time - start_time

    # Return the result of the executed command
    return {"Username": username, "Response Output": result.stdout.strip(), "
        ↪ Response Time (s)": response_time}
```

Code 5.16: Python File for Authentication Requests

In order to expedite the implementation of all categories and maintain a realistic representation, the ThreadPoolExecutor library is utilized to execute multiple calls concurrently. It supports parallel execution under Python. You can see how the deployment works in the code snippet 5.17.
To gather enough information, scenarios are examined to represent the real world. The local authentications are the first thing to mention. Herein, we simulate a user, whether a student or employee, dialing into their designated home network. Another scenario involves a roaming connection. In this instance, the user directly transmits a remote address for an authentication request. This would be a useful use case for employees where administrators have established an authentication server for all locations, for example. The next scenario is about hopping from one network to another. This is a case where three RADIUS servers are interconnected in such a way that requests to server A are passed through server B to server C. This case is almost equivalent to the eduroam service for students (see 2.7.1 Application Example - eduroam).
The settings for roaming via FreeRADIUS can be found in the appendix H Setting up Roaming in RADIUS.

```
      with ThreadPoolExecutor() as executor:
         # Run run_radius_test Method for every test in parallel
         results = executor.map(user_authenticate, [user["username"] for user in
             ↪ user_password_list],
                            [user["password"] for user in user_password_list], [
                                ↪ roaming_ip] * len(user_password_list),
                            [roaming_port] * len(user_password_list), [shared_secret
                                ↪ ] * len(user_password_list))

         # Write Results in CSV-File
         for result in results:
            writer.writerow(result)
            summarized_time += result['Response Time (s)']
```

Code 5.17: Python File for Parallel Execution

In summary, the setups can be described as follows. Every experiment is run twice. With and without AAA-me installed next to RADIUS:

1. FreeRADIUS Local Authentications (10, 100, 1.000, 10.000, 100.000 Requests)
2. FreeRADIUS Roaming Authentications (10, 100, 1.000, 10.000, 100.000 Requests)
3. FreeRADIUS Roaming Authentications with 2 RADIUS Clients (10,100,1.000,10.000, 50.000, 100.000 Requests)
4. FreeRADIUS Roaming Authentications with 4 RADIUS Clients (10,100,1.000,10.000, 25.000, 100.000 Requests)

Blockchain Authentication with AAA-me and FreeRADIUS

Two distinct instances are utilized for the purposes of conducting experiments with the blockchain. Both instances use different consensus methods, but are based on the same technology. One instance is managed locally, and therefore falls under the private chain category. The PoW consensus is implemented and, as a result, it is subject to a similar behavior as the Bitcoin blockchain. This encompasses the randomness of block discovery times and the identification of a legitimate miner. The working title of this network is BBSE. The second instance is a PoA Ethereum chain from the Bloxberg consortium. PoA reduces the randomness of the time intervals and the decision of write permission is also fixed, at least to an epoch. This particular instance can be classified as a permissioned blockchain, as access is granted solely to a select group of research institutions. Nonetheless, read access is not dissimilar, as the data has already been processed by consensus in the blockchain. Only the connection to a node has a significant impact on the duration of the operation. All Nodes for BBSE have been installed on the VMs and synchronized with the network. It is only possible to set up a Bloxberg instance if the network has authorized it. As a result, Goehte-University has a functioning node, but it is incapable of establishing additional ones. Nonetheless, it should be noted that data access is also feasible through other nodes. Thus, this circumstance is not a drawback in comparison to the local BBSE instance.

The respective instances specifications display further differences. For example, the average interval time between blocks is 7 seconds on Bloxberg and 14 seconds on BBSE, respectively. For the gas limit, we have very close values of 10,000,000 (BBSE) and 8,126,464 (Bloxberg). Nonetheless, no transactions are generated for the experiments, as the registrations are not scrutinized. Because of this, these specifications do not have an impact on the results.

```
contract UserAuthentication {
   struct User {
      address walletAddress;
      string username;
      bytes32 passwordHash;
      bool registered;
   }

   mapping(address => User) private users;
   mapping(string => bool) private usernameExists;

   event UserRegistered(address walletAddress, string username);
   event UserAuthenticated(address walletAddress, string username);

   function registerUser(string memory _username, bytes32 _passwordHash) public {
      require(!usernameExists[_username], "Username already exists.");
      require(bytes(_username).length > 0, "Username cannot be empty.");

      users[msg.sender] = User({
         walletAddress: msg.sender,
         username: _username,
         passwordHash: _passwordHash,
         registered: true
      });
      usernameExists[_username] = true;
      emit UserRegistered(msg.sender, _username);
   }

```

```
    function authenticateUserTx(string memory _username, bytes32 _passwordHash)
        ↪ public returns (bool) {
        require(usernameExists[_username], "Username does not exist.");
        require(users[msg.sender].registered, "User not registered.");

        bool isAuthenticated = users[msg.sender].passwordHash == _passwordHash;
        if (isAuthenticated) {
            emit UserAuthenticated(msg.sender, _username);
        }
        return isAuthenticated;
    }

    function authenticateUser(string memory _username, bytes32 _passwordHash)
        ↪ public view returns (bool) {
        require(usernameExists[_username], "Username does not exist.");
        require(users[msg.sender].registered, "User not registered.");
        return users[msg.sender].passwordHash == _passwordHash;
    }
}
```

Code 5.18: Solidity example for a PAP Authentication

An initial authentication using the PAP method can be seen in the code 5.18. This is a realization of the PAP authentication. However, due to the transparency problem (2.2.6 The Transparency Problem), it is not possible to send unencrypted or non-hashed password to this smart contract in production. Therefore, this code was created for research and investigation purposes only.
This code will initially be tested directly with the test user categories outlined above for the experiments. Therefore, you have a total of 100,000 requests with correct data and another 100,000 requests with incorrect data. In the code, there are two distinct authentication methods. One method, however, triggers a transaction. The execution of this request will take at least the statistical block interval until the next block. In a crowded blockchain, this can be further delayed. Therefore, the execution times are expected to be high and do not require further investigation. The method defined with `public view` does not possess these execution times. Hence, it is utilized for the purposes of experiments. In order to ensure comparability, a node was created on the VM where the code is executed. This corresponds to the local FreeRADIUS tests from above. Later, another test run, with a node on the remote network, will be performed.
It is anticipated that the process of authentication for transactions will require a significant amount of time, as they must be validated by the blockchain before being processed. It is, however, worth investigating this feature further, as the advantages of authentication with transactions lie in the ability to integrate settlements. This method generates an event that confirms or denies the authentication. Therefore, at this particular moment, a settlement may commence. The openness of a distributed ledger makes this event a verifiable transaction for everyone.

5.3 Man-in-the-Middle Prevention

It is considerably more challenging to demonstrate defensive proficiency against Man-in-the-Middle (MITM) attacks. One could attempt to execute previously known attacks. If those attacks are possible in your system, this may be a certainty for missing MITM resistance. An inability to carry out these attacks cannot be considered as a general resistance. This does not eliminate the possibility of new attacks. Attempts, as in showing Denial of Service (DoS) attacks, can thus be made with difficulty. It remains a systematic evaluation. As we do, we'll investigate the possible attack paths for established attack types and defense strategies. Attack techniques and measures of defense were briefly discussed in 2.5 Man-in-the-Middle Attacks outlined. The architecture of the system can be seen from the high-level view 4.7.3 Architecture of AAA-me, as well as in the 5.1.1 Architektur.

For systematic evaluation, we consider the interfaces of a RADIUS system extended by AAA-me with the interfaces to the blockchain. The interfaces of the RADIUS system have already been considered in the 4.1 figure with respect to possible interfaces to the blockchain had. When extending the situation to AAA-me, the potential interfaces can be analyzed in the figure 5.6. There are several potential attack vectors for each of these interfaces. Furthermore, we want to examine other potential attack vectors. Here, the connections of entities with the blockchain are referred to. Each of these communication steps has the potential for an attack. Furthermore, there exist the potential hazards of a MITM attack within the blockchain system itself. As the nodes are required to communicate with each other, these paths also serve as a means of launching attacks.

1. First, let's begin with the first communication point. A classic connection is made between a device and a system, such as a laptop or cell phone. Since this is a classic setup, the known attack options and protective measures also apply here. It is possible for the attacker to impersonate a Network Access Server (NAS) and forward the request to a legitimate NAS. This would enable him to fully observe and potentially manipulate the traffic of the victim. These attacks would be of the types *False Base Station (FBS)*, but also *spoofing based*.

   One potential measure to mitigate this issue would be to implement static Address Resolution Protocol (ARP) entries, which would prevent the poisoning of ARPs and guarantee that IPs are associated with MAC addresses. Furthermore, it is imperative to guarantee the authenticity and integrity of the information through appropriate cryptography. This includes choosing a suitable authentication method (chapter 3.1 Authentication Methods). It could also help detect any attacks by intrusion detection systems at an early stage in order to be able to react to them appropriately.

1B. In the AAA-me setup, it would be possible for the user to communicate directly with the blockchain. For example, in order to create or manage a digital identity. The user would then transmit this information to the NAS, thereby enabling the system to execute authentication, authorization, and accounting. Consequently, there is communication from this device to at least one Blockchain Node. A potential adversary could establish a fictitious Node and present it as legitimate, as exemplified in the initial instance. Besides that, this attacker could also attempt to alter the routing information so that other wallets and nodes can connect to its node in control. Since the attack paths are similar to the first case described, the protective measures from that case also apply. A Resource Public Key Infrastructure (RPKI) would be a suitable protective measure against Border Gateway Protocol (BGP) spoofing.

2. Since the NAS must communicate with a RADIUS server, a potential attacker could take advantage of this interface. An ARP poisoning technique is possible here, whereby the attacker manipulates the ARP cache of the NAS and the RADIUS server to impersonate each other and intercept traffic. Additionally, routing information can be manipulated in two ways. The first approach targets BGP routing tables (network side) to route traffic through their systems. The second approach targets the routing information contained within the RADIUS server settings, which is the method by which roaming is implemented. In case the attacker can manipulate these settings or use the settings for himself, Radius will route the communication through him.

   The use of static ARP entries can help protect against ARP poisoning. Further measures include the use of certificates and the encryption of data traffic.

2B. In the application of FreeRADIUS it is possible to provide a RADIUS client and RADIUS server. The two solutions differ only in execution and configuration. Due to this similarity, this point is comparable to 3B.

3B. The RADIUS server communicates with the blockchain through AAA-me configuration. This connection may hold significant significance as, similar to an Oracle service, the interaction between blockchain and local database can have immediate implications on the data. Therefore, it is possible that the attacker would be able to create data and thus create a legitimate user who can act with sufficient authority in the system.

It is mandatory that a RADIUS server be connected to a node in order to receive and process the blockchain information. It is possible for this node to be outside its own system, or it could be integrated into the system. This implies that the attacker must attempt to exert indirect control over this node. He can, for instance, employ BGP spoofing to carry out a successful Eclipse attack. If the linking node is successfully disconnected from the overall system, the attacker can ascertain which new blocks are forwarded and which are withheld. Depending on the consensus, the attacker may be able to create new blocks of his own and send them to the linking node, thus allowing him to add a user to the database.

The most effective measures to counteract such a successful attack entail enhancing the number of node connections. An Eclipse attack is disproportionately harder to perform if the node connected to the radius server is connected to many other nodes. However, it is important to note that a wide distribution of nodes in disjunct networks is necessary. If all connected nodes were connected to the same network, attacks using BGP spoofing would again be possible.

3. If RADIUS servers are interconnected as a proxy to facilitate roaming, an attacker may exploit this circumstance to impersonate a legitimate server. The passwords arrive at the hostile server unencrypted and can be read if the authentication method is PAP. To accomplish this, the adversary may seek a server in the proxy settings that exhibits the smallest level of protection against attacks, thereby gaining access to all other networks.

In addition to these vulnerabilities, there exist the weak points mentioned in point 2, thereby providing the attacker with a substantial attack surface that must be constrained. It is imperative that the security measures for data traffic, as well as those for the server settings, are implemented with utmost care. The correct setting of the passwords for the hops prevents an attacker from successfully intercepting and reading an authentication request, even if PAP is set as the authentication method and the attacker has successfully redirected the routing (a BGP attack)

Furthermore, it is important to segment the network, i.e., that access through RADIUS allows only limited authority in the network, or that there is even physical separation.

4B. In the roaming setup, as explained in point 3, an attacker can find the server where the connection to the blockchain is least secure. The attacker would then attempt to create a user in order to gain authority on the desired network. These attacks are problematic because administrators do not have any measures of influence over the remote servers, making defense more difficult. However, this attack is only possible if RADIUS and the blockchain implementation of the protocol is implemented as a hybrid system and a blockchain identity is created without restriction on the user's own network.

Measures against such an attack would be to know the information on the connected blockchain nodes of the remote network and to be able to monitor them. This way, any isolation due to an Eclipse attack can be detected, and further impact can be contained. Also, the current update state can be read, and mitigation measures can be initiated in case of a potential compromise.

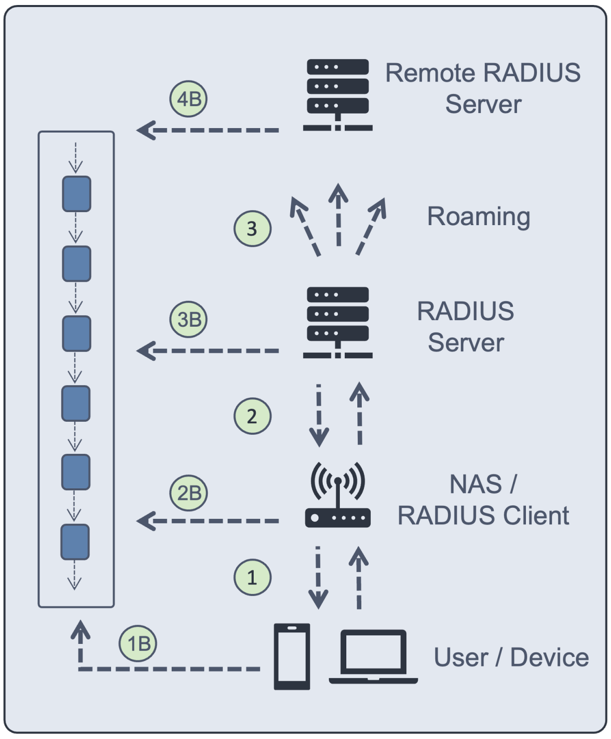


Figure 5.6: Interfaces of RADIUS and blockchain systems.

BC Even in the blockchain system itself, an attacker can try to partition the network through MITM attacks and thus manipulate the data exchange. It is still not feasible to generate a block in certain consensus methods, such as PoW. However, it's not the case in all the other procedures presented. The goal of the attack would be to create a record in order to gain access to other networks. Nonetheless, it is also feasible to implement a targeted suppression of individuals. For example, if partitioning is successful, the attacker could suppress blocks that contain a status change for a specific target person. This measure would prevent these modifications from reaching the intended network.

Protective measures are constantly implemented and adjusted in blockchain development. Such as the number of necessary connections to other peers or applied cryptographic systems.

Those vulnerabilities mentioned are considered to be promising. It is important to be aware that other types of attacks are also possible. Therefore, one should drive an all-encompassing security concept precisely because one does not know what the attacks will look like in the future.
Furthermore, it turned out that the connections to the blockchain nodes could be critical. A look at blockchain projects of public administrations showed that access to functionalities and data on the blockchain by a central service point increases the vulnerability of a MITM attack. Therefore, it is imperative to maintain the notion of distribution, or decentralization, and ensure that access is distributed. A client therefore cannot connect to only one node, or to only some nodes. There is a complete list of widely distributed nodes that can be queried simultaneously to ensure authenticity and integrity using a consensus process. A subsequent inquiry pertains to the implementation of connections to multiple data sources (nodes). Is there a procedure in place in case of inconsistency? How is the scalability of the system? Answers to this can be found in chapter 4.3.2 Blockchain Connection and for an implementation as a program in chapter 5.3.1 Simultaneous Node Connection can be found.

In general, measures against MITM attacks that need to be considered are:

1. Using more secure authentication methods (chapter 3.1 Authentication Methods). An appropriate

choice of authentication will make it more difficult for an attacker to read or manipulate a connection by implementing protective measures. In particular, by using certificates for client and server connections, this can greatly increase security.

2. Encrypted communication in general, through e.g., TLS. This makes it more difficult for the attacker to detect data traffic in general and manipulate it. The abovementioned point concerned only the data flow of the authentication essential information.

3. Use of authentication and certificates. These features make it more difficult for the attacker to impersonate a legitimate entity in the system. This is a crucial aspect that can be found in numerous protection extensions of protocols. For example, in the authentication methods based on Extensible Authentication Protocol (EAP). But also, DNSSEC uses signatures and certificates to exclude a potential MITM attacker. But there are more protocols like IPsec, S/MIME, SSH, and DTLS (Datagram Transport Layer Security). The list isn't exhaustive, but it highlights the importance of utilizing certificates, as well as the necessity of implementing and securing an appropriate PKI.

4. To secure the blockchain from MITM attacks, it is necessary to secure the access of the node. Currently, this is usually enabled by password files. Only the owner of these is allowed to dial into the node and make settings.

5. It is possible to hide nodes behind a TOR network, for example, and thus make access more difficult for the attacker, as routing information is difficult to keep track of, IPs are obfuscated and the owner is disguised. For example, a system administrator can manage multiple nodes in their system without any outside knowledge of which nodes they are.

6. A strong and widespread Peer-to-Peer (P2P) connection ensures wide distribution and, as a result, the need to manipulate all peer connections. The more widely a node is networked, the more difficult it is to mount a successful Eclipse attack by manipulating routing. In addition, redundancy ensures the loss of data and information in the event of a partial failure (possibly deliberate by the attacker).

Many options are provided with AAA-me for implementing these proposals. For example, the certificates on the blockchain can be used for distribution purposes. The procedures for verifying certificates themselves are provided by the used libraries in AAA-me. And thus in systems that have been configured with AAA-me. This increases the application potential of AAA-me beyond the AAA functionalities.

5.3.1 Simultaneous Node Connection

One way to protect against a MITM attack in the communication to the blockchain is to connect to multiple nodes simultaneously. In this particular instance, a server or client would not establish a connection solely to a single node of the blockchain. Multiple nodes would be connected, and data retrieval from all nodes would be matched. This has a significant impact on the success of the attack. However, such a solution also entails a number of drawbacks and obstacles that require resolution. The issues are mentioned again here as a reminder. In more detail, in chapter 4.3.2 Blockchain Connection covers the topic.

1. Consistency of Data (consensus)

2. Latency and Timeout Management

3. Load Balancing

4. Security Requirements (Authenticity and Integrity)

5. Error handling (consensus)

6. Scalability

It is possible to establish a connection to the blockchain using two methods. Either you use a so-called wallet software or library and connect to a node that is not part of your system, or you manage a node yourself in your system and access it with the wallet or library. Both systems possess their respective advantages and disadvantages. For security reasons, especially for protection against MITM attacks, it is important to never rely on just one node.
The blockchain technology already safeguards consistency and accessibility when you use your own self-managed nodes. However, there is an additional effort involved in managing and implementing the nodes in the system network.
To avoid the problems mentioned above, one must choose connections to nodes outside of ones own system. One potential approach, as previously mentioned, would entail utilizing a consensus procedure and referring to the data in a round-robin procedure through the trusting nodes.
In addition to the round-robin procedures, which have been implemented in PoA systems, for example, there are other consensus procedures that can address the points mentioned above. In the Hyperledger Indy project, for example, there is an exciting new method that is supposed to be very effective and can react dynamically to problematic data inputs. The so-called Redundant Byzantine Fault Tolerance (RBFT) method relies on continuous exchange of data blocks with signature validations of all participants. The data input is only considered valid when a certain threshold of positive signatures has been received.
Furthermore, there is also the possibility that the connections to the remote blockchain nodes are realized via own nodes. In this case, the nodes would address the aforementioned issues with the consensus realized by the blockchain. No separate consensus would be needed. However, the nodes would need to be managed and also distributed at different routing points for better protection.

In the subsequent code example, it is demonstrated how a connection to multiple nodes with checks can be established. The example code still doesn't have a penalty for a node if contradictory data can be gotten about it. This penalty can be implemented through a temporal deselection process, which will ensure that the node is trusted. This methodology was derived from the governance model employed at the Hyperledger Indy instance, as exemplified by Sovrin [249]. Again, trusted nodes are replaced by other nodes to penalize any delays or inconsistencies in the data.
In code 5.19 Node Connection Manager, a class has been set up to specify the allowed latency for node connections and a time interval for checking node conditions. Furthermore, a method was created that measures the time of a query to the node and compares it to the latency threshold. If the query takes too long, a false is returned. The method itself can be seen in lines 54 to 72. It is called in a helper method of the NodeManager class, which in turn is called for the periodic call functions.
Furthermore, a method is provided to check the consistency of the retrieved data. This method can be found in lines 47 to 51. Consensus procedures could be integrated into this method as to how to proceed in the event of inconsistency.
In the code example, only the client software Geth was considered. For the general implementation for different blockchain technologies, further client software would have to be considered for Ethereum, but also for other blockchain technologies such as Bitcoin, Hyperledger Indy or others.

```
12 class NodeManager:
13     def __int__(self, nodes: List[str], check_interval: int, latency_threshold: float)
         ↪ :
14         self.node_list = nodes
15         self.check_interval = check_interval
16         self.nodes = [Web3(HTTPProvider(url)) for url in nodes]
```

```
        self.active_nodes = [False] * len(node_list)
        self._start_health_check()
        self.latency_threshold = latency_threshold

    def _start_health_check(self):
        self._health_check_thread = threading.Thread(target=self._health_check_loop,
            ↪ daemon=True)
        self._health_check_thread.start()

    def _health_check_loop(self):
        while True:
            for i, node in enumerate(self.nodes):
                self.active_nodes[i] = self._is_node_active(node, self.latency_threshold
                    ↪ )
            time.sleep(self.check_interval)

    @staticmethod
    def _is_node_active(node: Web3, latency_threshold: float) -> bool:
        try:
            node_health = check_server_health(node, latency_threshold)
            return node_health
        except Exception as e:
            return False

    def check_data_consistency(self, data_getter):
        data = [data_getter(node) for node in self.nodes if self._is_node_active(node)
            ↪ ]
        return all(d == data[0] for d in data)

def check_server_health(node: Web3, latency_threshold: float) -> bool:
    latency = 0
    start_time = time.time()
    try:
        node.eth.block_number
    except Exception as e:
        latency = -1
    end_time = time.time()

    if latency != -1:
        latency = (end_time - start_time) * 1000
    if latency == -1:
        return False
    elif latency < latency_threshold:
        return True
    else:
        return False
```

Code 5.19: Node Connection Manager

5.4 Adoption in Hessian Central Office for Data Processing (HZD)

There are three ways in which AAA-me can be integrated into existing and new processes for Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD). And accordingly in processes that are necessary or sufficient for public administrations. A general overview of integration was given in the4.8.1 Adoption in Hessian Central Office for Data Processing (HZD). To reiterate, these three possibilities are integration of AAA-me with the FISBOX®, a realization of physical Single Sign-On (SSO), and the application of the RADIUS protocol to the Hessian WLAN.
Two of the three possibilities mentioned are to be studied only conceptually and depend on the results of the realization of the implementation of the RADIUS protocol on the blockchain by AAA-me. As a result, these instances will be investigated in the subsequent chapters. These two cases are the possibilities of using AAA-me in the Hessian WLAN and the implementation of a SSO on the physical layer.

The incorporation of AAA-me into the FISBOX remains to be described. This option offers many additional options of integration, as the FISBOX is already an integral part of various software solutions of the HZD for its customers. If an interface is provided here, it will automatically be available to future and current systems as well. The demonstration of the integration is illustrated by the case of the Smart Garden project. The project itself doesn't require a blockchain-based application. Nevertheless, this project is a great example of the integration process.
In the illustration depicted in the approach 4.9 on page 85, it can be observed that two distinct AAA-me integration points can be identified, which can be utilized for the Smart Garden project. To use blockchain reasonably, we used the example in 4.9 (a). This uses AAA-me for the configuration and monitoring of the blockchain data. Subsequently, in the event that monitoring is not required, AAA-me may be disabled and subsequently discontinued. The corresponding smart garden processes directly connect to the blockchain nodes themselves. This is where the potential of the Blockchain with its resilience against a SPOF first unfolds. The following considerations were made, which were significantly influenced by the fact that this is an application trial and all processes do not directly cost money or have to comply with security guidelines.

1. Blockchain Infrastructure Governance: Several projects run on the private blockchain instance that is assumed to be used. This provides complete control over the distribution of tokens to generate transactions. Furthermore, it is possible to disregard certain privacy concerns as access to this particular instance is restricted. It is possible to determine the utilization of the nodes and distribution. Consensus has also been established, and does not require probability voting. PoW was chosen to be able to investigate the probabilistic fluctuations in block generation and associated race conditions. Other existing consensus procedures are more planned and bring with them greater stability.

2. Kind of transaction and data content: Due to the possibility of issuing transactions at low cost, a separate transaction is issued for each piece of data information created. Scaling problems are not expected as a result.

3. Transaction or Program based: Both realizations were performed and compared.

The first case to be examined will emit a transaction as soon as a sensor has determined a data value. These data values are created by all sensors at the same time. As sensors typically possess limited computing capabilities, it falls to the control server of the sensors to generate a transaction to transmit the data to the Fisbox and the blockchain. In the program 5.20, a Python method is described that creates and sends an ETH transaction. In the web3 input, the interface to the corresponding ETH instance is defined.

```
 1 def send_tx(sender, reciever, amount, web3, data):
 2    account = web3.eth.account.from_key(sender)
 3 
 4    signed_txn = web3.eth.account.signTransaction(dict(
 5       nonce=web3.eth.getTransactionCount(account.address, 'pending'),
 6       chainId=107,
 7       gasPrice=web3.eth.gasPrice,
 8       gas=100000,
 9       to=reciever,
10       value=0,
11       data=data,
12    ),
13       account.privateKey,
14    )
15    web3.eth.sendRawTransaction(signed_txn.rawTransaction)
```

Code 5.20: Python Code for Creating a Ethereum Transaction

In line two, the secret key that is passed to the function is loaded into an account. A transaction is then created on lines four through 12. As a template, a JSON dictionary is used. The nonce describes a protection against replay attacks and is composed of a counter representing the total transactions made by the account used plus the transactions not yet validated but created. The "pending" note makes it possible to create multiple transactions without having to wait for validation per transaction each time. In the `gasPrice` the current charge rate is loaded and in the `gas` value the estimated number of gas comes along so that the transaction can be definitely validated. The `to` define the receiver address, which is actually not mandatory in the transaction-based system, but is needed for a legitimate transaction. Nonetheless, it is possible to reach a mutual agreement on an address to facilitate expeditious access to the data. The variable `value` allows transferring an actual token value. Therefore, it is set to 0 here because only data should be stored. And finally, the `data` field allows data to be stored in an encoded form.
Line 13 authenticates the entire transaction by signing it with the secret key, and line 15 transmits it to the connected blockchain node. It is now possible to use the method described in this way for a control-server to control the Smart Garden. Exemplarily, this control server is simulated by the code 5.21.

```
# Interfaces
url = '<URL-to-FISBOX>/SMARTGARD/FisBoxWebAPI/'
w3 = Web3(Web3.HTTPProvider('<URL-to-Node>'))

# JSON format
sensor_data[sensor_id] = {'Sensor_Name': sensor_name, 'Temperature': temperature}
jsonString = {'Sensor_Name': sensor_name, 'Temperature': temperature}

json_data = json.dumps(sensor_data)
temp = str(jsonString['Temperature']).replace(".", ",")
jsonString['Temperature'] = temp

headers = {'Content-type': 'application/json'}
response = requests.post(url + "api/smartguard/insert", data=json.dumps(
    ↪ jsonString), headers=headers)
ssl._create_default_https_context = ssl._create_unverified_context
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```

```
19
20 into_hex = w3.toHex(text=str(jsonString))
21 hash_hex = w3.keccak(hexstr=into_hex)
22 hash_hex2 = binascii.hexlify(hash_hex).decode()
23 input_text = w3.toText(text=into_hex) + w3.toText(text="00100") + w3.toText(text=
       ↪ hash_hex2)
24
25 send_tx(sender="81f57cba0babe06bb910a6a7a7a2075400ade489ea0cef043488061ef0c6e126"
       ↪ ,
26       reciever="0xe4CC5920f23E8d8726F48d05b8262be66a04FABd",
27       amount=rand_amount,
28       data=input_text,
29       web3=w3)
```

Code 5.21: Smart Garden Control-Server

First, connections to two interfaces, the Fisbox and the Blockchain Node, are defined. Lines six to eleven demonstrate the process of obtaining the relevant data from the sensors and storing it in JSON format. A call to the Fisbox that writes this JSON string to the database is generated in lines 13 to 18. Beginning at line 26, the transaction is created in order to store the record in the ledger. Lines 21 to 24 convert the string into a blockchain-ready data format.
The data on the Fisbox can now be verified against the blockchain-based information. Nevertheless, this approach is not in line with the requirements of data protection and must nonetheless be considered when implementing related initiatives.
If one wishes to access the data, it is imperative to refer all transactions to the designated "`reciever address`", verify the data, and store it in one's personal memories. As the data size increases, this process can become more complex and inefficient. Therefore, we are now exploring the program-based implementation. To accomplish this, we first deploy a blockchain-based application as a smart contract. For sensor data, this is exemplified by the code 5.22. In contrast to the transaction-based system, the data features have been enhanced as additional storage space has been made available.

```
pragma solidity ^0.8.11;

contract SensorData {
    struct SensorData {
        uint256 temperature;
        uint256 humidity;
        uint256 pressure;
        uint256 timestamp;
        bytes32 hash;
    }

    mapping(uint256 => SensorData[]) private _data;

    function addData(uint256 sensorId, uint256 temperature, uint256 humidity, uint256 pressure,
        ↪ uint256 timestamp, bytes32 hash) public {
        SensorData memory newSensorData = SensorData(temperature, humidity, pressure, timestamp, hash);
        _data[sensorId].push(newSensorData);
    }

    function getData(uint256 tokenId) public view returns (SensorData[] memory) {
```

```
20         return _data[tokenId];
21     }
22 
23     function getDataLength(uint256 tokenId) public view returns (uint256) {
24         return _data[tokenId].length;
25     }
26 }
```

Code 5.22: Smart Contract for Sensor Data by Smart Garden Project

In the smart contract 5.22, lines four to 10 describe the data points that a sensor provides. The hash value is formed over the sensor data and stored in the data structure to ensure data integrity.
Finally, line 12 defines the list of data that will be managed in the contract. A method for the storage of measuring points and two reading methods are also programmed in order to read out the data again and/or receive information over the total data set.
Finally, you can read the data using the following commands:

```python
contract = w3.eth.contract(address=contract_address, abi=contract_abi)
def get_data(contract, token_id):
    tx_hash = contract.functions.getData(token_id).call()
    return tx_hash
```

Code 5.23: Python Code for Retrieving Data from Smart Contract

In the code snippet 5.23, a contract class is first loaded to access its functions. Python knows what methods are available through the loaded ABI and how to execute them. Therefore, the described method does not require you to create a transaction. The method is an execution of the `getData` function from the smart contract. Nonetheless, this particular approach solely encapsulates the outcome of a singular data point. If you desire to possess additional data points or all of them simultaneously, it is possible to establish a getter method in accordance with the aforementioned criteria, thereby enabling this task.

The process of writing data points to the ledger again necessitates the execution of a transaction. This can be seen in the code 5.24 as an example. Since we have already discussed the process of creating a transaction to execute a method from the smart contract, we will only briefly outline it here.
First, the data needs to be converted to a different format. Then, a transaction is created with the required data and the rearranged data values, which is then verified by the secret key of the sender. Subsequently, the signed message is transmitted to a blockchain node.

```python
def store_data(web3, sender, contract, token_id: int, temperature: int, humidity:
    ↪ int,
            pressure: int, timestamp: int):
    data_hash = Web3.solidityKeccak(["uint256", "uint256", "uint256", "uint256"],
                                    [temperature, humidity, pressure, timestamp])
    # converts to bytes32
    hash_bytes32 = binascii.hexlify(data_hash).decode('utf-8')

    account = sender
    txn = contract.functions.addData(token_id, temperature, humidity,
                                     pressure, timestamp, hash_bytes32).buildTransaction(
        {
            'from': account.address,
            'gasPrice': web3.eth.gas_price,
            'nonce': web3.eth.get_transaction_count(account.address, 'pending')
        }
    )

    signed_txn = web3.eth.account.signTransaction(txn, account.privateKey)
    web3.eth.sendRawTransaction(signed_txn.rawTransaction)
```

Code 5.24: Python Code to Store Data in Sensor Smart Contract

All the necessary programs for the project were presented. The Smart Garden project can be controlled by transactions over the Blockchain with these. FISBOX was used as a data management solution in this case and extended with AAA-me to include blockchain functions.

5.5 Zero-Knowledge Proofs

The ZKP is currently being demonstrated using the Schnorr-Sigma method. Other libraries, such as Zokrates, could be utilized to facilitate further cryptographic algorithms. In chapter 2.1.3 Zero-Knowledge Proofs, additional information regarding the ZKP can be obtained.
The Schnorr-Sigma method can be seen in Python in the presented code 5.25. In the presentation, two methods are not depicted. One method is to extract a hexadecimal number from a hash value for better processing. Another method is used to generate the necessary key pairs for the process.
The two methods shown are required for the procedure. The method `schnorr_sigma_prover` in line one generates a proof. In order to accomplish this, it is imperative that the secret key and the desired information are conveyed to the method. This proof can then be checked against the public key and the source message with the `schnorr_sigma_verifier` method.

```
1  def schnorr_sigma_prover(params, secret_key, message):
2      p, q, g, y = params
3      k = random.randint(2, q - 1)
4      r = pow(g, k, p)
5      e = hash_to_int(str(r) + message) % q
6      s = (k - secret_key * e) % q
7      return e, s
8
9  def schnorr_sigma_verifier(params, public_key, proof, message):
10     p, q, g, y = params
11     e, s = proof
12     r_prime = (pow(g, s, p) * pow(public_key, e, p)) % p
13     e_prime = hash_to_int(str(r_prime) + message) % q
14     return e == e_prime
```

Code 5.25: Schnorr Sigma ZKP in Python

If the ZKP is implemented as shown, the data stored on the blockchain can be stored as ZKP with the matching public key. If a verifier wants to examine this information, they acquire it from the blockchain, acquire the original information from the user in the authorization request, and then examine it for accuracy. So, these steps would have to be added to the respective processes.

Chapter 6

Results and Discussion

This chapter focuses on the results of this thesis. This includes evaluating the experiments of AAA-me and the possible integration options that come from AAA-me's potential. The advantages and disadvantages of blockchain technology are also addressed. By utilizing pre-established metrics, a decision support is provided for each individual planned project to determine the feasibility of incorporating a blockchain. The results and research questions posed in the introduction will be summarized in this chapter.

6.1 Evaluation of Implementation

This chapter summarizes and discusses what has been achieved from the developments in chapter 5 Implementation for results that have been achieved. This brings up the issue of addressing some inquiry concerns from the introduction, such as avoiding a Single Point of Failure (SPOF). In summary, the experience of the developments is considered and weaknesses and strengths are present. The objective is to assess whether the use of AAA-me would be beneficial for public administrations.
In chapter 3.3 AAA-Technologies on Blockchains - Related Work, it is shown that scientific work already exists in the implementation of Authentication, Authorization and Accounting (AAA) methods on a blockchain. This is a first indication of the potential of blockchain technology. In the mentioned research papers, a complete transfer of AAA tasks on a blockchain is mainly discussed. Consequently, it can be concluded from these that an implementation is feasible. However, it is necessary to further investigate the reasonable implementation, as the cost from laboratory environments compares poorly with real-world applications.
The articles that describe a hybrid platform have emphasized a specific communication channel and the degree of integration. As a reminder from the chapter 4.1 Blockchain and RADIUS Interface Possibilities, there are varying degrees of expression for blockchain integration, as well as different points of integration. These topics are not examined in detail. This thesis therefore provides an overview of these interface possibilities, as a partial result.
AAA-me provides the possibility of using different parts of a Remote Authentication Dial-In User Service (RADIUS) system as an interface to the blockchain. Additionally, AAA-me supports the implementation of different degrees of blockchain application. Furthermore, AAA-me is designed as a configurator, as a result of previous failed projects. The reason for this is to eliminate potential SPOF resilience, by using a centralized point of interaction for a blockchain.

6.1.1 Can Blockchain prevent DoS Attacks?

A key part of SPOFs is the failure of a machine in the system. This section does not consider naturally occurring failures, such as hardware damage, power and Internet outages, or other external effects. Much more of a focus is a targeted shutdown caused by a Denial of Service (DoS) attack. Further limitations are made because another effective attack is performed internally to the system. When attempting to harm the network via erroneous transactions or other requests, one can select a node for removal. According to the strength of the botnet, it is also possible to disable other nodes. However, the size of a blockchain network allows the system as a whole to continue, so an effective DoS attack must be made through valid and high numbered transactions within the blockchain system. This would result in all nodes being overwhelmed with requests from other nodes, putting at risk the entire blockchain network. Usually, such attacks are intercepted by transaction fees. This makes it too expensive to cripple the entire network with valid transactions. However, if one is in smaller private networks or if the transaction fees have been reduced too much, the danger of DoS attacks can increase again.

Table 6.1: Results of time consumption of the experiment with FreeRADIUS setup in seconds.

Authentication FreeRADIUS	**10**	**100**	**1,000**	**10,000**	**100,000**
Local	2.11	17.76	173.27	1,741.47	17,253.20
Roaming	2.05	13.42	132.33	1,295.69	13,018.79
Roaming, 2 Clients 192.168.10.12	2.05	13.45	129.23	1,298.83	13,017.66
Roaming, 2 Clients 192.168.10.13	2.05	13.45	129.26	1,298.79	13,017.65
Roaming, 4 Clients 192.168.10.10	2.06	13.50	130.50	1,302.55	13,032.41
Roaming, 4 Clients 192.168.10.12	2.06	13.50	130.44	1,302.56	13,028.39
Roaming, 4 Clients 192.168.10.13	2.07	13.47	130.40	1,302.53	13,028.38
Roaming, 4 Clients 192.168.10.14	2.07	13.50	130.46	1,302.53	13,028.37

Does AAA-me with a blockchain configuration fulfill the promise of being more resilient to DoS attacks compared to FreeRADIUS? Can the promise be attributed to the support of blockchain technology?
In order to answer these questions, the experiments described in 5.2 Experiments, or more specifically in chapter 5.2.1 Experiment Setup, were executed and measured. A preliminary grouping of the test experiments was established in three roughly equal parts. For example, there are experiments designed to gain experience with a freshly installed Virtual Machine (VM) a FreeRADIUS instance. This involves running the code described in the above chapter 5.16 Python File for Authentication Requests to simulate a RADIUS client and forward authentication requests to the RADIUS server (our deployed FreeRADIUS instance). The first step involves setting up the client on the same machine as the server, then on a foreign network. Investigations of two and four radius clients simulated by Python code were performed for this purpose. The values for these investigations can be accessed in table 6.1.
There were also tests performed in the same categories as just mentioned, but this time a pre-release version of AAA-me was implemented. Nonetheless, the outcomes were akin to those obtained without the inclusion of AAA-me. Therefore, these will not be considered separately again. Nonetheless, a novel configuration sparked interest. Authentication requests will be investigated to see how proxy forwarding across a network affects them. The results can be found in table 6.2.
Finally, we are interested in implementing authentication requests over a blockchain. There are several concepts for this. The use of Password Authentication Protocol (PAP) was important for comparability, while ignoring important security aspects. The evaluations are recorded in table 6.3.

If we take a closer look at the values of table 6.1, we see that the number of (simulated) RADIUS clients has no effect on the total duration of processing all authentication requests. On the one hand, this may be due to the high-performance implementation of the RADIUS protocol by FreeRADIUS. However, the lab setup could also be improved in the settings to be able to cause the FreeRADIUS server to shut down due to overload. An authentication request took 0.53 seconds on average. With this processing speed, it would take numerous requests for the servers to be defeated. A closer look at the data indicates that denied authentication requests take a very significant amount of time, 1.02 seconds on average, longer than their legitimate counterparts at 0.02 seconds. Thus, targeted DoS attacks with faulty credentials would presumably reach the target more efficiently.

Table 6.2: Results of time consumption experiment with FreeRADIUS with AAA-me setup in seconds.

Authentication AAA-me	**10**	**100**	**1,000**	**10,000**	**100,000**
Local	2.05	13.43	129.22	1,302.63	12,925.54
Hop, 1 Client	4.00	26.49	255.06	2,561.45	25,508.22
Hop, 2 Clients 192.168.10.11	12.05	26.52	255.27	2,553.73	25,470.09
Hop, 2 Clients 192.168.10.21	4.07	26.50	255.19	2,555.81	25,480.89
Hop, 4 Clients 192.168.10.11	4.06	26.48	254.66	2,550.69	25,466.87
Hop, 4 Clients 192.168.10.21	4.11	26.56	255.16	2,549.37	25,466.87
Hop, 4 Clients 192.168.10.22	4.11	26.58	255.08	2,549.77	25,516.82
Hop, 4 Clients 192.168.10.23	4.14	26.59	255.13	2,546,47	25,516.94

The test results of the FreeRADIUS investigations with installed AAA-me instance can be seen in table 6.2. It is interesting to note that a setup was chosen that implements roaming with proxy forwarding. This means that a request to the RADIUS server must be forwarded to another RADIUS server. Here, the forwarding is also referred to as "hop". These hops bear resemblance to the principle of how RADIUS was implemented behind eduroam. However, it is shown here in a much more simplified form. In eduroam there are clearly several proxy settings and resulting hops (see chapter 2.7.1 Application Example - eduroam). In the test environment, only one hop was inspected.
The mentioned table shows, in the first column, the results of an investigation into the impact of AAA-me on FreeRADIUS. The values are identical to those found in table 6.1. Since other tests indicate the same values, it follows that AAA-me has no direct effect on FreeRADIUS instances performance.
If one examines the outcomes in the hops, it is evident that the times have significantly increased. Except for one measurement, the times are also largely independent of the number of simulated clients. The average duration utilized for all transactions is 1.02 seconds. If we examine the data more closely, as we previously did, it is evident that the accepted requests have remained roughly constant in terms of their duration. With average times of between 0.02 and 0.04, there are only a few outliers that demonstrate a longer duration. However, when a request was rejected, the negative case is significantly larger with an average of 2.02 seconds than in the setup without hops.

Table 6.3: Results of time consumption experiment with blockchain setup in seconds.

Authentication with Blockchain	**10**	**100**	**1,000**	**10,000**	**100,000**
Local Node, HTTP	0.09	0.91	9.37	91.81	1,011.09
Local Node, WS	0.08	0.76	15.82	100.38	1,144.39
Remote Node, LAN, HTTP	0.10	1.03	9.91	96.28	1,020.63
Remote Node, LAN, WS	0.08	0.78	15.30	88.58	1,226.26
Remote Node, Internet, HTTP	0.17	1.13	11.98	126.55	1,180.25
Remote Node, Internet, WS	0.26	2.31	19.54	89.92	681.03

In the last table, the time it takes to authenticate to a smart contract is shown. This smart contract was described in chapter 5.2.1 and implements PAP authentication. The transparency issue (2.2.6 The Transparency Problem) and the persistent storage of data, which would compromise security, prevent such a smart contract from being used in production systems. It remains an interesting scenario to investigate, as the mechanics of authentication are similar to the RADIUS setup.
First, one notices the highlighted fields in the table, which are light red. This is because of the connection to the blockchain node. There are two methods for connecting to it. One connection would be an HTTP connection, while the other would be a websocket connection. The websocket connection encountered issues with a high number of connection or authentication requests during repeated runs. And therefore, the results are highlighted red as an indicator for not comparable results to the other measurements. This results in strong deviations from the previous investigations and a lack of comparability.
For the setups, reference can be made to the chapter 5.2.1 Experiment Setup. We first compare the websocket connections with those through HTTP. The local setup has a tendency for faster execution with the HTTP. There is no tendency to be seen in the remote – LAN setup. It appears that there exists an alternate, faster access method in this instance. Even if the node is on a remote computer, the connection problem must be evaluated with the 10,000 and 100,000 user requests. Together with the measured times, the HTTP connection can clearly be evaluated as more efficient here. In summary, it can be said that the HTTP connections were more reliable and in many settings also better performing.
The measured times are significantly faster than the measured RADIUS set-up values. Therefore, requests with 100,000 users can be implemented faster by a factor of 25. This is a significant increase, which may not be fully compensated by settings on FreeRADIUS. The average processing time for all transactions is also significantly faster, at 0.05 seconds. With 0.05 and 0.06 average time for HTTP and websocket processing time, both cases are so close to each other on average that one can conclude that the result has no effect on the duration. In contrast to the measured time with the FreeRADIUS set-up.

To summarize, the outcomes can be categorized as follows. The blockchain implementation does indeed have the potential to prevent DoS attacks. This is a result of the reduced timeframe for authentication requests, as well as the possibility of replacing a failed node promptly. Especially in wide system RADIUS implementations with many hops, as found in eduroam, blockchain systems can significantly improve perfor-

mance. But where does this performance boost come from? And could it be improved even further?
First, the boost comes from a possible physical proximity of system-relevant information to one's own system, such as a RADIUS system itself. The data required for authentication does not require initial request and verification through a hierarchical and intertwining network. Many of the tasks associated with distributing and checking for correctness are handled by the locally operated blockchain node. This can reduce the load on the system, reduce the time it takes to obtain information, and consequently operate more efficiently.
Another aspect that makes the blockchain seem more performant here is the fact that long-lasting processes were not examined here. Authentication of a user requires registration to be carried out in advance. This can take a lot longer on a blockchain, especially if it is used with Proof-of-Work (PoW), than if it were done locally with a FreeRADIUS server. Many thousands of user registrations are put in time-dependent validation of the blocks because they require a transaction. A better scaling blockchain could reduce this limitation and is therefore one possibility for improvement. Another one can be seen at the consensus layer. The consensus algorithm PoW consumes a lot of energy and is therefore criticized. Changing the consensus algorithm to a more fitting system can eliminate this disadvantage.

6.1.2 Can blockchain prevent MitM attacks?

In chapter 5.3 Man-in-the-Middle Prevention, various points of attack for potential Man-in-the-Middle (MITM) attacks were examined. These have experienced an increase in magnitude as a result of the incorporation of a blockchain or Distributed Ledger Technology (DLT). But looking at current protection implementation against MITM attacks in networks, one may come to implement several of these methods on a blockchain. This is because countermeasures against MITM usually depend on a distribution of authority and on cryptographic certificates. Blockchain significantly supports both. If several measures to protect against MITM attacks are actually implemented on a blockchain, this technology can be considered as an infrastructure. Particularly if an increased number of other projects are also realized on a blockchain. This dissertation reveals the implementation of the RADIUS protocol on the blockchain. This alone could enable a far-reaching blockchain instance to be comprehended as an infrastructure, as the processes remain consistent even with diverse participants and can be executed on the same system. Should the Blockchain not be utilized to capacity by the RADIUS transactions, an extension of further functions can be considered. E.g., for certificates of a Public Key Infrastructure (PKI).
A blockchain alone cannot be considered the sole solution to MITM. It is itself too vulnerable to attack for that. If the distribution of the nodes is too small, for example. This case is assumed in private blockchain instances. This was demonstrated, for example, in the work of [89], that the distribution of nodes is essential for avoiding attacks on the Ethereum (ETH) blockchain. The same importance of node distribution is mentioned in the work of [250], in which a hard-to-detect partition attack is performed onBitcoin (BTC). Hence, a significant number of nodes is referred to as a countermeasure.
Nevertheless, a blockchain can be used to leverage the highly pronounced Peer-to-Peer (P2P) elements and raise a more MITM resistant environment. The transparency and immutability of the data are two characteristics that make an attack seem almost impossible, in addition to decentralization. Furthermore, the technology often incorporates cryptographic elements into its implementation, which can be embedded into the own systems. Due to this modularity, it appears that the implementation of blockchain could be a suitable option for preventing MITM attacks. Nevertheless, there are still numerous unanswered concerns that require confirmation in subsequent investigations.

6.1.3 Adoption in Hessian Central Office for Data Processing (HZD)

Until now, the work has shown that it is possible to port the RADIUS protocol through AAA-me on a blockchain. For now, economic and social factors aren't explored further. This observation leads to whether or how AAA-me or blockchain are suitable for the Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD).

Let us examine the scenario of Fisbox integration. Fisbox, a standardized, adaptable software-as-a-service (SaaS) solution for specialized information systems, offers a comprehensive interface for blockchain technology integration. We take advantage of the broad adoption within public administrations in the state of Hesse in order to achieve potential blockchain adoption. The implementations presented in the chapte5.4 Adoption in Hessian Central Office for Data Processing (HZD) demonstrate how the provided interfaces, not only from the Fisbox side, but also from the Blockchain, facilitate the integration process. The possibilities for implementation are numerous and often require the assistance of a blockchain engineer. At least be able to start a project on the Blockchain.

A transaction-based system was demonstrated, where data values are stored in the ledger by one transaction at a time. This provides the advantage of reducing the cost of transactions for data on the ledger. Although data sizes are limited, subsequent access to the data becomes increasingly complex. To analyze all the data given by a sensor, you must verify all transactions affecting a wallet and load the data accordingly.In contrast, a smart contract, as a program on the blockchain, exhibits superior capabilities in terms of data retrieval. Nonetheless, the initial deployment of a smart contract incurs significant expenses. Furthermore, there are more expensive transaction fees for a data update. Even if the used blockchain is a private one, the fee can be understood as a measure of how much a machine has to work to process the transaction. If the fee is high, then a lot of work will be processed. To put that into perspective, a transaction infused with information runs you around 22,200 gas. The deployment of the smart contract costs 332,000 gas, and a data update transaction costs around 130,000 gas. All numbers are from the smart garden project example and changes for new instances or other applications.

A general recommendation cannot be made here. The individual situation must be considered and evaluated. If the ledger is already heavily utilized, a program-based system may prove unsuitable.

In Hesse, a network of wireless network access points is currently being constructed, which will enable employees, as well as the public and guests, to access the Internet through the WLAN routers provided. Access is solely dependent on hardware that is capable of operating on WLAN. They must authenticate themselves via a captive portal when connecting to the network. By establishing a connection, the user acknowledges their agreement to the terms and conditions. The absence of full authentication and the requirement that all those granted access sign the agreements makes a RADIUS protocol seem excessive. Also, other areas of AAA scope are provided by RADIUS but not required here.

Since with FreeRADIUS, the RADIUS protocol implementation used by AAA-me, it is easily possible to set up a captive portal, a technical realization is possible and thus conceivable. Nevertheless, the idea remains that a technology is already used for the implementation and the use of RADIUS is not necessary at this point.

The last mentioned possibility for integration was exploring the application of RADIUS protocol by AAA-me for physical Single Sign-On (SSO) functionality. The possibilities of blockchain implementations through certificates were explained in great detail. These certificates allow passwordless authentication by storing the corresponding secret keys on hardware. During a challenge-response or Zero-Knowledge Proof (ZKP) procedure, the possession of this key is checked. In the ledger itself, additional links can be established through this process, thereby facilitating the possession or authorizations. Therefore, the hardware provided by a company can be assigned to the employee by certificates. If the employee is granted access, the hardware linked to him will also be granted access automatically. The integration possibilities of FreeRADIUS with existing user management systems, such as Active Directory or LDAP, allow for access on the physical level. AAA-me then retrieves information from the ledger and stores it in the appropriate systems.

Regarding public administrations, this could potentially enable the request for access to a building of another administration to simultaneously provide authorizations for the hardware to the Internet infrastructure. A WLAN network, similar to eduroam or Hessen-WLAN, could be used or a patch release for wired networks.For IT audits, such as those conducted by the BSI, this approach could be implemented quickly and efficiently.

To summarize, it can be asserted that the integration or implementation of blockchain technology is feasible in all three domains. The necessity of integration is, however, still to be investigated. More will be said about the importance of decentralization in later chapters. One reason may be given for a blockchain with the already seen potential of increased resilience to SPOFs. This was shown and discussed in chapter 6.1.1 Can Blockchain prevent DoS Attacks? and 6.1.2 Can blockchain prevent MitM attacks?. However, in addition to the potential for improvement, one also gets the potential for danger from the blockchain technology. These can include side effects that are hard to find today. These are highly dependent on the particular instance. There is a risk of overloading the ledger if one has a fully public system. Furthermore, the public development community has the potential to evolve in divergent directions from the required for public administrations. A set of measures to address the relevant threats is not only beneficial, but also imperative for ensuring the security of the data. The requirements of the respective projects must be clear in order to identify and build up the right instance. Having said that, the idea of a distributed ledger looks promising for public administration. Even though, a lot of time and effort is still needed to work on the system to combat the threats and make it safer.

6.1.4 Advantages and Disadvantages of using AAA-me

In chapter 2.7.1 RADIUS Protocol, the radius protocol, while in chapter 2.2 Blockchain Technologies, the blockchain was extensively discussed. Both systems possess their respective strengths and weaknesses, and by combining them, we aim to mitigate the shortcomings of the existing RADIUS systems. We also discovered a few advantages and drawbacks by attempting various experiments and evaluating the sources critically. For a comprehensive assessment of these outcomes, a SWOT (Strengths, Weaknesses, Opportunities, Threats) analysis is provided herein.

STRENGTHS:

- *Resilience to SPOF:* As described in chapter 6.1.1 Can Blockchain prevent DoS Attacks? and chapter 6.1.2 Can blockchain prevent MitM attacks?, AAA-me has the potential of increased resilience to SPOFs. This is essentially also accompanied by the two properties of decentralization and transparency.
- *Scalability:* Due to the inherent scalability concerns inherent in blockchains and DLTs, these technologies are not deemed to provide superior scaling capabilities. However, such a system can speed up the performance of some processes, as we discussed in 6.1.1 Can Blockchain prevent DoS Attacks?. For example, it took longer to check the information and verify it using multiple RADIUS protocol hops than if it was done using a blockchain. Increased scale can be achieved by saving time and resources by using AAA-me.
- *Verifiability:* Executive power can identify abuses through investigation, even though some pseudo anonymity applies because transparency makes interactions verifiable.

WEAKNESSES:

- *High Complexity:* Due to a hybrid architecture, the system is very complex and thus prone to errors. In addition, specialized personnel from different areas have to be acquired. Another issue is finding a suitable implementation or integration of AAA-me and blockchain for one's existing or non-existing RADIUS system. Also, future projects on blockchain infrastructure may lead to site effects that are hard to foresee.

– *Known Issues with Blockchain:* The scaling, transparency and Oracle issues are now taking hold on the RADIUS system. Possible measures and effects can be read in detail at corresponding places in chapter 2.2.6. Scaling is listed here as a disadvantage, along with strengths, since there are also AAA processes that can be slowed down by the blockchain. Contrary to what was written about scaling under Strength, this would negatively impact the performance of the system. One such problematic process could be the creation of a digital identity, such as a registration. This requires a transaction to be made on the blockchain, which could slow down the registration process if the workload is large. The need for storage is increased by the permanent storage of information in a ledger. Due to the chaining of the blocks, subsequent deletion is not easily possible, which results in a lack of deletability.

– *Expensive Infrastructure:* Blockchains are evaluated as expensive infrastructure. Public chains can have expensive fees if plenty of transactions have just been issued. A more private instance will require a lot of communication among the participants. This necessitates the conduct of discussions to achieve a consensus regarding the regulations of the DLTs. There are no clear process steps or definitions to identify the most appropriate system. In addition, there are many setting options, which further complicates the complexity of finding a suitable technology.

Opportunities:

– *Extension of AAA Methods:* In this thesis it was shown that by implementing a smart contract a new authentication method could be used with FreeRADIUS (see chapter 5.2.1 among others the code 5.18). Thus, PAP authentication was implemented using a smart contract. This authentication was subsequently translated for the FreeRADIUS-required system and adapted in accordance with the available data. This approach allows for the integration of other authentication methods.

  Furthermore, certificates could be used, which would allow an authentication with authorization rights to be completed. So, the user can perform a passwordless authentication, which increases their comfort and minimizes known problems with username and password authentication systems.

– *Protected Company Networks:* During a practical experiment, authentication could be performed in a closed corporate network. This required prior registration at an instance accessible via the public Internet. A smart contract was used to create a new identity on the blockchain. Subsequently, this identity could be authorized through an access authorization within the closed network. The user could prove the authenticity of himself by proving possession of the secret key, and the system could complete the authorization via the data in the smart contract. So, outside data could be integrated in a protected network by just opening one port for the blockchain node.

– *Action Log:* The data of a node is replicable, stored on it. If a node encounters failure or necessitates a re-setup, it retrieves historical data from other nodes within the system to ascertain its current status. Therefore, if a system needs to be reconfigured for authentication, it will also be able to obtain the current status from the system of nodes, the blockchain. The data contained within the ledger thus serves as an action log, wherein the corresponding status modifications are recorded. Upon failure, the data states are reproducible. It should be noted here that not all data can be recovered. This is attributed to the fact that data is stored in encrypted or obfuscated form as a result of the imperative measures taken to enhance privacy and data protection. If you lose the preimage of this data, you will not be able to retrieve it from the blockchain. Therefore, the data remains unrecoverable.

- *Blockchain Infrastructure:* Considering the blockchain instance as an infrastructure, it should be able to be used for other projects. This allows consideration of applications that could not be discussed before, when no infrastructure existed. Thus, a local PKI can be raised quickly and inexpensively that resides entirely on the blockchain.

THREATS:

- *Regulatory:* Systems that operate internationally often have difficulty adapting to the regulatory framework of different countries. Especially when regulators act diametrically different, complexity increases. Furthermore, the rules and laws change regularly, so that continuous checks for suitability and compliance must be implemented. Therefore, procedures should be in place in the event of non-compliance, in order to comprehend the consequences a suspension could have on a selected group of users.

  However, in addition to the regulation of the user countries, the regulation of the blockchain system could also be a threat. The processes initiated by a so-called "Black Swan" event, which leaves us unprepared, require swift intervention to stop. Having a quick response leaves no room for weighing all factors. In consequence, decisions could be made that could be considered detrimental to the overall system, perhaps only in the near or distant future. As an example, a prompt decision was imperative during the Decentralized Autonomous Organization (DAO) hack of the ETH Blockchain in 2017. The response was a hard fork that allowed the stolen tokens to be recovered from a smart contract. This approach is currently the subject of much discussion, especially from the point of view of the tamper-resistance of DLTs.
- *Acceptability:* A general acceptance of the technology cannot be found. In addition, previous failed projects have tended not to identify blockchain technology as a target-oriented technology for public administrations. This makes further research and testing projects by and with public administrations more difficult. But even detached from administrations, the general trend is often dependent on current events. If an exchange has gone bankrupt, or legal proceedings have been taken up against a blockchain project, the perception is increasingly subdued. Conversely, in boom phases, there is a desire to start a research project after all. The perception and acceptance of users can also often be observed to be dependent on current events. In this respect, general acceptance can be a major risk that is dependent on external events.
- *Blockchain-Bug:* There are bugs that appear at a later time in every software. In order to address these issues, updates are usually created and rolled out. In blockchain projects, these processes take a lot longer than in other projects. This is due to the decentralization and distribution of operators, which means that discussions can take a long time about what and how to build an update. Furthermore, each node operator is additionally provided with the option to persist in rejecting the update subsequent to its completion. In fact, an update is an opt-in, thereby requiring the node operator to only act in the event of a desired update. Blockchain technology can also be impacted by bugs or errors in smart contracts. These issues can become serious, particularly in the event that the data segregation paradigm has not been implemented, and business logic is accessed without a proxy. However, even with implemented paradigms, updates are expensive and require transactions. The biggest problem with errors is the undetected continuation of them. By connecting AAA-me of blockchain with FreeRADIUS into network systems, this vulnerability can form an attack point to penetrate a system.

It is important to note that these points only provide a rough overview of the respective properties. The results have emerged from the research for this dissertation and have been put together. It was tried to summarize some points to a superordinate and to explain further as a short bullet point.

6.2 Concealing Information on Blockchain

What kind of information can be stored on a distributed ledger? Since a blockchain is immutable, it follows that simply storing Personally Identifiable Informations (PIIs) must be excluded. But how does it work when this data is processed in advance?
The past has demonstrated through the disclosure of the activities of intelligence agencies that one approach is to store substantial amounts of data today for eventual decryption by superior hardware or decryption methods. This is referred to "Harvest Now, Decrypt Later (HNDL)". [133, 183] Furthermore, it is important to continuously adapt known and established procedures, such as the Rivest–Shamir–Adleman (RSA), in order to avoid the risk of effective decryption. As a result, the selection criteria for the keys are tightened, or other "formatting" is applied.
Knowing this, it would be irresponsible to actually deposit PIIs into the immutable blockchain, as they would be merely deposited for third parties. Decrypting the data later, by more efficient methods or hardware, could no longer be prevented. It is therefore canon in the blockchain scene that PIIs must never be stored in the blockchain, even if it is encrypted. But now there is a conflict that this information is needed for various projects. Also, or especially, smart contracts need this information for the realization of their possibilities.
One alternative approach to furnishing services with information without jeopardizing the security of the data is through the utilization of Oracle services (see chapter 2.2.6 The Oracle Problem). These can provide data from outside into the blockchain on demand. There are two methods to ensure that unencrypted PIIs does not inadvertently enter the blockchain in this step.

1. Cryptographic Hash Functions (2.1.2 Cryptografic Hash Functions)
2. Zero-Knowledge Proofs (ZKPs) (2.1.3 Zero-Knowledge Proofs)

An important property of cryptographic hash functions is collision resistance. This means that no two different values x, y can be found that produce the same hash value ($x, y : H(x) = H(y)$). A strong and a weak resistance can be distinguished. However, this distinction is not relevant here. More about this in chapter 2.1.2. Most cryptographic hash functions are surjective functions. Accordingly, the input area is larger than the image area and thus there must be collisions. This might lead to the assumption that if the hash function is broken, the preimage x is not necessarily found, but some x' that produces the same image ($H(x) = H(x')$). However, the reality is different and in the work [65] it could be shown that even for widely used and established cryptographic hash functions the direct original image can be found, or the function is considered broken. There are no immediate methods for breaking cryptographic hash functions if a method is considered broken. Many of the theoretical attack methods require large computations or some partial knowledge. Nevertheless, caution should be exercised and conclusions drawn for further development so that no great damage can be done.
The second method is the ZKPs. In these methods, hash values are also used, but they are processed in further cryptographic procedures. This further reduces the risk of information being discovered unintentionally. This can also explain and understand a certain trend towards research in this area. We saw an application of the Schnorr-Sigma method, in which a proof was created from a secret key and information. The verification was done with the public key and according to the information. The data is further processed by employing the discrete logarithm, which can lessen the risk of a failure of the hash function.

In summary and as a conclusion, it can be said that there are methods to store data on the blockchain. As things stand today, these methods are also secure. By using multiple methods, one additionally reduces the risk of decryption by breaking a single method. However, there is still a residual risk, albeit a minimal one. If one plans to regularly store PIIs in a blockchain, one should look at other aspects. for example, a restricted readable blockchain could reduce the significance of the risk. However, such a blockchain also directly reduces the possibilities of use. Another potential aspect could be the provision of the capability to eliminate data from the blockchain. If information were stored in parts of the transactions that could be made removable by pruning or other deletion mechanisms, it would be possible to delete information again afterward, even if it was only delayed. The knowledge gained in the field of cryptography has shown that a cryptographic technique isn't deemed broken overnight. Rather, through ongoing research, successive methods for breaking, but also possibilities for preserving the procedures are found. If old data were made erasable, this would make it possible to store cryptographically modified PII.

6.3 The (De-) Centrality Issue of Blockchains

The most widely used characteristic of blockchain technologies is their decentralization. Decentralization is used to distinguish blockchain technologies from classical database systems. It is important, but it is also often very controversial and ambiguous. Furthermore, it is difficult to answer what exactly decentralization is or means, or what its implications are.
Many projects rely on the intangible circumstances to sell their own, sometimes dubious, ideas. Particularly in comparison to BTC, which is plagued by a scaling issue. Attempts are being made to bring more efficient blockchain technologies to market, like Solana, Cardano, or IOTA [58, 240, 41, 150]. However, as discussed in 2.2.1 Bitcoin Blockchain, these projects are also subject to the CAP theorem (Figure 2.1). This implies that better scaling will lead to a lower level of security or reduced decentralization. For example, in the Solana and BNB Blockchain projects, a hack occurred where the entire blockchain was stopped by one instance in order to limit the damage [72, 198]. For Solana, the blockchain has already had to be halted several times [12]. Also ETH and BTC had to improve problems by forks in the past (see chapter 2.2.1 Bitcoin Blockchain and 2.2.3 Ethereum and Blockchains 2.0). Beyond the blockchain projects mentioned, there are many others that should be mentioned. The focus here will be on clarifying the term, not on providing comprehensive project descriptions. I will explain the meaning of decentralization using blockchain implementations and instances, such as BTC or ETH. This is since, despite the ambiguity of the definition, the term holds significant value in assessing the significance of blockchain technologies. Consequently, this explanation will be further used in this work.

One comes into contact with the concept of *distribution* if one searches for a meaning or definition of the decentralization. Even though the two terms can be distinguished, there is a tendency for confusion to arise in this context. One possible cause of this discomfort could be the existence of two distinct definitions. One definition comes from the year 1964 by Paul Baran [15]. One particular figure by him to illustrate the concepts of centralization, distribution, and decentralization is taken up in many instances in the literature (see figure 6.1 Centralized, Decentralized and Distributed Networks by Baran on page 139). A second definition is offered in response to a question on the Ethereum Stack Exchange forum, which unfortunately describes the definition of distributed and decentralized in complete opposition to Baran. This is also noted negatively by Vitalik Buterin in his description of decentralization [55]. Compared to the image 6.1, the response presented here has created a similar graph and swapped the labels for decentralized and distributed. Consequently, we discover a variety of definitional approaches to the aforementioned facts.
There is a word description available at Digitales Wörterbuch Deutscher Sprache (DWDS). The term 'central' here refers to a central point or a midpoint [87]. In Latin, the prefix "de-" means away. Hence, the term "decentral" denotes the distance from the center, as exemplified by the citation [88]. In the context of politics, the term is often described. A democratic system that possesses a separation of powers is an illustration of a decentralized system. Power structures are distributed among three main branches: the legislative, executive, and judicial branch. In contrast, monarchical or autocratic systems are established by a singular individual or a small group of individuals who amalgamate all authority. In addition to the separation of powers, there is also the possibility of establishing federal systems. The government of a country is splintered into numer-

ous local authorities. If the governments have a separation of powers, it is also done on the federated states. The depicted separation of powers exemplifies the process of decentralization of power at the political level. A further separation of powers should be possible with reference to blockchain technologies such as BTC. The current management of the money supply, which is currently performed by non-democratically elected institutions such as central banks, could be reorganized in this manner. With Bitcoin in particular, it should be possible to prevent people from intervening in monetary policy and thus being exposed to corruption or fraudulent manipulation. Politicians, of course, would continue to exercise control over the funds at their disposal. Artificial expansion, which is seen as one cause of financial problems, would no longer be feasible. However, there are also projects in IT besides the blockchain that can be taken as an example of the concept of decentralization. In Germany, the development of contact tracing software for the Corona pandemic raised the issue of data processing. Should this be done centrally at a government agency, or decentrally? For data protection reasons, a decentralized approach was quickly established. Although a central server entity is involved, it could not access the personal information [4].

The term distribution is a synonym for decentralization and more often describes the physical description of the situation [251, 252]. In contrast, political power is an abstract concept, where the term decentralization is preferred. These descriptions of the two terms are also found in the technological context. Although the computers of large corporations, such as Google, Apple, Facebook, Amazon, Microsoft (GAFAM), necessitate international distribution for efficiency purposes, their services are primarily subject to the terms of use of the respective company. To limit the power of respective companies on political level, the companies must adapt to the jurisdictions of the countries. This was the case, for example, with Microsoft when the EU stipulated that the browser constraint be dissolved and alternatives allowed [17].

Unfortunately, a lack of definition increases the irritation about decentralization. The use of this term in different fields has made it difficult to define with clear demarcations, which makes it difficult to understand. As a consequence, it is inevitable that erroneous and abusive applications will be made. Furthermore, the perspective can exert an impact on one's comprehension. It is important to be aware of the inflationary usage of these terms in DLTs, when you see the many projects that argue for decentralization. However, frequent shutdowns or even censorship of transactions demonstrated that the decentralized nature of the system should be questioned. As can be well observed with the shutdowns from the Solana network [12]. There are also many other failed projects that want to use decentralization to implement their projects. Among them are Terra Luna, Celsius or FTX. [190, 62, 74]

The mentioned examples promise positive aspects from decentralization, as well as BTC and ETH. But one must also look at the negative aspects. One of these aspects can be directly deduced from the CAP theorem. Decentralized systems induce scaling or security costs (see also CAP Theorem graphic 2.1 Representation of CAP Theorem with Blockchain Trilemma). Thus, excessive decentralization could result in a system that is too slow or that places a significant strain on its security. I would like to remind you that security here means the availability of the overall system in case of a failure of isolated nodes.

Blockchain technologies are subject to frequent debates regarding their scalability, especially when it comes to their adoption capabilities. Current solutions envision side-by-side systems (e.g., sharding [104] or parachains [248]) or accumulation of transactions to reduce the number of on-chain transactions (lightning [208], roll-ups [255]). In all cases known so far, the CAP theorem kicks in, and solutions reduce the security or decentralization property. This can be seen in the need for bilateral communication in accumulations of transactions, as well as the significantly increased complexity and inherent security issues. A more profound understanding of scaling solutions and their implications in relation to the CAP theorem can be found in 2.2.4 Solutions for Scalability Problem on page 25.

Even positive characteristics, such as censorship resistance, can be viewed critically. Text input in a transaction could already be immortalized with illegal material. Due to the lack of deletion functions, information thus remains in the ledger. For example, in the case that links to websites with child pornography were writ-

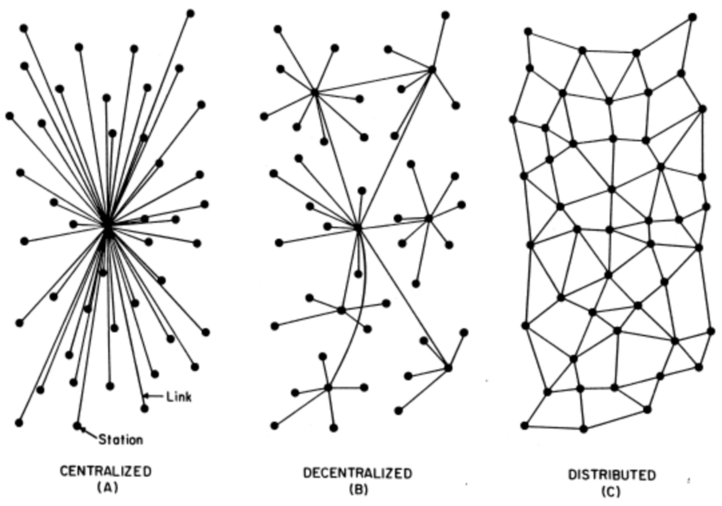


Figure 6.1: Centralized, Decentralized and Distributed Networks by Baran [15]

ten into the transactions [138]. Other data, such as threatening or abusive content, would not be able to be removed or mitigated.
Furthermore, in addition to the actual data on a decentralized system such as a blockchain, the implementation of update procedures is also more challenging. It is imperative to establish a coordinated approach for planning and execution with all relevant stakeholders. Due to the influx of meetings, which present the potential for conflict, it is possible to delay updates.
The separation of powers is an example of decentralization of an order, as mentioned earlier. In politics, authoritative system improvements are also lengthy processes. Votes that possess a distinct majority advantage are imperative, and they must be executed in a time-consuming and costly manner. The process of enacting or altering laws necessitates a considerable amount of time to undergo voting in all instances. An overview of how laws are created in Germany can be found in [48]. It becomes evident that changes take time. Thus, one also has here an example of poorer scalability due to increased decentralization.

When describing IT systems with the systems described in the figure 6.1, you must distinguish between the organization and the system design. Centralized IT organizations are a model wherein resources and employees are managed by a central management structure. This model facilitates a unified approach, enhances communication, and frequently reduces redundant tasks. This can lead to a more efficient use of resources and reduce IT costs. On the negative side, IT decisions may be positioned distant from end users, resulting in a potential for slower resolution times for issues. There is also the possibility of a halt in operations in the event that the central unit becomes unavailable. The impact of this would be felt throughout the entire system.
The central IT organization is different from the decentralized IT organization. In a decentralized organization, task areas are divided among different units. This model can reduce response times and provide greater flexibility, as each unit can adapt its IT strategy to its own specific needs. But a decentralized IT structure can also lead to inefficient use of resources because tasks and functions may be duplicated across multiple units. It may also be more difficult to implement a unified IT strategy or policy. This representation of centralized and decentralized systems can be compared to Baran's explanation in figure 6.1 (a) and (b). The remaining representation (c) is rather uncommon, as it implies that the units operate autonomously, yet possess the ability to cooperate with one another.
Furthermore, mixed systems might exist within the organization. In the work [254] a further classification is made in the organization. In that context, a distinction is drawn between distributed systems at the decision-

making level and decentralized systems at the communication level.
Besides the organization, however, the architecture can also be described in terms of centralization. In a centralized system, all resources and control are located in one place. For example, a centralized server may store a database that is accessed by all clients on the network. Centralized systems are typically easier to manage and control, but can be a SPOF and could potentially create a bottleneck in performance.
Decentralized systems have no central point of control. Instead, multiple nodes or systems can make decisions simultaneously. Each node has the capability to operate autonomously and possesses its own resources that are utilized to execute its tasks. However, this can also result in ambiguities, and administration can become more complex.
In a distributed IT system, resources are spread across multiple nodes, which may be geographically separated, but communicate and coordinate with each other over a network. These nodes can collaborate on tasks or utilize the resources of other nodes. The combination of centralized and decentralized systems can bring their management, performance, and security challenges to distributed systems. Many such systems include social media services that make data available globally, but organize and store it in meaningful regional subsystems primarily through Content Delivery Network (CDN) or similar techniques.
The costs incurred for decentralized systems must therefore be proportional to the respective projects and must be justified. It is not feasible to provide an obligatory and comprehensive response to such a justification. Rather, it is imperative to conduct individual analyses and engage in discourse. The past has shown that many projects related to blockchains and DLTs have failed (3.2 Digitalization and Blockchains for Public Administration).
The different perspectives have already been addressed. However, one should additionally realize that there are also different dimensions on which systems can operate centrally or decentrally. A report commissioned by the Pentagon from Trail of Bits identified a total of six dimensions for blockchains. According to the report, these are [35, p.3]:

- Authoritative centrality: What is the minimum number of entities necessary to disrupt the system? This number is called the Nakamoto coefficient, and the closer this value is to one, the more centralized the system. This is also often referred to as “Governance Centrality”.
- Consensus centrality: Similar to authoritative centrality, to what extent is the source of consensus (e.g.,PoW) centralized? Does a single entity (like a mining pool) control an undue amount of the network’s hashing power?
- Motivational centrality: How are participants disincentivized from acting maliciously (e.g., posting malformed or incorrect data)? To what extent are these incentives centrally controlled? How, if at all, can the rights of a malicious participant be revoked?
- Topological centrality: How resistant is the consensus network to disruption? Is there a subset of nodes that form a vital bridge in the network, without which the network would become bifurcated?
- Network centrality: Are the nodes sufficiently geographically dispersed such that they are uniformly distributed across the internet? What would happen if a malicious internet service provider (ISP) or nation-state decided to block or filter all DLT traffic?
- Software centrality: To what extent is the safety of the DLT dependent on the security of the software on which it runs? Any bug in the software (either inadvertent or intentional) could invalidate the invariants of the DLT, e.g., breaking immutability. If there is ambiguity in the DLT’s specification, two independently developed software clients might disagree, causing a fork in the blockchain. An upstream vulnerability in a dependency shared by the two clients can similarly affect their operation.

These dimensions are incredibly extensive and also incorporate outdated definitions. For example, Baran definition of a decentralized system can be found in the authoritative centrality.

It can be concluded that the term "decentrality" is difficult to define or categorize. The answer varies greatly depending on the viewpoint (system design or organization). The process of decentralization is always associated with expenses, as exemplified by the CAP theorem. Two questions remain unanswered. Is blockchain technology truly decentralized, and can the degree of decentralization be quantified?
Let us commence with the second inquiry, as its response has the potential to have a direct impact on the first. In the literature, it is common to find three methods. Among these are the Gini coefficient, Shannon entropy, and the Nakamoto coefficient. The initial two have been previously described before the emergence of blockchains, and as such, they have been utilized solely as a measure for the specific use case. The Nakamoto coefficient, on the other hand, measures decentralization by measuring the number of entities (such as nodes or miners) required to control more than 50% of a network's resources. These resources could include computing power, voting power, cryptocurrency ownership, or other types of control or influence. [179]
The SoK framework presents a new framework for examining decentralization. Eight dimensions are considered here. These are: "(1) hardware, (2) software, (3) network, (4) consensus, (5) economics ("tokenomics"), (6) API, (7) governance, and (8) geography". [157]
At this point, it is clear that the task of measuring decentralization is not straightforward. A reinforcement of this statement can be found in [168]. The paper argues that achieving complete decentralization is impractical. Accordingly, the question changes from "do we have a decentralized system?" to "do we have a sufficiently decentralized system". And for this one can choose the individually necessary dimensions for.
The work mentioned has also often been related to blockchains. This is a good segue to answer the last open question. Are blockchains decentralized enough to deliver on their promises. For this, the aforementioned work provides an answering guide, as measures and metrics have been defined and evaluated that could give one a number as to the degree of decentralization. The evaluation of this figure, however, is left to everyone. It can be stated, however, that in the sense of Baran's definition of the term, that decentralization means resistance to failures, it is precisely the two blockchains used, BTC and ETH, that promise sufficient decentralization in their public instances. If one builds a private instance, the metrics have to be re-evaluated.

In conclusion, decentralization in general need not be a good thing. But there are areas, as in politics, the separation of powers for example, where a decentralized system is desirable. How far decentralized such a system is therefore an important question, since one must be able to answer whether the system is sufficiently decentralized. Decentralization is often achieved through transparency and protection against manipulation.

6.4 Blockchain or no Blockchain?

The question whether blockchain technologies should be used or are even useful is a frequent topic of discussion in the literature, but even more so in the mainstream media. There is a regularity in discussion when the price of cryptocurrencies falls or rises. If the price goes down, people often say that the technology is going to fail. An overview of the alleged failure of BTC can be found in the source [1]. One also recognizes an accumulation of negative articles with the course ups and downs. It is therefore important to make the application possibility independent of any price fluctuations. This information is not intended to be utilized to draw conclusions regarding the quality of the technology and should not be utilized for that purpose.

Blockchain is often compared to the history of the Internet in terms of development and emergence. The advent of the Internet has significantly altered both business procedures and the means of communication

among individuals. However, its success was not predetermined. In the early stages of its creation, the Internet was a military project that evolved into a research network of universities. A bubble arose, which gained recognition as the dotcom bubble, or Internet bubble. The sudden increase in Internet use, however, also brought many difficulties. Among them was unawareness on all sides. Strategic, technical, and also operational measures determine whether a company survives the bubble or goes bankrupt. The dotcom bubble was a stock market speculative bubble that lasted from 1995 to around 2000. During this time, the stock market experienced unprecedented increases in stock prices, especially for technology and Internet companies.
The bubble was driven by several suspected factors, including

The Rise of the Internet: Investors began to see the potential for massive profits from online business models as the Internet became more widespread. This opportunity was taken advantage of by many new companies, known as "dotcoms."

Speculative investments: Several of these new dotcom companies had little or no profits, but that didn't stop investors from investing heavily in them. The consensus was that profits would be realized in the future, once the companies had attained sufficient market penetration.

Monetary policy easing: In the late 1990s, the Federal Reserve lowered interest rates, which increased the availability of money for investment.

During the late 1990s, the Federal Reserve lowered interest rates, which increased the availability of money for investment. The internet bubble finally burst in 2000. Many of the high-valued online businesses that had achieved substantial valuations ended up insolvent because they were unable to realize the anticipated revenue. This resulted in significant losses for investors and a sharp decline in the stock market.
Nevertheless, a few firms that emerged from the epoch of internet mania have persevered and have enjoyed considerable success. These companies include Amazon and Google, which have emerged as among the most valuable enterprises globally.
The dotcom bubble is a renowned illustration of a speculative bubble and is frequently cited as a cautionary tale regarding the perils of excessive speculation and exaggerated valuations. Further elaboration and detailed rationales for further success or failure are examined and listed in the paper [213].
The rise and fall of Internet hype is comparable to the development of blockchain technology, especially for cryptocurrencies. The prospect of smart contracts via ETH generated a lot of hype, thereby facilitating the implementation of one's own token. This Initial Coin Offering (ICO) hype affected the stock market, just as it did back in the dotcom bubble. Furthermore, the first companies went bankrupt, and financial damage was done through misallocations, fraud, and theft.
Besides the shared financial history, both the Internet and the blockchain were designed as decentralized systems. No institution or even country should be able to significantly intervene in the infrastructure. Using the Internet, we could see that these decentralized structures were in place. Nevertheless, there exists a strong centralization that significantly affects internet usage today. A group of technology companies that are economically strong and control our interaction with technology and the Internet is called Google, Apple, Facebook, Amazon, Microsoft (GAFAM). Therefore, centralization is becoming increasingly prevalent. One can discern a resemblance within the blockchain realm. The inclination towards centralized wallet management is resulting in the emergence of exchanges, which oversee exchange assets and custody in a manner akin to that of a bank, and hardware offerings for the custody of crypto assets.
The Internet has raised questions about privacy, security, and information. In a similar vein, blockchain technology presents challenges such as scalability, energy consumption, and the utilization of cryptocurrencies for illicit purposes. Perhaps, blockchain technology can address two types of problems facing the Internet. This would be privacy and security. But further observation and exploration will be required in order to more accurately identify the methods.

Despite the strong overlaps between the Internet story and the blockchain story, no prediction can be made about the future. However, the comparison gives us two conclusions. Blockchain technology should be evaluated as an infrastructure first. By inference, it is not feasible to assign the complete expense of operating a blockchain to a singular project. Similar to how one would not create a separate internet for each website. Secondly, the future possibilities and potential projects for this infrastructure are uncertain. This complicates the design today. Still, adjustments are possible later, as the update “the merge” at ETH from PoW to Proof-of-Stake (PoS) consensus showed.

The inability to execute tasks efficiently is one of the main reasons cited for avoiding blockchain usage. The blockchain is regarded as a database with inadequate performance. Since we have already established database systems, the question arises as to the specific use case. Because many kinds of blockchain projects have already been founded and brought to market. Thus, in the ICO hype, many new tokens were created to be applied to specific projects. Nonetheless, the application was unsuccessful in the majority of instances. Those projects that are currently demonstrating their dominance in the market are of a financial nature. They enable the exchange of diverse assets, integrate off-chain information into the respective ledgers, or provide a token with a predetermined exchange rate. Despite the existence of financial transactions, projects remain a rare occurrence. Currently, supply chains and digital identities are not yet realized through a blockchain in productive operation. Even after the Non-Fungible Token (NFT) hype, in which monkey pictures were confirmed as a link in the possession of a user, projects in production are missing. This conception was prone to fraud and error, since if the link were to break down, the claim of ownership would be extinguished. In addition, there were no legal regulations in place to prove or justify ownership. It was also the case that this hype collapsed financially. However, the market still exists and the work in 5.1.3 Blockchain Programms for AAA-me showed that NFTs have potential, but needs better design in terms of security. Likewise, it is noticeable that the sense of a technology is justified in it, how the current course stands. The technology is overvalued in the hype phase. It is considered undervalued if the price regresses.
For the purpose of optimizing blockchain performance, it is recommended to refer to the CAP theorem presented in the section 2.2 Blockchain Technologies, accompanied by the image 2.1. Please note that it is not possible to build or improve a system in more than two dimensions. The dimensions are divided into three points: decentralization, security, and scalability. The term security refers to the system’s consistency. If one wishes to improve all three points, one could relate the problem to the computer science theoretical problem $P = NP$ and solve it [119]. It is regarded as one of the most significant unresolved issues in the field of computer science. The limit of what computers are theoretically capable of computing is determined by this question. Both sides of the equation represent a class of issues. The P refers to the class of problems that can be computed in polynomial time, while the NP refers to the class of problems that can be efficiently verified, although finding the solution may require a considerable amount of time. If the equation could be resolved, one would have discovered a method that can be solved in polynomial time, i.e., effectively, to solve the issues associated with the NP class. Several of the problems encountered by NP are the basis for today’s common cryptographic methods. For example, the factorization of large numbers in the RSA method. Solving the scaling problem of blockchains while maintaining decentralization and security could be considered such an algorithm. Therefore, the potential damage would be huge. However, it is more likely that the issue will not be resolved, and solutions touting a scalable blockchain will have realized this at the expense of decentralization or resiliency (consistency).

The notion that blockchains provide solutions to issues that are not yet present or would not exist at all without this tech is a frequent point of contention. This argument bears a strong resemblance to a frequently uttered statement regarding mathematics. It is ironically argued that mathematicians solve problems that they would not have without mathematics. However, the context is crucial. As mathematics serves as a means of resolving issues that may otherwise prove challenging, blockchain holds the potential to address

both present and future challenges. The expansion of solutions is facilitated by the advancement of novel technologies, resulting in an expanded range of options.
Throughout history, it has been shown that discoveries and inventions were often initially made without any apparent practical use, but later found important fields of application. A well-known example of this is prime number research. Prime numbers initially appeared to be a pure mathematical curiosity with no practical application, but today they are the basis of cryptography, which enables the secure exchange of information on the Internet.
The situation is similar with microwave technology. Originally developed in the 1940s for radar, it found its way into our kitchens accidentally and has been revolutionizing our everyday lives ever since. Fractals are another example. These mathematical entities, characterized by self-similarity, long seemed to be of academic interest only. But today they are used in a wide variety of fields, including computer graphics and signaling, modeling natural phenomena, and even financial analysis.
In this sense, blockchain technology could also be seen as a solution to problems we have yet to discover. However, this very characteristic makes it an exciting and valuable tool for the future. After all, the ability to develop new solutions before we fully understand the problems involved is a crucial aspect of progress and innovation.

One of the most important features of a blockchain is decentralization. This enables the construction of an infrastructure that is invulnerable to censorship. The decentralized nature of this system implies that no single entity, whether it be a government or private enterprise, has the ability to regulate, impede, or reverse transactions. This aspect is particularly important in contexts where freedom of expression or commerce may be at risk. An example is sending money to countries with strict capital controls or political instability. Any technology, including blockchain, can be used both for good and for shady purposes. A comparison can be made here with the Tor network, which relies on anonymity. This is used on the one hand by activists and journalists working in countries where freedom of expression is suppressed, but also by criminals who want to hide illegal activities. Blockchain technology has the potential to be utilized for the purpose of laundering funds or circumventing legal restrictions in general. Nonetheless, the technology has the potential to aid in combating criminal activity. Because the ledger is publicly viewable and the use of a wallet can no longer be denied upon proof of ownership. Furthermore, one is capable of tracing all transactions once the person responsible is identified. The successful recovery of the ransom from a pipeline hack in the United States is a noteworthy illustration of successful identification of the owner of a wallet, as documented in [227].
Despite their utilization in both ethical and questionable settings, both blockchain technology and the Tor network demonstrate how decentralized systems possess the capability to alter the balance of power and empower individuals over centralized authorities. It is ultimately up to us how we use these tools and how we maintain the balance between freedom and responsibility.

The discussion surrounding blockchain technology has changed noticeably over the years. While in the early days, the mere mention of the term blockchain was enough to trigger a run on the stock exchanges, today a more differentiated view is required.
Critics point to challenges in the context of blockchain, such as the lack of a directly convincing use case, comparatively poor performance, the possibility of its exploitation for criminal activities, and a lack of evidence of the need for decentralization. Nevertheless, these arguments can be countered with positive aspects that underscore the benefits and potential of blockchain technology. Not explicitly addressed were the problems of oracle and transparency.
The main advantage lies in the increased resilience to SPOF and decentralization. This has already given rise to beneficial applications, particularly in the area of financial transactions in corrupt or structurally underdeveloped regions [224]. Further, blockchains have proven useful in the past in investigating criminal cases [224]. It is also useful to think of blockchain not just as a technology, but as a type of infrastructure. This

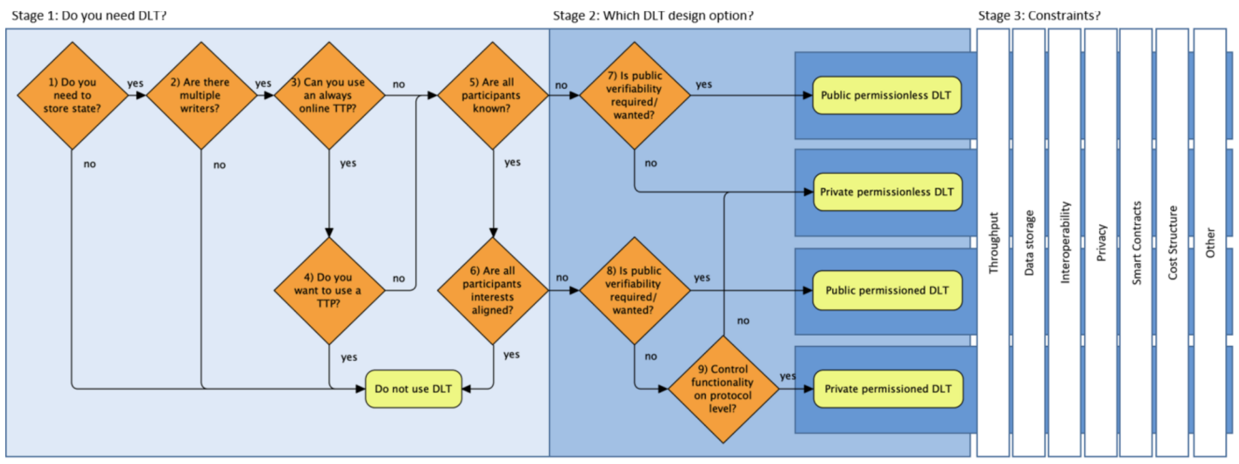

Stage 1: Do you need DLT?
Stage 2: Which DLT design option?
Stage 3: Constraints?
1) Do you need to store state?
2) Are there multiple writers?
3) Can you use an always online TTP?
4) Do you want to use a TTP?
5) Are all participants known?
6) Are all participants interests aligned?
7) Is public verifiability required/ wanted?
8) Is public verifiability required/ wanted?
9) Control functionality on protocol level?
yes
no
Do not use DLT
Public permissionless DLT
Private permissionless DLT
Public permissioned DLT
Private permissioned DLT
Throughput
Data storage
Interoperability
Privacy
Smart Contracts
Cost Structure
Other

Figure 6.2: Presentation of a decision framework for deciding the applications of blockchains and DLTs. [144]

makes it possible to evaluate the costs associated with its implementation and use differently. However, it is also evident that the blockchain technology has been misused in the past, for example, when it comes to exit heists or faulty investment solutions.

Ultimately, this ambivalence leads to a rather nuanced view of blockchain technology. It is difficult to make a general statement about its value and usefulness, as these depend heavily on the specific context and project in which it is used. Each project therefore requires individual consideration and decision.

6.4.1 Identifying the Need for Decentralized Approaches

For some time, attempts have been made to identify the use case for blockchain technologies, despite the digital currency. Often referred to in the literature as a "killer application", this refers to identifying a problem that can only be addressed by utilizing a blockchain. The purpose of this chapter is to aid in determining, on an individual basis, whether the use of a DLT is a viable option.

The question has been a topic of discussion for a long time in the literature. One can find many representations and frameworks on the title "Do you really need a blockchain?" both in journalistic media [125], public administrations [269], and in scientific papers [265, 144].

Numerous of the aforementioned representations bear a striking resemblance in their fundamental characteristics. In the figure 6.2, this can be observed in Stage 01 (the area in light blue). The question whether a blockchain can be used in a meaningful way is answered by the question whether multiple entities are involved in the process of getting and creating data. Some earlier representations of this graphic have argued more generally in the first point that if a relational database could perform the tasks, the blockchain would not be necessary. However, this generalization is not accurate. Later on, the question of decentralization is tested by the question of participating users. Nonetheless, this approach would be insufficient in the general scenario of the initial assessment, as numerous initiatives would have already ceased operations subsequent to the initial inquiry. It is more appropriate to use the wording used in the diagram, as it is more specific and allows testing for decentralization.

Stage 02 then describes access to the ledger. Do you need a public or (partially) private blockchain? The final Stage 03 clarifies constraints, which should provide answers to questions of governance, scaling, and publicity. Questions that have also been addressed in the Thesis. More detailed insights on the corresponding questions and constraints can be found in the source [144].

The question that arises is, are there projects that meet these requirements and provide an answer to the question posed at the beginning? Is there a killer app that justifies the use of a blockchain? In order to anticipate the answer, it is no. None of the projects can be comprehended as an application of this nature. Nonetheless, there exist certain advantages in certain domains where its utilization is deemed worthwhile. In order to comprehend the utilization, we examine two distinct projects and attempt to utilize the framework derived from the work [144]. Furthermore, we intend to examine the projects to determine their success or stability. Three categories should be investigated. The *technical metrics* shed light on the technical aspects. How is the assurance of high code quality ensured, are their security measures and procedures in place, and on which blockchain technology is the project based? In the case of *financial metrics*, one examines the financial performance of the project. Data on market capitalization or trading volume can be used. Lastly, the *user-related metrics* should be highlighted. The number of users, user growth, and user interaction are key factors. It is important to note that not all three categories can be evaluated equally well for all projects. Furthermore, the primary objective of this dissertation is not to conduct a comprehensive evaluation of blockchain projects. Therefore, it will only provide a rough overview. The two projects under study are not comparable. One project is a BTC Lightning implementation. Lightning Network Daemon (LND) is given as one of the most widely used implementations of the Lightning network in the world. The software was developed by Lightning Labs and is capable of supporting multiple platforms and programming languages. LND can also work with various BTC and Litecoin nodes [177]. More on lightning in 2.2.4 Solutions for Scalability Problem. The second project is an Oracle solution provided by Chainlink. As a result, it permits data to be inserted into the accounting system outside of it. The off-chain world is connected to the on-chain world by this. This enables smart contracts to access and process external data. [60]

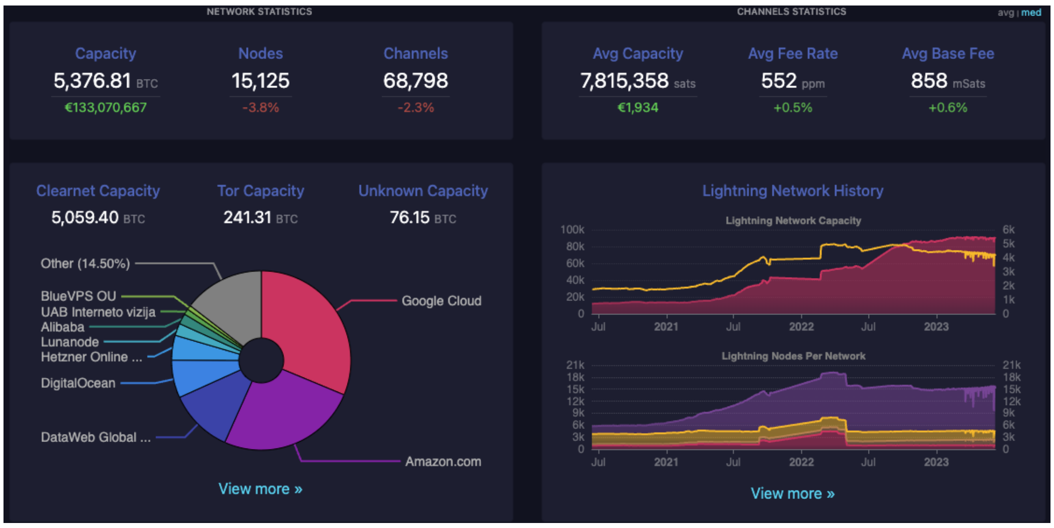
The two projects thus exhibit divergences not only in terms of the fundamental blockchain technologies, but also in their distinct implementations. While Chainlink uses smart contracts to implement the services, LND is that most of the computations happen off-chain and only a few operations are needed on the ledger. This brings us to a wider debate that has developed in the crypto community: the tradeoff between on-chain and off-chain computations. On-chain computations, such as those used by Chainlink with smart contracts, involve executing all transactions and contracts on the blockchain itself. Off-chain computations, on the other hand, as used in the Bitcoin Lightning Network, allow transactions and contracts to be processed off the main blockchain and only selected information to be written to the blockchain for final settlement.

The debate focuses on which method is more efficient, secure, and suitable for various applications. On-chain computations offer high security and immutability, as each transaction is permanently recorded on the blockchain. Off-chain computations, on the other hand, can be faster and more cost-effective because they can work around block size and network capacity limitations. So the two approaches have their advantages and disadvantages, and their applicability may vary depending on the specific use case. It remains an exciting topic that will greatly influence the further development of blockchain technology.

Lightning Network Daemon (LND)

The goal of the Lightning Network is to increase the scalability of the BTC blockchain by accumulating transactions. The accumulation is done by bilateral transmission of cryptographic signed balance updates. Although the majority of implementations involve BTC, the concept and method can also be applied to other blockchain technologies. The basic idea remains that one can send values. These values have the potential to represent a currency or assume a corresponding monetary value.

Let's look at the metrics of LND. From the figure 6.3 of the Lightning statistics, we can see that there was a continuous increase in available and transferable BTC. The number of open channels for dispatch experienced a gradual decrease in the first half, followed by a steady decrease thereafter. The reason for the incline can only be guessed at here. Due to a new project, the utilization of the BTC blockchain is very high, as are the fees for transactions. Opening new channels is only worthwhile for larger amounts. Channels just to try out Lightning are thus less likely to be opened. This also explains why the capacity in the network has continued


NETWORK STATISTICS

Capacity 5,376.81 BTC €133,070,667

Nodes 15,125 -3.8%

Channels 68,798 -2.3%

CHANNELS STATISTICS avg | med

Avg Capacity 7,815,358 sats €1,934

Avg Fee Rate 552 ppm +0.5%

Avg Base Fee 858 mSats +0.6%

Clearnet Capacity 5,059.40 BTC

Tor Capacity 241.31 BTC

Unknown Capacity 76.15 BTC

Figure 6.3: Statistics of Lightning Metrics. [176]

to increase, even if only slightly. The exact causes, however, require their research, which is beyond the scope of this paper. However, one can state that a slight increase in interest persists. On the GitHub site of the project, one can also see that a lot of code is being programmed and that a larger community is contributing to it [177]. No more in-depth insight into the technical implementation should be given here, which could mean that the high activity is a good measure from a technical perspective. To the finances, there is a strong dependence on about the BTC in the market. This can be taken and interpreted by oneself.
There often remains a subjective character in the evaluation of a project, so it is difficult to give a final conclusion here. Especially in projects that are close to a border of transition, especially if this border has been drawn blurred. However, the above facts indicate that LND is a promising project and that it will run for some time.

Related to Lightning Network Daemon (LND), we look at the decision issues of a blockchain from the figure 6.2. We briefly summarize and justify each of the issues.

1. Store State?: **Yes**: Indeed, a global state is necessary to exchange and execute the transactions. As a scaling solution of the BTC blockchain, this ledger takes care of that.

2. Multiple Writers?: **Yes**: Each participant must be able to create transactions for value transfers.

3. Are There Trusted Third Party (TTP)?: **Yes**: In fact, service providers exist for sending values or money. Banks and private providers such as Paypal are examples. Service providers are also becoming established for the Lightning protocol.

4. Want a TTP?: **No**: For security reasons to avoid unconscious and conscious SPOF one would like to avoid TTPs. Or one cannot trust these.

5. Participants Known?: **No**: Not all participants are known, or even trusted. At least in the case of the application for BTC.

6. Public Verifiability?: **Yes**: Public verifiability is a necessary security measure to avoid double-spending and other attacks!

Through evaluation, it is clear that a public, unpermissioned blockchain is needed. LND offers an improvement in throughput and data storage as a potential scaling optimization solution. In addition, LND can also be used for other blockchains. This provides a great deal of interoperability. Aggregation of multiple transactions makes it more difficult for external observers to track the exact history. Nonetheless, crime-fighting institutions may also derive advantages from this approach, as non-repudiable evidence of transactions can be obtained once the relevant hardware has been confiscated and access secured. Smart contracts are not required, thereby minimizing the possibility of error through an increase in complexity. Depending on how much transaction fees one accumulates, the cost has gone down. However, the system also has opportunity costs that should not be hidden. For instance, the execution of the transactions in their entirety is more intricate, thereby increasing the likelihood of error. This increases additionally if so-called routing is needed. For this purpose, a credit must be allocated beforehand so that it can be moved off-chain at a later time. So it is like a pre-paid system. A rebalancing needs to be done on-chain.

Chainlink

The primary objective of Chainlink is to facilitate the transfer of data that is only accessible externally to a ledger into the ledger. This data is incorporated through the utilization of smart contracts and thoroughly scrutinized for its authenticity. Originally designed for ETH, the project has expanded to include other technologies. To use the service, there is a separate created Ethereum Request for Comments (ERC)-20 token called a link.
As with LND, this project is also drawn with a lot of activity on GitHub [212]. By using one's token, one can assume usage of the service for any supported blockchain. Between 2,000 and 8,000 transactions per day were recorded in the period from December 13, 2022, to June 11, 2023 [94]. Between five and 30 million links were sent each day. Again, there is a strong dependence on the blockchain infrastructures used. Nevertheless, a strong use of both financial and user-related metrics can be observed. The technical metrics also suggest high usage. In summary, although it remains a subjective decision, this project can be classified as successful. Along with high potential for continued use.

1. Store State?: **Yes**: Use case to be implemented by Chainlink. External data must be stored. Therefore, must be yes.
2. Multiple Writers?: **Yes**: Chainlink provides a service to store data on the ledger. The user decides which data from which sources. In addition, corresponding smart contracts can be deployed and managed by the user.
3. Are There TTP?: **Yes**: Chainlink can be considered as a service provider. As such, it ensures the integrity of the data, especially when multiple data sources can write to the ledger. But self-administration is possible through the own use of the contracts.
4. Want a TTP?: **No**: Possible at the data integrity level. But can be realized individually depending on the project without TTP. On a deeper level, there is a prevention of loss or manipulation of the dataset, and therefore one does not want TTP here in general.
5. Participants Known?: Depending on the type of blockchain used and the use case.
   a) **No**: Usually, and implemented on current blockchains, the participants in Chainlink are not all known.
   b) **Yes**: In the case of a (partially) private blockchain, participants are known.
6. Public Verifiability?:
   a) **Yes**: Public verifiability is a necessary security measure to avoid double-spending and other attacks!

b) **Yes**: The (partial) public verifiability is a necessary protection measure to avoid double-spending and other attacks and to exploit the data accordingly!

It should be noted that this evaluation is not intended to be a recommendation to use Chainlink. Rather, it is meant to be an exemplary example of how the decision model should and can help clarify the need for a blockchain. The evaluation shows that one should either use a fully public blockchain, or one that is restricted in writing, called a permissioned blockchain. What is exciting here is that the possible use of smart contracts enables the application of the software on one's own blockchain. This increases the application potential of Chainlink. Chainlink's product is itself to put data on the Blockchain. Thus, its need for a ledger is inherent. The importance of need is thus transferred to the projects that need Chainlink's data. If there were no projects, the application potential would be 0. However, due to the high usage and exploitation data shown, a slight tendency towards necessity can be assumed.
It is also exciting that Chainlink has provided a centralization. This again creates a dependency on trust. However, this dependency is to be enacted on a different level. While the trust level in the question of a blockchain is rooted in the fact that data creation and integrity can be compromised by manipulation, in the case presented, it is data integrity that must be ensured. It is possible for anyone to check the state of the data sources by observing them before validating the data in a block. However, subsequent rejection is significantly more difficult and is thus understood to be impossible.

Conclusion

The examples show how a project can be examined for a potential application of a DLT. The mentioned examples are, however, already in progress and are partially geared towards utilizing a distributed ledger. It is important to show whether the individual applications have this proximity to the technology. In the next chapter (6.4.2 Public Key Infrastructure vs. decentralized Public Key Infrastructure), a different case is examined. There, the already existing PKI systems are compared with a decentralized Public Key Infrastructure (dPKI) on a blockchain and necessities are discussed. Therefore, an already established procedure is ported and evaluated.

The utilization of the decision diagram aids in the evaluation process. To arrive at a quicker conclusion, the first part of the decision diagram (blue in the 6.2) can also be summarized as follows: Information from bilateral transactions is also required for other bilateral, or even multilateral interactions.
If this situation does not exist, the use of blockchain should be discussed again.

6.4.2 Public Key Infrastructure vs. decentralized Public Key Infrastructure

This work focuses on the implementation of the RADIUS protocol on blockchain technology. In this regard, the discussion will focus on the use cases of public administrations. So far, we have observed that certificates play a key role in the implementation of the RADIUS protocol. Digital certificates are not a novel invention and have been utilized numerous times in contemporary Internet communication. They are often used to secure website calls or ensure the integrity of email traffic. The concepts for password-free authentication are also based on digital certificates. You can also use certificates to protect Domain Name System (DNS) assignments. The list of possible uses for certificates appears long, and the need is often justified by a higher level of security in integrity and authenticity. Due to these factors, this matter will now be addressed in greater detail.
Despite this, certificates still have a serious disadvantage. Any person is capable of generating a certificate. Can the verification now ensure that the certificate was created from the correct place? This issue was extensively discussed in the section 2.4 Public Key Infrastructure, leading to the resolution of a Public Key

Infrastructure (PKI). For further details on the structure, please refer to the referenced chapter. To summarize, aPKI serves as a Trusted Third Party (TTP) and is incorporated into the certificate creation process. This allows a concatenation of certificates to be created and replicated. By trusting the PKI, a trust can be passed on to the certificate.

PKIs have the problem of distribution of a so called Certification Revocation List (CRL). The expired and revoked certificates are kept in the CRL. This list is essential for ensuring an existing authenticity and integrity in the certificates issued by them. A blockchain could be used as a transport layer to distribute this information. In this case, either the CRL itself would be managed on the blockchain by the PKI, or the certificate itself would be managed on the blockchain. The certificates on the blockchain was discussed in 5.1.3 Blockchain Programms for AAA-me implemented by smart contracts on the ETH blockchain and in chapter 4.4 Blockchain Transactions and Programs for Triple-A functions conceived. In addition to the blockchain approach, there are some methods for distributing the current CRL, such as the Server-based Certification Verification Protocol (SCVP), Online Certificate Status Protocol (OCSP), or OCSP stapling. In [161], they describe the CRLs distribution issues in more detail. This entails the construction of an infrastructure based on PKI that is accountable for authenticating certificates in real-time. Although, privacy is often compromised by these solutions, as the PKI can track a users' website history through certificate queries.

This dissertation demonstrated the practical viability of utilizing a blockchain for certificate management. But other works have dealt with it before this one. Among others, [206, 136]. In addition to PKI approaches, there are also approaches to implement a DNS on blockchain. These have been around from the beginning and are known as Namecoin [196], or Ethereum Name Service [92]. The latter is implemented as ERC-20 tokens on the Ethereum blockchain. All projects have in common that they want to decentralize the respective tasks.

If there are already many approaches to counter the problems of PKIs, why would you use a blockchain? Is the decentralization offered by the technology necessary, or is the expense excessive? The problems underlying a PKI system can be subsumed to the SPOF. Emerging issues could be triggered by mishaps, but they could also be triggered with intent. Numerous of the aforementioned solutions relied on distribution, in order to prevent the compromise of the entire system by the failure or abusive attack of a single entity. This can refer to the distribution of resources, duties, or duties among the various entities. For instance, a computer system can be replicated and configured in various locations to ensure that, in the event of an emergency, the malfunction of a particular component can be detected. But when authority or decision-making processes are distributed, this is referred to as decentralization. In addition, one finds in chapter 6.3 The (De-) Centrality Issue of Blockchains for more information and details. This reduces the outgoing danger of SPOF.

The danger arising from a SPOF is not just a subject of scientific inquiry. In the overview of documented failures of Certification Authorities (CAs) [247], some examples are shown that demonstrate errors or tampering. A more recent breakdown, albeit related to DNS, left customers of a German Internet provider unable to access the Internet when routed through the provider's servers [132]. In spring 2022, a Border Gateway Protocol (BGP)-hijacking attack was successfully carried out and Twitter data was routed via Russia [148]. This was made possible through the conscious or unconscious manipulation of the corresponding entries in a Resource Public Key Infrastructure (RPKI) [236]. As mentioned earlier, effective measures may include decentralization. In the work by [174], another method is investigated that allows tampering to be avoided. This is given by increased transparency. Providing clear documentation about the current status of certificates makes it possible for domain owners to quickly identify false certificates, such as those that could be used in MITM attacks.

The distribution, decentralization, and transparency-enhancing methods for securing PKIs suggest a positive effect on security through the use of blockchains and DLTs. It has the potential to combine all of the above measures. The resulting costs of a blockchain infrastructure need to be re-evaluated in this context. However, the technical feasibility has been demonstrated in numerous works ahead and also in this thesis.

In summary, there is a table in the 2019 paper [136] that compares common PKI systems with blockchain-based ones. There, the aforementioned reasons for a Blockchain based PKI become clear in a table.
Blockchain-based systems have other potentials besides increased resilience against SPOF. One of the advantages is that the infrastructure utilized can be repurposed. This can be meant at the certificate level. For example, a blockchain infrastructure with suitable wallet software can serve as PKI. However, other entities with certificates, such as RPKI, DNS or even Virtual Private Network (VPN) can also benefit from the Blockchain instance. Furthermore, the blockchain has the capability to operate beyond the certificates. As demonstrated in this thesis, the RADIUS protocol can be additionally mapped to the Blockchain. Furthermore, the currently most widespread application, the distribution, and management of values, is also possible in the area of possible integration. It is imperative to consider that each project has the potential to adversely affect the other projects. If one project scales poorly, this may have an impact on all of them, since if the pool of all open transactions is full, new transactions can only be created and validated at high cost for other projects. The need for clear planning of infrastructure capabilities and the needs of individual projects is crucial. This presents challenges for many system architects today, due to the multitude of setting and customization options of blockchains, the current lack of application besides public instances, and the lack of ability to foresee future projects today.
The report [160] examined how PKI systems are known and perceived in the business community. In addition, an outlook was cast on further integration and adoption. It was found that 23 % of firms use external PKI systems (cloud solutions, for example), 18 % use public PKI systems, and 42 % use an internal PKI. The companies that relied on external PKI providers anyway could easily be ported to a dPKI once a service provider offers it. The advantages are transparency and portability of certificates. The internal entities could consider whether a private blockchain could be used to allow various systems to be mapped over it. The question of the trust environment does one trust other entities of the company enough or insufficiently, or can the security factors be sufficient to justify the use, of the DLT.
If the public PKI systems port to a blockchain or Distributed Ledger- (DL-) technology, then their customers would also be included. Thus, all three of the aforementioned PKI implementation options are supportable by a blockchain.

In summary, the following points can be identified that make using a dPKI preferable to an ordinary PKI.

- Compromising the PKI: The security of PKI systems depends heavily on their trustworthiness. If compromised or acted upon improperly, attackers can create fake certificates that are mistaken for legitimate websites or services. This can lead to MITM where user data being intercepted and potentially manipulated. In other PKI systems, such as RPKI, this can lead to false routing and form an additional attack vector.

- Scaling and Management: PKI systems have problems with managing certificates in real time and communicating with subscribers. Consequently, some certificate revocations do not work as desired. The most common methods for certificate revocation are CRLs and OCSPs. However, as shown earlier, these methods have their weaknesses, such as latency, privacy, and security issues. Not only revoking certificates, but managing them in general can be problematic. The process of issuing, extending, revoking, and examining certificates can be particularly complex in large or distributed organizations. This induces security gaps.

- Trust Model: Classic PKI systems are designed hierarchically. This implies that there exists a central trust center that is authorized to grant authorizations for further PKI systems. These, in turn, possess additional authorizations. This creates a chain of trust that can be checked by the user and whose validity is based on trust in the root PKI. It is this central position that is the key strength and weakness of the overall system. If one loses trust in the root PKI, then all certificates issued under it are not trustworthy. No new state changes can be made in the event of failure. And they provide a target for deliberate manipulation through hacking attacks or censorship by states.

- Archiving and Transparency: The custody of expired certificates is a necessary step in maintaining the security of a PKI system. This task can be challenging to manage. Furthermore, all certificate state change processes should be traceable to ensure transparency and thus tamper resistance. In a blockchain, the inherent transparency and immutability of data appear to be tailored solutions to this problem.

- Superior Application: A PKI is needed in many areas. However, new (sub-) systems are often created and implemented. It would be possible to map several PKI systems using a blockchain as an infrastructure. Simultaneously, supplementary applications may be implemented, provided that the architecture accommodates them and does not exhibit inadequate scaling.

- Adoption and Integration Opportunities: The public blockchain technologies provide a diverse range of software solutions. These already take over most of the important and also cryptographic tasks for a secure app. These solutions, such as wallet or node software, can be incorporated into existing systems. Even if a blockchain interaction does not take place, the library can be used for other operations, such as signature validation.

At this juncture, it is noteworthy that the expression PKI has consistently been employed in this context. To be precise, it is imperative to mention in this chapter the role of a Certification Authority (CA), which is accountable for issuing certificates and is an integral component of PKI systems. However, since there is no PKI without CA and this term is more common in the literature, this chapter uses the more generalized term. The well-known blockchain systems, such as BTC, ETH, or Hyperledger Indy, appear to be designed to counter the current problems of PKI systems. It appears probable that further research in these areas will be conducted, and it is also a deliberate action. Nevertheless, the mentioned technology is not obligatory. Continuous improvements are currently delivered quickly, which makes it difficult to take stock of the technology.

6.4.3 Private Blockchains

It is frequently debated whether the utilization of a private blockchain is a viable option. The circumstance is caused by the frequent use of a BTC or BTC like blockchain in presenting the application. As a result of the consensus method PoW, this technology is indeed expensive. It is therefore important to consider a different consensus when evaluating whether to use a private blockchain. In chapter 2.2 Blockchain Technologies, one can find an explanation of consensus methods and a table 2.4 that shows the most popular consensus methods on the suitability of blockchain types.
A private blockchain is seen as a technology where there are restrictions on either writing to the ledger or reading. In order to distinguish between the two cases, the term permissioned for restricted write access has been established in the literature. A private blockchain is then meant to be restricted in read access. This is a short summary from the section 2.2 Blockchain Technologies.
A too small circle of users poses a certain problem, the so-called Oracle problem, which should not be ignored. The diversity and number of parties involved in a blockchain affect the security and immutability of the data. Some users pose a risk of a single entity or a small group gaining control of the blockchain and adjusting its properties. However, depending on the use case, it might make sense to shrink the user base. For applications that are tailored to specific requirements and control mechanisms, the constrained user base may contribute to enhancing the efficiency and performance of the application while simultaneously reducing expenses.

The utilization of restricted forms of blockchain, whether private or permissioned, is gaining significance in both research and industry. Through continuous evaluation, we can identify novel applications and utilize the technology in innovative ways. Although they may not offer the same degree of decentralization as a public, private and permissioned blockchains present novel opportunities for utilization in specialized use cases.
Permissioned blockchains bring flexibility that may be necessary on an application-specific basis. For instance, they can be tailored in terms of accessibility, transaction speed, and the volume of data that can be processed in a transaction. In addition, it may provide greater control over the network in order to ensure data privacy or meet regulatory requirements. Hence, permissioned blockchains can turn out to be a valuable addition to numerous applications, although they may not offer the complete decentralization and immutability of public chains.
The study [162] compared surveys on what the current perception of the technology is in the industry. The advantages and disadvantages of public and private blockchains are detailed, and the differences are drawn. The results of the survey indicated that larger companies, especially those focused on finance, were concerned about the technology.

Ultimately, it remains a decision for the project organization whether a public or (partially) private blockchain should be used. The decisive factors here are which participants take part and how flexibly the blockchain infrastructure must be used. The flexibility is ultimately bought by the partial loss of decentralization. The choice of consensus has an enormous impact on flexibility, decentralization and running costs.

6.5 Blockchain for Public Administrations

A research question of this dissertation is the use and impact of blockchains for public administrations. We have previously identified the potential and complexity of an eduroam system and integrated it into a HZD public administration system, but here we will discuss its use in general.
In chapter 3.2 Digitalization and Blockchains for Public Administration, there were described the driver's license and certificate projects, that were quickly closed. As a Self-Sovereign Identity (SSI) system on a blockchain, these projects are significant for evaluating an application for public administrations. Nonetheless, the outcomes derived from these projects may exhibit considerable variances. Critics see themselves confirmed that blockchain technologies are not suitable for public administrations. If one explicitly identified the problems that led to the failure, proponents of the technology cite that it was not the ledger that was corrupted, but a central entity (e.g., a web server) that had caused the failure. The read access was not compromised, but neither was an interface provided to perform this function without the web server. This turned the instance into a SPOF, which the technology was supposed to avoid.
However, blockchain projects have not solely been subjected to testing in Germany. Estonia provides an illustration that can be readily observed and assimilated into existing systems through the diligent research and integration of (blockchain) technology. However, the use of a private and access-restricted blockchain raises questions about the main benefit. Naming it decentralizing digital processes. Other problems could be identified and gathered in the work [230]. With Belgium, one can find another project that has tested SSI on a blockchain for digital processes in public administration. But here, too, strong critical opinions come to light in the work [184]. Even as the potential of blockchain continues to be held up. The first administration to actively engage with blockchains is the canton of Zug in Switzerland. Here, it was possible to use BTC early on to pay for local services, such as commissioning an ID card. In addition to using cryptocurrencies as an actual payment medium, however, the administration is also issuing a "guide to blockchain in cantonal administration Zurich" [269].
Even though the projects strongly had the focus of a SSI and this theoretically does not need a blockchain as a basic system, the projects mentioned in this chapter have been implemented with a write-constrained

blockchain. Regardless of the reasons for failure or identified reasons and potential for application, the lessons learned from these projects are a valuable experience for future development.
The enlightenment in the projects resulted, on the one hand, in a diminution in enthusiasm for further publicly funded blockchain research that is not exclusive to academic institutions and, in the aforementioned, the pending projects being withdrawn from publication. An indication of this can be observed in a blockchain strategy presented in 2019 [34] and in the election program for the 2021 Bundestag elections. The term blockchain appears only three times in relatively general formulations [165].

By implementing a radius system on the blockchain, we were able to understand that the tasks of the acronym AAA are technically doable on a blockchain. Furthermore, we have observed that this solution likewise demonstrates the potential for overcoming SPOF resistance. Therefore, further research should be done to recommend an explicit architecture and thus also potentially compare the costs compared to conventional systems.
There is also significant interest in the implementation of a digital identity for various processes in public administrations. This aspect is considered to be the most promising application opportunity for administrations in the context of blockchain technology. In particular, the Federal Republic of Germany as a federal system offers a wide range of opportunities for decentralized applications. This is because federalism implies mutual control and shared power, aspects that are fundamentally embedded in the principle of blockchain. Despite its promising potential, several factors pose significant challenges. These include, in particular, regulatory issues that arise in international systems. Which law should take effect when the laws of different jurisdictions conflict? How are global blockchains able to exist and operate under different legal regimes?
Another important aspect is the protection of data. Protection of personal data is becoming increasingly important and complex in the age of digitalization. Are blockchains capable of ensuring that data is stored in a decentralized, but protected manner? Blockchains are supposed to deliver decentralization, but this approach often runs diametrically to existing structures at the institutional level. Many public institutions are designed for centralized operations and controls, making it difficult to adopt decentralized technologies. Technical challenges should also not be underestimated. These include dependence on public projects and the complexity of building infrastructure. Many participants must actively participate, and experience has shown that this can be a critical barrier to adoption of new technologies.
Furthermore, the definition of the term "decentralized" is unclear. There are numerous interpretations of this term, and its necessity cannot be presented or refuted in a clear, structured way. This ambiguity frequently leads to misinterpretation and miscommunication during the implementation and utilization of blockchain technologies. Despite these challenges, blockchain technology still offers many opportunities to be explored and exploited.

To summarize, this dissertation demonstrates that the utilization of blockchains in public administrations is a highly intricate and promising subject that necessitates a thorough examination of technological, legal, and organizational obstacles. The most significant obstacles arise from the technical implementation and the handling of crucial aspects such as data protection, decentralization, and regulation. Mistakes in implementation led to project failure in some cases, but these failures also offer important lessons for future efforts. Observed difficulties are not necessarily a failure of the blockchain itself, but often a result of the way it was implemented. In conclusion, the dissertation emphasizes that blockchain in public administration is neither a panacea nor a failed technology. It is a promising and complex technology that requires careful examination, understanding, and implementation. Its mix of potential and complexity makes it an exciting field of research that will require continued intensive investigation and experimentation. It offers numerous possibilities, but can only reach its full potential through careful analysis, planning, and implementation. This paper contributes to deepening the understanding of this technology and to advancing the debate on how it can be used most effectively in public administration.

6.6 Conclusion

This thesis sought to investigate the suitability of blockchain technologies for public administrations. The RADIUS protocol was intended to serve as an exemplary implementation to clarify and highlight any porting and problems that come with it. This motivation led to an extensive investigation that examined current and past projects integrating blockchain technologies for RADIUS protocols such as eduroam, as well as for public administrations. The process revealed numerous issues, which entitles the blockchain to be legitimately questioned. However, what were the narratives and motivations that ultimately paved the way for blockchain and its continued utilization, and what issues were identified?
The pursuit of decentralization is prominently a component of the objectives and aspirations. The term is utilized when attempting to promote a blockchain as a viable solution. This albeit ambiguous term possesses a favorable connotation in numerous contexts. However, due to the lack of an agreed definition, it appears that this term is often misused. This dissertation has demonstrated how the terminology can be understood and the implications this may have for blockchain projects. In general, it is recommended to evaluate the term decentralization objectively and according to individual requirements. Furthermore, it is imperative to consider the impact of the perspective. A decentralized structure of PKI systems can currently be assumed, wherein each provider implements its systems. A blockchain solution would homogenize the architecture of the PKI implementation and store it on a singular instance. From this perspective, one would have centralization. One insight gained is that any decentralization also has centralization characteristics, depending on the perspective from which one looks.
One crucial term that cannot be overlooked when discussing decentralization is the Oracle issue. As this is precisely the factor that hinders the efforts to decentralize projects with a DLT. The Oracle issue refers to the lack of ability to port blockchain-unrelated data into the ledger. Any potential Oracle service may result in the reduction of decentralization to a centralized service. We saw the Oracle problem in the federal government's failed certificate projects (3.2 Digitalization and Blockchains for Public Administration).
Additional issues arise when attempting to identify the cost of a blockchain or DLT. Due to the sheer number of configuration options, as with consensus, transaction processing, and storage, it is not possible to perform a general cost illustration. This leads to issues such as scaling, which in turn leads to poor ecological balance and transparency. The development of comparable criteria from these spongy numbers appears to be a daunting undertaking. However, this comparability is necessary in order for existing systems to be reconsidered and, if necessary, mapped to the blockchain.
Throughout the history of blockchain technology, it has been extensively used for illicit and abusive activities. Silk Road was a non-regulated platform for the exchange of goods and services that allowed payment via BTC. This was intended to provide sufficient anonymity at the time. Emerging projects in the ICO hype that resulted in exit scams, and the same then in the NFT hype, leave a stale taste attached to the technology. These aspects, allied to the issues already mentioned, demonstrate the complexity and ambiguity of the technology when it comes to its use and comprehension.

To reduce the negative impact, solutions could be found for every named problem. First, it is imperative to consider the technology as a database solution for a specific project. Second, it's important to think of the blockchain as an infrastructure. Although the conception of the internet is not mandatory for web applications, it is essential for their success. Now you can move this analogy for DLTs, meaning that the infrastructure is important but should not be redeveloped for each project. Based on this insight, one might conclude that a blockchain should not be designed for a single project. Instead, various projects should be discussed for deployment. This gives greater importance to technologies that are unrestricted to the public and reduces costs many times over. Furthermore, the issue of identifying participants becomes more persuasive, as there is no need to identify a specific undertaking.
For the mentioned problems, solutions must be implemented architecturally. The scaling and energy issues

can be addressed with an appropriate consensus process. For the Oracle issue, it would be necessary to examine the rationale for using the blockchain in more detail. If the issue can be resolved through the use of a certificate or similar means, cryptographic techniques can be employed to enhance privacy.

Why bother with the effort of designing or finding the best-fit blockchain technology when there are already core technologies that perform most of the stated tasks? The gathered arguments presented in this section can be categorized into two primary propositions.

Conclusion 1. *The reduction of the Single Point of Failure (SPOF).*

The decentralization created by the high availability of data on all nodes and the distribution of write permissions ensures that the central locations for any attacks or accidents are reduced. Confirming this assumption, common protection measures against SPOF rely on resource distribution. Comparisons of centralized and decentralized instances have been discussed extensively in the comparisons of PKI systems with dPKI systems (see 6.4.2 Public Key Infrastructure vs. decentralized Public Key Infrastructure). Continuing the thought, a DLT system can be applied in other places, such as other network components like DNS or RPKI, or for digital identities in general that should be able to obtain credentials through attestations. This continuation of the consideration leads to the second theorem:

Conclusion 2. *Result from bidirectional communication is used for other bi- or multidirectional communication.*

It is central in many technologies to use results from one interaction for further bidirectional or multidirectional communications. To be clear, bidirectional means entity, so it doesn't have to be a single person but also can be a company or similar. To be informed about moving credits is essential in the BTC system in order to avoid the double-spend problem. Acquiring certificates is done in PKI-systems, which means they can be used for further processes, depending on the application, like TLS for website calls. The principle of the conclusion is particularly evident in the implementation of digital identities. Attestations, i.e., formal confirmations of identity characteristics, can be created in digital form and then used for validation in other contexts. For example, an attestation issued by a trusted authority could be used to provide proof of age, citizenship, or qualifications in various digital services. These attestations could be stored on the blockchain to ensure security and immutability, and could then be retrieved by different parties within a system.
This concept also has broad implications for blockchain technology itself. From this theorem, it can be deduced that information generated on the blockchain is best used, and external data should be brought in by Oracles less frequently. This limits the Oracle problem. Both coins/tokens and attestations of digital identities are examples of digital information that can be generated and stored on the blockchain. Based on the information at hand, it is then possible to interact with the physical world. Porting the inversion of physical information to the ledger is complex and leads to the Oracle problem.

Based on the advancements made by AAA-me, it can be inferred that hybrid porting to blockchain technologies is feasible and, conversely, this porting can be highly promising. The authentication times on a blockchain node were significantly faster than in the FreeRADIUS implementation, demonstrating this. Furthermore, the FreeRADIUS modules permit access to internal database systems without the necessity of employing additional Oracle services. Even a fast implementation of a SSO system would be possible with LDAP and Active Directory modules. Therefore, a physical variation of the known procedure can be readily realized.

6.6.1 Research Objectives

In this thesis, the research questions described in the introduction and defined before the thesis were addressed. Here you will find a brief overview of the answers distributed throughout the thesis.

The motivation was to establish a framework for implementing the RADIUS protocol on the blockchain, as mentioned at the beginning of the chapter. AAA-me has been demonstrated to be a framework that fulfills this requirement. By showing the corresponding code sections in chapter 5 one can observe that. In addition, the strengths and weaknesses of AAA-me were shown (6.1.4 Advantages and Disadvantages of using AAA-me). Through the utilization of a laboratory environment utilizing an eduroam network, an application could be demonstrated directly. In further considerations, additional application possibilities are presented. The question remains open only in the case of cost estimates. It is difficult to determine the cost of a highly configurable technology. Attempts to find figures for conventional PKI systems also proved difficult. Thus, a direct comparison remains elusive.

In addition to the explicit framework development, general blockchain technology related questions were also drafted. The answers to these are given in bullet point form as an overview. The full answers are in the thesis.

- **What are the capabilities of a blockchain, and what are the limitations?**
  Among the challenges faced by blockchain technologies, some are well-known. In this work, these and many others have been demonstrated. Among them are the problems with transparency, scaling, and Oracles. From these alone, the inquiry of the economic viability and significance of the technology is already evident. However, if one further discusses, possible applications can be identified. These application possibilities are subject to the considerations outlined in the two Conclusions outlined in the preceding section (Conclusion 1 and Conclusion 2). Further aspects, such as the further use of a blockchain instance and the associated consideration as infrastructure, are concealed in these two conclusions. In general, a blockchain is not a panacea, however, it does offer promising advantages. Among them are the resistance against SPOFs and the support for cryptographic systems.

- **How can a blockchain be incorporated into existing systems?**
  Integration with existing systems is a challenging task, as it can put them at risk. The newly integrated feature has the capability to generate novel attack vectors, thereby exerting an impact on my overall system. In conclusion, this case should be considered with the greatest caution. This encompasses the implementation of the blockchain system securely, as well as the creation of contingency protocols to mitigate any potential harm in the event of a breach of the blockchain system.

  Regarding the security aspects of the blockchain, there are two main questions that should be asked during integration. Do I want to operate my node, or do I want to access an external one? With the own node, the effort to operate it securely is incurred. With the external node, a solution must be found for avoiding a new SPOF exposure. This can be enabled by connecting to multiple nodes and a concept in case of conflicting data. The second inquiry concerns the process of implementation. Would it be preferable for me to conduct and oversee straightforward transactions involving information or would more intricate structures, resulting in the necessity of a programmable blockchain (also known as smart contracts), be required?

  The interface between a database and a blockchain can also be realized using Oracle services. It is possible to understand AAA-me as just that. Through this linkage, it is possible to define processes that can interact on and with the blockchain and have an impact on one's own system. With AAA-me, it was shown that registrations could be recorded on the blockchain and used on various other networks for authentication. It is recommended that certificates be designed over in consideration of privacy laws. These technologies also have the potential to achieve a passwordless authentication method. The modularity of the underlying libraries in AAA-me allows further integration into common database systems.

- **What are the reasons for implementing a blockchain in public administrations?**
  It should be clarified that there is currently no necessity to use a blockchain for public administration. Various advantages and disadvantages are discussed in this thesis. Due to high complexity regarding data protection and due to legal regulations, a final result cannot yet be conclusively clarified.

  That said, the advantages of blockchain in general and for public administrations should be mentioned. There is, on the one hand, the potential protection against SPOF. As generally, with security features, the need is hard to justify and even harder to demonstrate financially. However, the potential for faster processing, as demonstrated with AAA-me, could provide a financial justification for using a blockchain. Until this point, the advantages, and disadvantages have been the same between the public administrations and the (private) business sector. With a focus on public administration, you have a federal system, at least in Germany. This division of areas of responsibility forms a decentralization that has been seen as inhibiting previous digitalization. If blockchain is conceptually adapted to meet the needs, this would be a way to digitally map federal structures and enable digitization. Blockchain technology also provides transparency, in addition to decentralization. The ability to trace would be of interest to law enforcement agencies, and would still comply with data protection requirements when current cryptographic systems such as Zero-Knowledge Proof (ZKP) are used. If it turns out that digital identity can be mapped well on a blockchain, this would be another opportunity for the sovereignty of the individual in a federal system while promoting the digitization of processes. This is accompanied by a rise in efficiency, which is derived from the digitization process itself and not solely through the blockchain.

  Furthermore, Conclusion 1 and 2 can be used as arguments in favor of using blockchain technology.

- **Is blockchain technology capable of enhancing democracy?**
  It is important to clarify at this point that this thesis can only outline the question due to its complexity. Since the focus of this thesis is on a technical implementation, and a democracy consists of many factors, it is not sufficient to answer the question in its entirety. Despite this, an attempt has been made to summarize the question with the findings from this thesis at this point. Nevertheless, due to the high complexity, further research must be undertaken, and an answer here can only be considered an assumption for this research.

  This research question has intersections with the question of what are the reasons for using a blockchain for public administration, which I have already answered above. The increase in efficiency of bureaucratic processes that could have occurred through a blockchain contributes to a better distribution of manpower and can stabilize the bureaucratic system as a whole. Transparency and traceability could increase trust in democratic processes and counter fraud and corruption. It has the potential to aid official and police investigations, as it is challenging to deny existing transactions.

  In addition to enhancing the efficiency of processes, election systems are mentioned with reference to blockchains. Blockchain technology would validate the digital identity and, with it, the authorization to vote. Through the digitization of the electoral process and the accompanying simplification, it is hoped to increase voter turnout again, with a simultaneous reduction in the cost of executions. These aspects promote direct democracy and could facilitate the implementation of referendums. However, there is a great risk of manipulation and technological barriers that could limit implementation to federal or state elections in Germany.

  Besides digital identities, the application of a token as a currency can still be considered. Energy costs and legal challenges also come into play here, making implementation complex and potentially less environmentally friendly.

  In conclusion, it is only assumed that democracy will be enhanced. The associated expenses associated with constructing a suitable infrastructure suggest that further investigation is only feasible in the distant future. Only when a system is considered secure in the context of computer security, should its adoption for large-scale administrations be considered. Proper implementation, regulation, and acceptance in society will be crucial.

- **What are the prerequisites for a successful blockchain implementation?**

  It is important to define what "successful" actually means. Since this definition is beyond the scope of this thesis, this section is limited to the partial results of this work. A first insight that can lead to failure is the incorrect conception of the interface to the ledger. If only a single instance is built for interaction, the risk of SPOF remains and the benefits of a blockchain are nullified. The operation of a project requires the utilization of a designated node or nodes, or the connection to multiple nodes through a system that ensures consensus in the event of conflicting information.

  Another essential aspect is to consider the need for a blockchain. As discussed in 6.4.1 Identifying the Need for Decentralized Approaches, there are several points that must be met in order for the use of a blockchain to make sense. These findings were summarized above in two conclusions (Conclusion 1 and Conclusion 2).

  The successful implementation of a Blockchain in your project also requires:

  – A clear definition of the use case, technological infrastructure, security, and legal and regulatory compliance are essential.
  – Interoperability with existing systems, scalability, community engagement, and stakeholder engagement.
  – The ability to access skilled professionals, ensure data quality and management, and establish a clear governance framework.
  – Economic considerations, environmental sustainability, ease of use, extensive testing and quality assurance, and ethical considerations.

  It is critical to consider all of these factors, along with a careful understanding of the specific needs and challenges of the project, in order to reap the full benefits of blockchain and minimize potential risks.

Chapter 7

Outlook and Recommendations for Action

In this thesis, AAA-me, a framework that can implement a Remote Authentication Dial-In User Service (RADIUS) protocol on a blockchain, was developed. Thanks to external libraries and frameworks, the framework is able to interact with multiple blockchains and database systems. Furthermore, an overview was provided of the state of research regarding blockchain implementations. Even though the state of the art was given in general terms, there was a focus on public administration, as this is where an application was studied. However, it is not possible to conclude that developments are complete. Other areas beyond technology have not yet been considered in depth, and further consideration is also needed from a technological perspective. This section discusses a few possible points for further research and attempts to provide a recommendation for action if blockchain projects are planned.

7.1 Regulatory and Social Factors

This work emphasized the technical implementation as its primary focus. Although the economic, environmental, and regulatory considerations were considered, there was a lack of depth in order to not exceed the scope. Nevertheless, these aspects are crucial and must be considered for further investigation. Especially if the government wants to solve the problem.
Social factors affect the possibilities of addressing adoption for end users. If I map a system digitally but do not find users, the project can be considered a failure [134]. To avoid this, this point must be examined in detail. The focus here is on aspects such as costs for operators, as well as possible simplifications for the end user.
Many critics of blockchain and Distributed Ledger Technology (DLT) systems come from the poor environmental balance. But as we have read, the issue is more complex and subsuming it to Bitcoin (BTC) with its consensus process Proof-of-Work (PoW) is an oversimplification. Extensive research on BTC, but also other systems can already be found. However, the chosen architecture ultimately determines the actual cost. Therefore, consensus methods for this architecture must be investigated and compared. Quite exciting here is the comparison of PoW and Proof-of-Stake (PoS) which is led by the conversion with Ethereum (ETH) increasingly also in popular media. Alternatives to both systems also exist.

There is a significant increase in complexity expected from regulation. It is not only complex that the task itself is complex. Regulators are charged with the responsibility of devising regulations that guarantee the safeguarding of participants, while simultaneously incorporating the promotion of innovation. This immense task is constantly subject to debate and evolution. Digitization complicates these tasks by creating overlapping regulatory layers through international operations. How can a transaction be handled when regulations contradict each other? It is imperative to derive and adapt lessons from these experiences for the operation of blockchain infrastructures.

The complexity of setting up a WLAN system in a public administration (e.g., municipalities) is demonstrated using the example of legal aspects. For this purpose, a guideline from the federal state of Hesse was used as an example that outlines the establishment of public WLAN infrastructure [109]. As a federal state of the Federal Republic of Germany, the complexity due to international operations does not seem to apply here. However, European law can at least come into play in this case as well, which we will see shortly. Thus, the complexity on the international stage can be guessed. This is not intended to be a legal briefing, and it does not set out to be exhaustive. The example is only intended to illustrate that legal complexity exists.
Legal aspects of setting up a WLAN in the state of Hesse in Germany according to [109]:

- Public Procurement Law (Vergaberecht): Requirements are determined by certain threshold values. These are described in § 97 ff. of the Act against Restraints of Competition (GWB - Gesetz gegen Wettbewerbsbeschränkungen) [3], as well as additional procurement regulations. If the threshold value is exceeded, a Europe-wide invitation to tender must be conducted.

- State Aid Law (Beihilferecht): It must be checked whether the construction is in accordance with EU state aid law (EU-Beihilferecht). State aid law is shaped by Article 107 et seq. of the Treaty on the Functioning of the European Union (TFEU) (Vertrag über die Arbeitsweise der Europäischen Union, AEUV) [11], as well as the related regulations, directives, and communications.

- Municipal Economic Law (Kommunales Wirtschaftsrecht): This regulates whether the participation of public administration in economic activities is permitted. For Hesse, the provisions of § 121 et seq. of the Hessian Municipal Code (Hessische Gemeindeordnung, HGO) [2] contain such guidelines. 'In particular, a municipality may only engage in economic activities if, among other things, the public purpose justifies the activity.' [109].

- Civil Liability Issues (Zivilrechtliche Haftungsfragen): "Bei der Bereitstellung von WLAN-Diensten stellen sich zivilrechtliche Haftungsfragen. Relevant wird dies für die Kommune in aller Regel nicht in Netzbetreiber- oder Kooperationsmodellen, sondern dann, wenn sie selbst als Anbieter eines WLAN-Hotspots auftritt. Mit der mehrmaligen Änderung des Telemediengesetzes (TMG) hat der Gesetzgeber die Haftungsrisiken für Anbieter inzwischen wesentlich begrenzt und insbesondere die sogenannte Störerhaftung abgeschafft[1]" [109].

- Telecommunications Law (Telekommunikationsrecht): "Beim Betrieb eines WLAN sind außerdem Regelungen des Telekommunikationsrechts, die sich insbesondere im Telekommunikationsgesetz (TKG) finden, zu beachten. Art und Umfang der Pflichten hängen auch hier wesentlich von dem jeweiligen Umsetzungs - modell ab[2]"[109].

The legal tasks to be handled are an overview of what can be considered for the provision of a WLAN infrastructure. If one now provides a RADIUS system via the blockchain for providing Authentication, Authorization and Accounting (AAA) features, these areas also access those of the blockchain. Considerations here can thus also affect and condition the architecture. Thus, the importance of clarifying these issues is significant.

[1]The provision of WLAN services raises questions of liability under civil law. As a rule, this is not relevant for the municipality in network operator or cooperation models, but when it itself acts as the provider of a WLAN hotspot. With the repeated amendment of the German Telemedia Act (TMG), the legislator has now significantly limited the liability risks for providers and, in particular, abolished the so-called "Stoererhaftung" (Breach of Duty of Care).

[2]When operating a WLAN, the regulations of telecommunications law, which can be found in particular in the Telecommunications Act (TKG), must also be observed. The nature and scope of the obligations here also depend to a large extent on the respective implementation model.

7.1.1 AAA-me Integration in HZD and Public Administrations

With Fisbox, there is a good interface for how AAA-me can be integrated into the infrastructure of Hessian Central Office for Data Processing (German: Hessische Zentrale für Datenverarbeitung) (HZD). As the IT service provider for the federal state of Hesse, HZD develops, operates and maintains IT applications, offers IT consulting services and supports the digital transformation of the Hessian administration. Thus, it lends itself to continuing research and development, as integration into HZD systems favors integration for public administrations.
However, AAA-me is strongly designed for AAA functions. With Oracle systems, much more general solutions can be provided. Therefore, this aspect should not be neglected and it is worth doing further research here. Also, results could be integrated here, through the modularity of AAA-me with respect to blockchains and DLTs. The feature set could thus be extended from pure AAA functions to more general interactions.

7.2 Outlook for AAA-me Development

The AAA-me project demonstrated a framework capable of implementing the RADIUS protocol on the blockchain. Even if potentials are already being investigated and explored, the development still needs further time to become a program that can run without errors. In addition to the advancements in stability and security, there are additional aspects that will be elucidated in bullet points, which can enhance the functionality of AAA-me and mitigate any future security concerns.

- AAA-me can be quickly extended with further DLTs due to its modularity. This may become necessary insofar as the developments of already used technology are not target-oriented for the own projects or are progressing too slowly. Furthermore, security concerns could pose a threat to the own systems due to advancement. These dangers exist because most actively used blockchain technologies are developed open source, and therefore the target direction cannot be specified. However, the modularity mentioned above makes it possible to integrate new technologies, possibly in-house developments, quickly and easily.

- Integrated blockchain solutions have respective advancements for themselves that should be considered in the focus of AAA-me's outlook.

  - Ethereum (ETH): ETH has undergone extensive further developments during the time of this thesis. The biggest one is the change from PoW to PoS. Even if this change were successful, further large updates are planned. For instance, the implementation of sharding, wherein parallel chains are managed. The planned updates are only the beginning of the many updates that can be rolled out. Each update has a significant impact on the stability and security of the ledger, necessitating a diligent monitoring and evaluation process. If necessary, a change to another technology should be considered.
  - Bitcoin (BTC): The BTC is considered the most decentralized project. As a result, it is in high demand for distributed update mechanisms. However, these "democratic" processes take a lot of time and due to the large number of interested parties, even highly controversial updates are discussed and incorporated. For instance, consider the Ordinals protocol, which enables the creation of tokens that resemble Non-Fungible Token (NFT) on the ledger. This presents potential hazards for AAA-me and necessitates continuous monitoring to enable intervention. The high level of decentralization comes at a high cost, which is reflected in transaction fees. To counter this, there is a lot of research for so-called second-layer solutions. These can be exciting for AAA-me, especially if they can be applied to further DLTs, such as the Lightning Network.

  – Hyperledger Indy: Indy is specifically designed for digital identities. As such, the system inherently offers different functionalities than the other two supported blockchains. However, the high level of functionality cannot currently be used by AAA-me, as the functionality is used purely in the storage of information. Other functions of Indy are, for example, the inherent support of creating and validating attestations or also the establishment of Peer-to-Peer (P2P) channels to avoid the tapping of AAA information during communication.

- InterPlanetary File System (IPFS): Blockchain technologies are prone to the transparency problem. This becomes a challenge when individual details are required for relevant actions. Such processes could occur with digital identities if there is a Zero-Knowledge Proof (ZKP) of a birthday, for example. To reduce the dangers of the transparency problem, external data storage could be used. However, classic cloud and web servers again have the problem of centralization. A suitable solution here could be redundant encryption of data (à la RAID system) or the IPFS. See also chapter 2.2.5 IPFS and other Off-Chain-Databases Solutions. With such a storage solution, data can be stored redundantly across different systems. The retrieval of the data is therefore ensured by the redundancy. Encryption helps prevent unauthorized access to this data.

- Quantum computing is a challenge that affects all programs that employ asymmetric cryptography. The implementation of these procedures may pose a threat to the security of current protocols. By applying asymmetrical cryptography, such as signatures, the danger is great that such a quantum computer will be able to compute the secret keys and thus steal the appropriate identities. Therefore, with focus on the corresponding supported DLTs, the respective efforts to protect against quantum computers must be investigated and implemented quickly.

- The primary focus was on the authentication process on various systems. The integration of the two additional functions from the AAA protocols is a feasible proposition. Nevertheless, for scientific reasons, they should still be realized and tested. It is irreplaceable to test in the same laboratory environment as for the authentications.

- Programming is complex and, especially in non-IT areas, an arduous task to solve. Hence, efforts are being made to develop automated systems that can streamline programming in a structured manner, thereby enabling employees with limited programming expertise to make even minor adjustments. Similarly difficult is the accomplishment for Decentralized Applications (dApps). Due to decentralization, paradigms need to be adapted. A component with a low-code or no-code platform could allow users to create or customize applications, often through visual interfaces and drag-and-drop designs. This could potentially facilitate the programming of smart contracts without requiring any programming expertise, thereby significantly augmenting the functionality of AAA-me.

- The utilization of cryptographic keys necessitates the establishment of a prudent and secure environment for their generation and management. These aspects of hardware security were not considered in this thesis. This is because this consideration can be well externalized and addressed in further research and the results integrated into AAA-me. Outside AAA-mes Progressive Web App (PWA) functionality, hardware security should not have any impact on the functioning of AAA-me. This is because the use of the key is modular and can accordingly be ported to secure hardware in a modular fashion. However, in the realization of PWAs it is much more difficult and requires new knowledge in the field of security in research. Keeping track of these developments is indispensable for using PWAs for AAA-me. If it turns out that it is not securely possible to manage the keys, then this functionality must be questioned.

- Considering the recommendation to use certificates to realize the RADIUS protocol, or digital identities in general, on a blockchain, another aspect comes to light that can be realized by these certificates.

Passwordless authentication aims to provide authentication with comparable or, in some cases, superior security. It is possible to create certificates that can be checked by challenge-response procedures and access granted according to the validation. WebAuthn (Web Authentication) provides an example of authentication without a password. This feature facilitates authentication with a web service without the requirement of a password. Biometric data, such as fingerprints or facial recognition, security keys, or other authenticators, are used for the login. These items are used to create a token or certificate for the challenge-response procedure mentioned above.

- In this thesis, little attention has been given to Self-Sovereign Identity (SSI). Nonetheless, this technology has garnered significant interest among public administrations. SSI is a concept of digital identities that seeks to give users control over their data. Various aspects are considered to fulfill this goal. One of the factors that contributes to this is the resistance to correlation in various validation or authentication procedures. But also P2Ptunnels, for a secure communication in validation processes or a DLT especially designed for identity. However, it is not mandatory to use a DLT for SSI systems. There are 3 main points to the SSI model. First, the individual stores his identity data and certificates locally on a device. These can be shared with other entities over the network as the second point, or passed as ZKP for validation. The third point then takes effect, that the individual can decide what data is passed. The ZKPs helps, for example, to verify the age of majority without having to transfer the date of birth itself.

- AAA-me is able to extend Single Sign-On (SSO) systems with RADIUS functionalities. Essential here is cross-network authentication. If a company is equipped with palm vein scanners or similar physical sensors, corresponding certificates and access to databases could be extended so that access permission could be granted not only for a person in the digital system, but for all the hardware that comes with it. The work cell phone and work laptop, like the person, would have access to the digital and physical infrastructure as well.

- All laboratory tests for AAA-me have to be extended by further components. Among them belongs further functionalities of AAA-me, but also extensions of network systems. So should be thought about RADIUS settings, which perform more than only one hop. Also, larger and more widely branched networks, as they are found on the Internet, can be considered. Subsequently, the laboratory environment should be ported to the cloud in similar settings and the times measured there.

Bibliography

[1] *"Bitcoin Is Dead" - The #1 Database of Notable Bitcoin Skeptics*. en. URL: https://buybitcoinworldwide.com/bitcoin-is-dead/ (visited on 2023-06-05).

[2] *§ 121 HGO - Wirtschaftliche Betätigung*. de. URL: https://gesetze.io/gesetze/he/hgo/121 (visited on 2023-07-02).

[3] *§ 97 GWB - Einzelnorm*. URL: https://www.gesetze-im-internet.de/gwb/__97.html (visited on 2023-07-02).

[4] Fabian A. Scherschel. *Dezentrale oder zentrale Datenverarbeitung?* de. Apr. 2020. URL: https://www.heise.de/hintergrund/Corona-Tracking-Wie-Contact-Tracing-Apps-funktionieren-was-davon-zu-halten-ist-4709903.html (visited on 2022-10-19).

[5] B. Aboba and D. Simon. *PPP EAP TLS Authentication Protocol*. Request for Comments 2716. Internet Engineering Task Force, Oct. 1999. URL: https://tools.ietf.org/html/rfc2716.

[6] Dr. Bernard D. Aboba and John Vollbrecht. *Proxy Chaining and Policy Implementation in Roaming*. RFC 2607. June 1999. DOI: 10.17487/RFC2607. URL: https://www.rfc-editor.org/info/rfc2607.

[7] *About – Hyperledger Foundation*. en-US. URL: https://www.hyperledger.org/about (visited on 2023-04-11).

[8] Hilary J Allen. „The 'Merge' did not Fix Ethereum". en. In: (Oct. 2022). (Visited on 2023-08-04).

[9] Jake Archibald. *Introducing Background Sync*. en. 2017. URL: https://developer.chrome.com/blog/background-sync/ (visited on 2023-03-10).

[10] Archiveddocs. *How the Kerberos Version 5 Authentication Protocol Works: Logon and Authentication*. en-us. Oct. 2009. URL: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc772815(v=ws.10) (visited on 2023-03-03).

[11] *Art. 107 (ex-Artikel 87 EGV)*. URL: https://dejure.org/gesetze/AEUV/107.html (visited on 2023-07-02).

[12] Dan Ashmore. *Solana shuts down for eighth time this year*. en-GB. June 2022. URL: https://coinjournal.net/news/solana-reminds-me-of-my-broken-earphones/ (visited on 2022-11-14).

[13] Pierre-Louis Aublin, Sonia Ben Mokhtar, and Vivien Quéma. „RBFT: Redundant Byzantine Fault Tolerance". In: *2013 IEEE 33rd International Conference on Distributed Computing Systems*. ISSN: 1063-6927. July 2013, pp. 297–306. DOI: 10.1109/ICDCS.2013.53.

[14] BAMF, Fraunhofer FIT, and Universität Luxemburg. *Digitalisierung der Bescheinigungsprozesse im Asylverfahren mittels digitaler Identitäten*. de. Apr. 2021. URL: https://www.BAMF.de/SharedDocs/Anlagen/DE/Digitalisierung/blockchain-whitepaper-2021.html?nn=282388 (visited on 2023-08-29).

[15] Paul Baran. *On Distributed Communications: I. Introduction to Distributed Communications Networks*. en. Tech. rep. RAND Corporation, Jan. 1964. URL: https://www.rand.org/pubs/research_memoranda/RM3420.html (visited on 2022-10-16).

[16] Massimo Bartoletti and Livio Pompianu. „An Analysis of Bitcoin OP_RETURN Metadata“. en. In: *Financial Cryptography and Data Security*. Ed. by Michael Brenner et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 218–230. ISBN: 978-3-319-70278-0. DOI: 10.1007/978-3-319-70278-0_14.

[17] BBC. „Microsoft fined by European Commission over web browser“. en-GB. In: *BBC News* (Mar. 2013). URL: https://www.bbc.com/news/technology-21684329 (visited on 2022-10-19).

[18] Svenja Beller. *Greenpeace Magazin | Kann der Bitcoin grün werden?* de. June 2021. URL: https://www.greenpeace-magazin.de/aktuelles/kann-der-bitcoin-gruen-werden (visited on 2023-08-04).

[19] Eli Ben-Sasson et al. *Scalable, transparent, and post-quantum secure computational integrity*. Mar. 2018. URL: https://eprint.iacr.org/undefined/undefined (visited on 2023-08-03).

[20] Juan Benet. „IPFS-content addressed, versioned, P2P file system (DRAFT 3)“. In: *arXiv preprint arXiv:1407.3561* (2014).

[21] Christoph Bergmann. *Ethereum, Staking und der verflixte Zufall*. de-DE. Dec. 2022. URL: https://bitcoinblog.de/2022/12/05/ethereum-staking-und-der-verflixte-zufall/ (visited on 2023-08-06).

[22] Albrecht Beutelspacher, Heike B. Neumann, and Thomas Schwarzpaul. *Kryptografie in Theorie und Praxis - Mathematische Grundlagen für Internetsicherheit, Mobilfunk und elektronisches Geld*. 2. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2010. ISBN: 978-3-834-89631-5.

[23] Patrick Beuth. „»ID Wallet«: Was nach dem Fehlstart mit dem digitalen Führerschein passiert“. de. In: *Der Spiegel* (Oct. 2021). ISSN: 2195-1349. URL: https://www.spiegel.de/netzwelt/apps/id-wallet-was-nach-dem-fehlstart-mit-dem-digitalen-fuehrerschein-passiert-a-f4bc10bc-08ab-42b4-9325-5de5cdc66e05 (visited on 2023-08-29).

[24] Bharat Bhushan, G. Sahoo, and Amit Kumar Rai. „Man-in-the-middle attack in wireless and computer networking — A review“. In: *2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall)*. 2017, pp. 1–6. DOI: 10.1109/ICACCAF.2017.8344724.

[25] Monowar H. Bhuyan et al. „Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions“. In: *The Computer Journal* 57.4 (2014), pp. 537–556. DOI: 10.1093/comjnl/bxt031.

[26] Kemal Bicakci and Bulent Tavli. „Denial-of-Service attacks and countermeasures in IEEE 802.11 wireless networks“. en. In: *Computer Standards & Interfaces*. Specification, Standards and Information Management for Distributed Systems 31.5 (Sept. 2009), pp. 931–941. ISSN: 0920-5489. DOI: 10.1016/j.csi.2008.09.038. URL: https://www.sciencedirect.com/science/article/pii/S0920548908001438 (visited on 2023-03-22).

[27] Jonathan Bier. *The Blocksize War - Chapter 1 - First Strike*. en-US. Mar. 2021. URL: https://blog.bitmex.com/the-blocksize-war-chapter-1-first-strike/ (visited on 2023-08-04).

[28] Andreas Biørn-Hansen, Tim A. Majchrzak, and Tor-Morten Grønli. „Progressive Web Apps: The Possible Web-native Unifier for Mobile Development“. In: *International Conference on Web Information Systems and Technologies*. 2017.

[29] Anna Biselli. *Interview zu ID Wallet: Konzeptionell kaputt und ein riesiger Rückschritt*. de-DE. Oct. 2021. URL: https://netzpolitik.org/2021/interview-zu-id-wallet-konzeptionell-kaputt-und-ein-riesiger-rueckschritt/ (visited on 2023-08-29).

[30] . *Bitcoin (BTC) statistics - Price, Blocks Count, Difficulty, Hashrate, Value*. en. URL: https://bitinfocharts.com/bitcoin/ (visited on 2023-05-12).

[31] Bitcoin-Wiki-Team. *Script - Bitcoin Wiki*. 2021. URL: https://en.bitcoin.it/wiki/Script (visited on 2022-08-23).

[32] *bitcoin/bips: Bitcoin Improvement Proposals*. URL: https://github.com/bitcoin/bips (visited on 2023-08-04).

[33] *bitcoin/bitcoin: Bitcoin Core integration/staging tree*. URL: https://github.com/bitcoin/bitcoin (visited on 2023-08-04).

[34] Bitkom. „Bestandsaufnahme: Ein Jahr Blockchain Strategie der Bundesregierung“. de. In: *Infopapier* (2020). URL: https://www.bitkom.org/sites/main/files/2020-09/200928_umsetzungsstand_blockchain-strategie.pdf (visited on 2023-06-13).

[35] Trail of Bits. *Are blockchains decentralized?* en-US. June 2022. URL: https://blog.trailofbits.com/2022/06/21/are-blockchains-decentralized/ (visited on 2023-07-28).

[36] *blockscout/docker-compose at master · blockscout/blockscout*. en. 2022. URL: https://github.com/blockscout/blockscout (visited on 2023-02-07).

[37] Blockscout-Team. *ENV Variables*. en. URL: https://docs.blockscout.com/for-developers/information-and-settings/env-variables (visited on 2023-02-07).

[38] Blockstream. *Bitcoin Explorer*. Jan. 2009. URL: https://blockstream.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f (visited on 2022-08-16).

[39] Blockstream. *Bitcoin Explorer BTC Transaction für Appendix*. July 2017. URL: https://blockstream.info/tx/b657e22827039461a9493ede7bdf55b01579254c1630b0bfc9185ec564fc05ab (visited on 2022-08-17).

[40] C. Blundo and M. Roe. *Generic Security Service Application Program Interface Version 2, Update 1*. Request for Comments 2743. Internet Engineering Task Force, Jan. 2000. URL: https://tools.ietf.org/html/rfc2743.

[41] BNB. *What is Binance Smart Chain?* en. URL: https://www.binance.com/en/news/top/1127704 (visited on 2022-10-19).

[42] Hanno Böck. *ID Wallet: Digitaler Führerschein nicht mehr im Appstore - Golem.de*. de-DE. Sept. 2021. URL: https://www.golem.de/news/id-wallet-digitaler-fuehrerschein-nicht-mehr-im-appstore-2109-159950.html (visited on 2023-08-29).

[43] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/info/rfc5280.

[44] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/info/rfc5280.

[45] Sebastian Brenza, Andre Pawlowski, and Christina Pöpper. „A Practical Investigation of Identity Theft Vulnerabilities in Eduroam". In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. WiSec '15. New York, New York: Association for Computing Machinery, 2015. ISBN: 9781450336239. DOI: 10.1145/2766498.2766512. URL: https://doi.org/10.1145/2766498.2766512.

[46] Eric Brewer. „CAP twelve years later: How the "rules" have changed". In: *Computer* 45.2 (2012), pp. 23–29. DOI: 10.1109/MC.2012.37.

[47] Alberto Attilio Brincat et al. „On the use of Blockchain technologies in WiFi networks". en. In: *Computer Networks* 162 (Oct. 2019), p. 106855. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2019.07.011. URL: https://www.sciencedirect.com/science/article/pii/S1389128619306073 (visited on 2022-05-03).

[48] *Deutscher Bundestag - Weg der Gesetzgebung*. de. 2022. URL: https://www.bundestag.de/parlament/aufgaben/gesetzgebung_neu/gesetzgebung/weg-255468 (visited on 2022-12-02).

[49] Peter Bundschuh. *Einführung in die Zahlentheorie*. 6. Aufl. Berlin Heidelberg: Springer-Verlag, 1998. ISBN: 978-3-540-76490-8.

[50] George Burlakov. *(26) Evaluating the Top 5 EVM-Compatible Blockchains: A Comprehensive Comparison | LinkedIn*. Mar. 2023. URL: https://www.linkedin.com/pulse/evaluating-top-5-evm-compatible-blockchains-george-burlakov/ (visited on 2023-08-04).

[51] W E Burr et al. *Electronic authentication guideline*. en. Tech. rep. NIST SP 800-63-1. Edition: 0. Gaithersburg, MD: National Institute of Standards and Technology, 2011, NIST SP 800–63–1. DOI: 10.6028/NIST.SP.800-63-1. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-63-1.pdf (visited on 2023-03-05).

[52] William E. Burr et al. *Electronic Authentication Guideline*. en. Tech. rep. NIST SP 800-63-2. National Institute of Standards and Technology, Nov. 2013, NIST SP 800–63–2. DOI: 10.6028/NIST.SP.800-63-2. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-2.pdf (visited on 2023-03-03).

[53] Randy Bush and Rob Austein. *The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1*. RFC 8210. Sept. 2017. DOI: 10.17487/RFC8210. URL: https://www.rfc-editor.org/info/rfc8210.

[54] Vitalik Buterin. *Hard Forks, Soft Forks, Defaults and Coercion*. Mar. 2017. URL: https://vitalik.ca/general/2017/03/14/forks_and_markets.html (visited on 2023-08-04).

[55] Vitalik Buterin. *The Meaning of Decentralization*. en. Feb. 2017. URL: https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274 (visited on 2022-10-19).

[56] Giulio Caldarelli and Joshua Ellul. „The Blockchain Oracle Problem in Decentralized Finance—A Multivocal Approach". In: *Applied Sciences* (2021).

[57] Brian Campbell, Chuck Mortimore, and Michael Jones. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*. RFC 7522. May 2015. DOI: 10.17487/RFC7522. URL: https://www.rfc-editor.org/info/rfc7522.

[58] Cardano. *Cardano is a decentralized public blockchain and cryptocurrency project and is fully open source.* en. URL: https://cardano.org/ (visited on 2022-10-19).

[59] Miguel Castro and Barbara Liskov. „Practical Byzantine fault tolerance“. In: *Proceedings of the third symposium on Operating systems design and implementation*. OSDI ’99. USA: USENIX Association, Feb. 1999, pp. 173–186. ISBN: 978-1-880446-39-3. (Visited on 2023-08-04).

[60] *Chainlink: The Industry-Standard Web3 Services Platform*. en. URL: https://chain.link/ (visited on 2023-06-09).

[61] . *Chainlist*. 2023. URL: https://chainlist.org/ (visited on 2023-01-12).

[62] Clem Chambers. *Celsius, The Trouble With Crypto*. en. Section: Investing. Oct. 2022. URL: https://www.forbes.com/sites/investor/2022/10/18/celsius-the-trouble-with-crypto/ (visited on 2023-07-28).

[63] Jenny Chang. *55 Important Password Statistics You Should Know: 2023 Breaches & Reuse Data*. en. Apr. 2021. URL: https://financesonline.com/password-statistics/ (visited on 2023-03-09).

[64] Zhe Chen et al. „Modeling of Man-in-the-Middle Attack in the Wireless Networks“. In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. ISSN: 2161-9654. Sept. 2007, pp. 2255–2258. DOI: 10.1109/WICOM.2007.562.

[65] Vincent Cheval et al. „Hash Gone Bad: Automated discovery of protocol attacks that exploit hash function weaknesses“. In: *IACR Cryptol. ePrint Arch.* 2022 (2022), p. 1314.

[66] *Client Diversity | Ethereum*. en. URL: https://clientdiversity.org (visited on 2023-08-07).

[67] ELECTRIC COIN COMPANY. *Zcash - How It Works*. en-US. 2021. URL: https://z.cash/technology/ (visited on 2022-08-18).

[68] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. „A Survey of Man In The Middle Attacks“. In: *IEEE Communications Surveys & Tutorials* 18.3 (2016), pp. 2027–2051. DOI: 10.1109/COMST.2016.2548426.

[69] Danny Cooper et al. „On the risk of misbehaving RPKI authorities“. In: *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks* (2013).

[70] Gabriel-Cătălin CRISTESCU and Victor CROITORU. „Volumetric Distributed Denial-of-Service and Session Replay Attacks-Resistant AAA-RADIUS Solution Based on EAP and LDAP“. In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. 2020, pp. 1–4. DOI: 10.1109/ISETC50328.2020.9301119.

[71] Gabriel-Cătălin Cristescu and Victor Croitoru. „Spoofed Packet Injection Attack-Resistant AAA-RADIUS Solution Based on LDAP and EAP“. In: *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*. 2021, pp. 1–4. DOI: 10.1109/ISSCS52333.2021.9497398.

[72] Mike Dalton. *Solana Blockchain Halted for Four Hours*. en-US. June 2022. URL: https://cryptobriefing.com/solana-blockchain-halted-for-eight-hours/ (visited on 2022-10-19).

[73] Erik Daniel and Florian Tschorsch. „IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks“. In: *IEEE Communications Surveys & Tutorials* 24.1 (Jan. 2022), pp. 31–52. ISSN: 1553-877X. DOI: 10.1109/COMST.2022.3143147.

[74] Darreonna Davis. *What Happened To FTX? The Crypto Exchange Fund’s Collapse Explained*. en. Section: Business. June 2023. URL: https://www.forbes.com/sites/darreonnadavis/2023/06/02/what-happened-to-ftx-the-crypto-exchange-funds-collapse-explained/ (visited on 2023-07-28).

[75] Alex De Vries. „Cryptocurrencies on the road to sustainability: Ethereum paving the way for Bitcoin“. en. In: *Patterns* 4.1 (Jan. 2023), p. 100633. ISSN: 26663899. DOI: 10.1016/j.patter.2022.100633. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666389922002653 (visited on 2023-08-04).

[76] Alan DeKok. *Deprecating RADIUS/UDP and RADIUS/TCP*. Internet-Draft draft-dekok-radext-deprecating-radius-01. Work in Progress. Internet Engineering Task Force, Mar. 2023. 16 pp. URL: https://datatracker.ietf.org/doc/draft-dekok-radext-deprecating-radius/01/.

[77] Alan DeKok and Jouni Korhonen. *Dynamic Authorization Proxying in the Remote Authentication Dial-In User Service (RADIUS) Protocol*. RFC 8559. Apr. 2019. DOI: 10.17487/RFC8559. URL: https://www.rfc-editor.org/info/rfc8559.

[78] Qi Deng, Yun Wang, and Shaobo Ji. „Design Science Research in Information Systems: A Systematic Literature Review 2001-2015“. In: *International Conference on Information Resources Management*. 2017.

[79] *Die versteckten Kosten der Microsoft CA eBook Download*. de. Dec. 2022. URL: https://www.globalsign.com/de-de/lp/versteckten-kosten-microsoft-ca (visited on 2023-04-07).

[80] Digiconomist. *Bitcoin Energy Consumption Index*. en-US. Aug. 2023. URL: https://digiconomist.net/bitcoin-energy-consumption/ (visited on 2023-08-04).

[81] digiconomist. *Bitcoin Electronic Waste Monitor*. en-US. Sept. 2019. URL: https://digiconomist.net/bitcoin-electronic-waste-monitor/ (visited on 2022-03-14).

[82] *Digital Public Services in the Digital Economy and Society Index | Shaping Europe's digital future*. en. July 2022. URL: https://digital-strategy.ec.europa.eu/en/policies/desi-digital-public-services (visited on 2023-08-28).

[83] Robert F. Dittmar and Di (Andrew) Wu. *Initial Coin Offerings Hyped and Dehyped: An Empirical Examination*. en. SSRN Scholarly Paper. Rochester, NY, Mar. 2019. DOI: 10.2139/ssrn.3259182. URL: https://papers.ssrn.com/abstract=3259182 (visited on 2023-08-07).

[84] . *Django*. en. 2005. URL: https://www.djangoproject.com/ (visited on 2023-04-09).

[85] Anke Domscheit. *Quo Vadis, Verwaltungsdigitalisierung? Das Onlinezugangsgesetz als Megafail Eine Kleine Anfrage mit entmutigenden Ergebnissen - Anke Domscheit-Berg*. de-DE. Nov. 2022. URL: https://mdb.anke.domscheit-berg.de/2022/11/quo-vadis-verwaltungsdigitalisierung-das-onlinezugangsgesetz-als-megafail-eine-kleine-anfrage-mit-entmutigenden-ergebnissen/ (visited on 2023-08-28).

[86] Michael M. Dowling. *Is Non-fungible Token Pricing Driven by Cryptocurrencies?* en. SSRN Scholarly Paper. Rochester, NY, Mar. 2021. DOI: 10.2139/ssrn.3815093. URL: https://papers.ssrn.com/abstract=3815093 (visited on 2023-08-07).

[87] DWDS. *DWDS – Digitales Wörterbuch der deutschen Sprache*. de. URL: https://www.dwds.de/wb/zentralisieren (visited on 2022-10-19).

[88] DWDS. *DWDS – Digitales Wörterbuch der deutschen Sprache*. de. URL: https://www.dwds.de/wb/dezentral (visited on 2022-10-19).

[89] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. „Impact of Man-In-The-Middle Attacks on Ethereum“. In: *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*. 2018, pp. 11–20. DOI: 10.1109/SRDS.2018.00012.

[90] *Ethereum Chain Full Sync Data Size*. URL: https://ycharts.com/indicators/ethereum_chain_full_sync_data_size (visited on 2023-05-12).

[91] *Installing Geth*. en. URL: https://geth.ethereum.org/docs/getting-started/installing-geth (visited on 2023-01-10).

[92] *Ethereum Name Service*. URL: https://ens.domains (visited on 2023-05-30).

[93] *Ethereum Whitepaper*. en. Aug. 2023. URL: https://ethereum.org (visited on 2023-08-07).

[94] etherscan.io. *ChainLink Token (LINK) Token Tracker | Etherscan*. en. URL: https://etherscan.io/token/0x514910771af9ca656af840dff83e8264ecf986ca (visited on 2023-06-12).

[95] ETSI. *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Diameter applications; 3GPP specific codes and identifiers (3GPP TS 29.230 version 11.7.0 Release 11)*. en-gb. URL: https://www.etsi.org/ (visited on 2023-03-15).

[96] Victor Fajardo et al. *Diameter Base Protocol*. Request for Comments RFC 6733. Num Pages: 152. Internet Engineering Task Force, Oct. 2012. DOI: 10.17487/RFC6733. URL: https://datatracker.ietf.org/doc/rfc6733 (visited on 2023-03-15).

[97] Frank Felden and Thilo Zelt. *Schneller, einfacher, nutzerorientierter – drei zentrale Forderungen an die Digitalisierung der deutschen Behördendienste*. en. June 2023. URL: https://www.bcg.com/publications/2023/germany-digital-government-citizen-survey (visited on 2023-08-28).

[98] Roy T. Fielding, Mark Nottingham, and Julian Reschke. *HTTP Semantics*. RFC 9110. June 2022. DOI: 10.17487/RFC9110. URL: https://www.rfc-editor.org/info/rfc9110.

[99] Santiago Figueroa-Lorenzo, Javier Añorga Benito, and Saioa Arrizabalaga. „Modbus Access Control System Based on SSI over Hyperledger Fabric Blockchain“. eng. In: *Sensors (Basel, Switzerland)* 21.16 (Aug. 2021), p. 5438. ISSN: 1424-8220. DOI: 10.3390/s21165438.

[100] Hal Finney et al. *OpenPGP Message Format*. RFC 4880. Nov. 2007. DOI: 10.17487/RFC4880. URL: https://www.rfc-editor.org/info/rfc4880.

[101] Licia Florio and Klaas Wierenga. „Eduroam, providing mobility for roaming users“. In: *Proceedings of the EUNIS 2005 Conference, Manchester*. 2005.

[102] *Flutter - Build apps for any screen*. en. 2023. URL: //flutter.dev/ (visited on 2023-03-10).

[103] *Forks · bitcoin/bitcoin*. URL: https://github.com/bitcoin/bitcoin/forks (visited on 2023-08-04).

[104] Ethereum Foundation. *Sharding | ethereum.org*. 2022. URL: https://ethereum.org/en/upgrades/sharding/ (visited on 2022-11-28).

[105] J. Franks et al. *HTTP Authentication: Basic and Digest Access Authentication*. Request for Comments 2617. Internet Engineering Task Force, June 1999. URL: https://tools.ietf.org/html/rfc2617.

[106] *Gästeregistrierung - free-key #besafe*. de. URL: http://free-key.eu/news-digitale_dorflinde_verlaengerung.html (visited on 2023-05-18).

[107] freeRADIUS. *Modules*. URL: https://freeradius.org/modules/ (visited on 2023-03-23).

[108] . *freeRADIUS numbers*. 2018. URL: https://freeradius.org/about/#usage_statistics (visited on 2023-04-11).

[109] Dr. Matthias Freund and Klaus Eichler. „Errichtung öffentlicher WLAN-Netze“. de. In: (May 2023), p. 20. URL: https://redaktion.hessen-agentur.de/publication/2023/4061_Leitfaden_WLAN_2023_Web.pdf.

[110] Yanduo Fu et al. „Exploring the Security Issues of Trusted CA Certificate Management“. In: *Information and Communications Security*. Ed. by Debin Gao et al. Cham: Springer International Publishing, 2021, pp. 384–401. ISBN: 978-3-030-86890-1.

[111] A. Funk et al. *Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)*. Request for Comments 5281. Internet Engineering Task Force, Aug. 2008. URL: https://tools.ietf.org/html/rfc5281.

[112] Corinna Funke. „Digitization, fast and slow : comparing the creation of digital public services in Denmark, France and Germany“. en. Accepted: 2022-10-24T08:46:56Z. Thesis. European University Institute, 2022. DOI: 10.2870/401344. URL: https://cadmus.eui.eu/handle/1814/74971 (visited on 2023-08-28).

[113] Hisham S. Galal, Muhammad ElSheikh, and Amr M. Youssef. „An Efficient Micropayment Channel on Ethereum“. en. In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Ed. by Cristina Pérez-Solà et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 211–218. ISBN: 978-3-030-31500-9. DOI: 10.1007/978-3-030-31500-9_13.

[114] Xianyi Gao et al. „Forgetting of Passwords: Ecological Theory and Data“. In: *USENIX Security Symposium*. 2018.

[115] Jeff Garzik. *python-bitcoinrpc*. original-date: 2011-04-06T19:02:32Z. Mar. 2023. URL: https://github.com/jgarzik/python-bitcoinrpc (visited on 2023-04-11).

[116] GÉANT-Team. *Home - eduroam.org*. 2008. URL: https://eduroam.org/ (visited on 2022-04-26).

[117] Nathan George et al. *Announcing Hyperledger Aries, infrastructure supporting interoperable identity solutions! – Hyperledger Foundation*. en-US. May 2019. URL: https://www.hyperledger.org/blog/2019/05/14/announcing-hyperledger-aries-infrastructure-supporting-interoperable-identity-solutions (visited on 2023-04-11).

[118] Martin Georgiev et al. „The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software“. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS ’12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 38–49. ISBN: 9781450316514. DOI: 10.1145/2382196.2382204. URL: https://doi.org/10.1145/2382196.2382204.

[119] David Gerard. *If you want a scalable Ethereum blockchain, then P=NP*. en-GB. Apr. 2022. URL: https://davidgerard.co.uk/blockchain/2022/04/04/if-you-want-a-scalable-ethereum-blockchain-then-pnp/ (visited on 2023-06-06).

[120] Seth Gilbert and Nancy Lynch. „Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services“. In: *SIGACT News* 33.2 (June 2002), pp. 51–59. ISSN: 0163-5700. DOI: 10.1145/564585.564601. URL: https://doi.org/10.1145/564585.564601.

[121] PivIT Global. *RADIUS vs. TACACS+: Which AAA Protocol Should You Choose?* en. Mar. 2023. URL: https://info.pivitglobal.com/resources/radius-vs.-tacacs-aaa-protocol (visited on 2023-03-15).

[122] Bitcoin Wiki Gmaxwell. *Script*. https://en.bitcoin.it/wiki/Script. 2018. (Visited on 2022-08-18).

[123] *Govdigital*. URL: https://www.govdigital.de/ (visited on 2023-08-29).

[124] Matthew Green. *Zero Knowledge Proofs: An illustrated primer*. en. Nov. 2014. URL: https://blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer/ (visited on 2023-08-03).

[125] Gideon Greenspan. *Avoiding the pointless blockchain project | MultiChain*. Nov. 2015. URL: https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/ (visited on 2023-06-06).

[126] George Gross et al. *Generic AAA Architecture*. Request for Comments RFC 2903. Num Pages: 26. Internet Engineering Task Force, Aug. 2000. DOI: 10.17487/RFC2903. URL: https://datatracker.ietf.org/doc/rfc2903 (visited on 2023-03-15).

[127] Sebastian Grüner. *Zeugnisse in der Blockchain: Kaputter Blödsinn mit Ansage - Golem.de*. de-DE. Feb. 2022. URL: https://www.golem.de/news/zeugnisse-in-der-blockchain-kaputter-bloedsinn-mit-ansage-2202-163040.html#cmts (visited on 2023-08-29).

[128] Akhil Gupta and Rakesh Kumar Jha. „Security threats of wireless networks: A survey“. In: *International Conference on Computing, Communication & Automation*. 2015, pp. 389–395. DOI: 10.1109/CCAA.2015.7148407.

[129] Harry Haramis. *Council Post: Public Vs. Private: Unlocking The Full Potential Of Public Key Infrastructure*. en. Section: Innovation. Dec. 2021. URL: https://www.forbes.com/sites/forbestechcouncil/2021/12/08/public-vs-private-unlocking-the-full-potential-of-public-key-infrastructure/ (visited on 2023-04-07).

[130] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012. DOI: 10.17487/RFC6749. URL: https://www.rfc-editor.org/info/rfc6749.

[131] Nicola Hauptmann. *Was macht govdigital?* de. Mar. 2023. URL: https://www.egovernment.de/was-macht-govdigital-a-e85ab61f4577124decc7f5686de50fcb/ (visited on 2023-08-29).

[132] heise online heise. *DNS-Panne: heise.de landet bei 1&1 im Copyright-Filter*. de. Mar. 2023. URL: https://www.heise.de/news/DNS-Panne-heise-de-landet-bei-1-1-im-Copyright-Filter-7561803.html (visited on 2023-05-31).

[133] heise online heise. *NSA-Überwachungsskandal: Von PRISM, Tempora, XKeyScore und dem Supergrundrecht – was bisher geschah*. de. Aug. 2013. URL: https://www.heise.de/news/NSA-Ueberwachungsskandal-Von-PRISM-Tempora-XKeyScore-und-dem-Supergrundrecht-was-bisher-geschah-1931179.html (visited on 2023-04-24).

[134] heise online heise. *Ziele gänzlich verfehlt: Bundesrechnungshof rügt De-Mail-Fiasko*. de. Dec. 2021. URL: https://www.heise.de/news/Ziele-gaenzlich-verfehlt-Bundesrechnungshof-ruegt-De-Mail-Fiasko-6281540.html (visited on 2023-07-02).

[135] Pavol Helebrandt et al. „Blockchain Adoption for Monitoring and Management of Enterprise Networks“. In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. Nov. 2018, pp. 1221–1225. DOI: 10.1109/IEMCON.2018.8614960.

[136] Thomas Hepp et al. „Exploring Potentials and Challenges of Blockchain-based Public Key Infrastructures“. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2019, pp. 847–852. DOI: 10.1109/INFCOMW.2019.8845169.

[137] Hien Do Hoang, Phan The Duy, and Van-Hau Pham. „A Security-Enhanced Monitoring System for Northbound Interface in SDN using Blockchain". In: *Proceedings of the Tenth International Symposium on Information and Communication Technology*. SoICT 2019. New York, NY, USA: Association for Computing Machinery, Dec. 2019, pp. 197–204. ISBN: 978-1-4503-7245-9. DOI: 10.1145/3368926.3369709. URL: https://doi.org/10.1145/3368926.3369709 (visited on 2022-05-05).

[138] Martin Holland. *Bitcoin: Forscher finden Kinderpornographie in der Blockchain*. de. Mar. 2018. URL: https://www.heise.de/news/Bitcoin-Forscher-finden-Kinderpornographie-in-der-Blockchain-4000693.html (visited on 2023-07-28).

[139] *Home - Django REST framework*. URL: https://www.django-rest-framework.org/ (visited on 2023-04-11).

[140] Matthias Horst et al. „Breaking PPTP VPNs via RADIUS Encryption". en. In: *Cryptology and Network Security*. Ed. by Sara Foresti and Giuseppe Persiano. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 159–175. ISBN: 978-3-319-48965-0. DOI: 10.1007/978-3-319-48965-0_10.

[141] SiteGround Web Hosting. *What is DNS propagation and why does it take so long?* en. URL: https://www.siteground.com/kb/what_is_dns_propagation_and_why_it_takes_so_long/ (visited on 2022-08-08).

[142] Russ Housley. *Internationalization Updates to RFC 5280*. RFC 8399. May 2018. DOI: 10.17487/RFC8399. URL: https://www.rfc-editor.org/info/rfc8399.

[143] *How Does eduroam Work? - eduroam.org*. July 2018. URL: https://eduroam.org/how/ (visited on 2023-04-24).

[144] Jens J. Hunhevicz and Daniel M. Hall. „Do you need a blockchain in construction? Use case categories and decision framework for DLT design options". en. In: *Advanced Engineering Informatics* 45 (Aug. 2020), p. 101094. ISSN: 1474-0346. DOI: 10.1016/j.aei.2020.101094. URL: https://www.sciencedirect.com/science/article/pii/S147403462030063X (visited on 2023-06-06).

[145] *Hyperledger Indy – Hyperledger Foundation*. en-US. URL: https://www.hyperledger.org/use/hyperledger-indy (visited on 2023-04-11).

[146] . *HYPERLEDGER URSA*. original-date: 2018-11-19T22:03:24Z. Mar. 2023. URL: https://github.com/hyperledger/ursa (visited on 2023-04-11).

[147] IETF. *Secure Shell (SSH) Protocol Architecture*. 2006. URL: https://tools.ietf.org/html/rfc4251.

[148] *Internet kaputt? Wichtiger Sicherheitsmechanismus RPKI ausgehebelt*. de. Oct. 2022. URL: https://t3n.de/news/internet-kaputt-wichtiger-sicherheitsmechanismus-rkpi-ausgehebelt-1504511/ (visited on 2023-06-01).

[149] . *Introduction — web3.py 6.1.0 documentation*. Dokumentation. URL: https://web3py.readthedocs.io/en/stable/# (visited on 2023-04-11).

[150] IOTA. *IOTA*. en. URL: https://www.iota.org (visited on 2022-10-20).

[151] Xudong Jia et al. „A2 Chain: A Blockchain-Based Decentralized Authentication Scheme for 5G-Enabled IoT". en. In: *Mobile Information Systems* 2020 (Dec. 2020). Publisher: Hindawi, e8889192. ISSN: 1574-017X. DOI: 10.1155/2020/8889192. URL: https://www.hindawi.com/journals/misy/2020/8889192/ (visited on 2022-05-05).

[152] Xin Jiang et al. *A Blockchain-Based Authentication Protocol for WLAN Mesh Security Access*. 2019.

[153] Xin Jiang et al. „A Blockchain-Based Authentication Protocol for WLAN Mesh Security Access". In: *Computers, Materials & Continua* (2019). URL: https://www.techscience.com/cmc/v58n1/23011 (visited on 2022-04-26).

[154] Thomas Joos. *Mit Hyperledger Avalon Off-Chain-Daten nutzen*. de. July 2021. URL: https://www.blockchain-insider.de/mit-hyperledger-avalon-off-chain-daten-nutzen-a-1032200/?cmp=nl-436&uuid=884fff2fd5727259906d02841d4ec190 (visited on 2022-08-16).

[155] Austin Jul. *Research Summary: A Systematic Literature Review of Blockchain Governance - Governance and Coordination*. en. Section: Governance and Coordination. June 2022. URL: https://www.smartcontractresearch.org/t/research-summary-a-systematic-literature-review-of-blockchain-governance/1640 (visited on 2023-08-07).

[156] B. Kaliski. *PPP Challenge Handshake Authentication Protocol (CHAP)*. Request for Comments 1994. Internet Engineering Task Force, Aug. 1996. URL: https://tools.ietf.org/html/rfc1994.

[157] Dimitris Karakostas, Aggelos Kiayias, and Christina Ovezik. *SoK: A Stratified Approach to Blockchain Decentralization*. arXiv:2211.01291 [cs]. Nov. 2022. DOI: 10.48550/arXiv.2211.01291. URL: http://arxiv.org/abs/2211.01291 (visited on 2023-07-28).

[158] Bianca Kastl. *Degitalisierung: Effizienz ist ein konservatives Wesen*. de-DE. Aug. 2023. URL: https://netzpolitik.org/2023/degitalisierung-effizienz-ist-ein-konservatives-wesen/ (visited on 2023-08-28).

[159] Michal Kedziora et al. „Analysis of segregated witness implementation for increasing efficiency and security of the Bitcoin cryptocurrency". In: *Journal of Information and Telecommunication* 7.1 (Jan. 2023). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/24751839.2022.2122301, pp. 44–55. ISSN: 2475-1839. DOI: 10.1080/24751839.2022.2122301. URL: https://doi.org/10.1080/24751839.2022.2122301 (visited on 2023-08-04).

[160] Keyfactor and The Ponemon Institute. *The State of Machine Identity Management 2021*. Accessed: 2023-04-07. 2021. URL: https://f.hubspotusercontent40.net/hubfs/408597/Keyfactor%20White%20Papers/State-of-Machine-Identity-Management-Keyfactor-Ponemon-2021.pdf.

[161] Mohammad Khodaei and Panos Papadimitratos. „Efficient, Scalable, and Resilient Vehicle-Centric Certificate Revocation List Distribution in VANETs". en. In: *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (June 2018). Conference Name: WiSec '18: 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks ISBN: 9781450357319 Place: Stockholm Sweden Publisher: ACM, pp. 172–183. DOI: 10.1145/3212480.3212481. URL: https://dl.acm.org/doi/10.1145/3212480.3212481 (visited on 2023-04-12).

[162] BMWK-Bundesministerium für Wirtschaft und Klimaschutz. *Blockchain im Mittelstand*. de. May 2022. URL: https://www.bmwk.de/Redaktion/DE/Publikationen/Mittelstand/blockchain-im-mittelstand.html (visited on 2023-06-13).

[163] BMWK-Bundesministerium für Wirtschaft und Klimaschutz. *Nachhaltigkeit im Kontext der Blockchain-Technologie*. de. Apr. 2022. URL: https://www.bmwk.de/Redaktion/DE/Publikationen/Digitale-Welt/blockchain-nachhaltigkeit.html (visited on 2023-08-07).

[164] Dirk Knop. *Schlechtes Zeugnis für Zeugnisse in der Blockchain*. de. Feb. 2022. URL: https://www.heise.de/news/Schlechtes-Zeugnis-fuer-Zeugnisse-in-der-Blockchain-6370807.html (visited on 2023-08-29).

[165] *Koalitionsvertrag 2021*. de. Nov. 2021. URL: https://www.bundesregierung.de/breg-de/service/gesetzesvorhaben/koalitionsvertrag-2021-1990800 (visited on 2023-06-13).

[166] koe, Kurt Alonso, and Sarang Noether. *Zero to Monero - Second Edition*. en. Apr. 2020. URL: https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf (visited on 2022-08-18).

[167] Dr. Daniela Kühn et al. *Blockchain und Self-Sovereign Identity (SSI) in der Steuerverwaltung*. de-DE. June 2022. URL: https://www.lfst.bayern.de/aktuelles/details?tx_news_pi1%5Baction%5D=detail&tx_news_pi1%5Bcontroller%5D=News&tx_news_pi1%5Bnews%5D=30&cHash=f6ceb6d0a78e141f0590caa7329beb0c (visited on 2023-08-29).

[168] Yujin Kwon et al. *Impossibility of Full Decentralization in Permissionless Blockchains*. arXiv:1905.05158 [cs]. Sept. 2019. DOI: 10.48550/arXiv.1905.05158. URL: http://arxiv.org/abs/1905.05158 (visited on 2023-07-29).

[169] Leslie Lamport. „Time, clocks, and the ordering of events in a distributed system". In: *Communications of the ACM* 21.7 (July 1978), pp. 558–565. ISSN: 0001-0782. DOI: 10.1145/359545.359563. URL: https://dl.acm.org/doi/10.1145/359545.359563 (visited on 2023-08-04).

[170] Leslie Lamport, Robert Shostak, and Marshall Pease. „The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://dl.acm.org/doi/10.1145/357172.357176 (visited on 2023-08-04).

[171] lebing, theymos, and et. al. *Turing complete language vs non-Turing complete (Ethereum vs Bitcoin)*. 2014. URL: https://bitcointalk.org/index.php?topic=431513.0 (visited on 2022-08-24).

[172] Jiyeon Lee et al. „Pride and Prejudice in Progressive Web Apps: Abusing Native App-like Features in Web Applications". In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 1731–1746. ISBN: 9781450356930. DOI: 10.1145/3243734.3243867. URL: https://doi.org/10.1145/3243734.3243867.

[173] Pete LePage. *The Cache API: A quick guide*. en. 2020. URL: https://web.dev/cache-api-quick-guide/ (visited on 2023-03-10).

[174] Bingyu Li et al. „Certificate Transparency in the Wild: Exploring the Reliability of Monitors". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2505–2520. ISBN: 9781450367479. DOI: 10.1145/3319535.3345653. URL: https://doi.org/10.1145/3319535.3345653.

[175] Chunlei Li et al. „Trustroam: A Novel Blockchain-Based Cross-Domain Authentication Scheme for Wi-Fi Access". en. In: *Wireless Algorithms, Systems, and Applications*. Ed. by Edoardo S. Biagioni, Yao Zheng, and Siyao Cheng. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 149–161. ISBN: 978-3-030-23597-0. DOI: 10.1007/978-3-030-23597-0_12.

[176] *Lightning Explorer*. en-US. URL: https://mempool.space/lightning (visited on 2023-06-09).

[177] *lightningnetwork/lnd*. original-date: 2016-01-16T08:19:33Z. June 2023. URL: https://github.com/lightningnetwork/lnd (visited on 2023-06-09).

[178] Qinwei Lin et al. „Measuring Decentralization in Bitcoin and Ethereum using Multiple Metrics and Granularities“. In: *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*. 2021, pp. 80–87. DOI: 10.1109/ICDEW53142.2021.00022.

[179] Qinwei Lin et al. „Measuring Decentralization in Bitcoin and Ethereum using Multiple Metrics and Granularities“. In: *2021 IEEE 37th International Conference on Data Engineering Workshops (ICDEW)*. ISSN: 2473-3490. Apr. 2021, pp. 80–87. DOI: 10.1109/ICDEW53142.2021.00022.

[180] Peggy Joy Lu, Lo-Yao Yeh, and Jiun-Long Huang. „An Privacy-Preserving Cross-Organizational Authentication/Authorization/Accounting System Using Blockchain Technology“. In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422733.

[181] Peggy Joy Lu, Lo-Yao Yeh, and Jiun-Long Huang. „An Privacy-Preserving Cross-Organizational Authentication/Authorization/Accounting System Using Blockchain Technology“. In: *2018 IEEE International Conference on Communications (ICC)*. ISSN: 1938-1883. May 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422733.

[182] Yi Ma and Hongyun Ning. „Improvement of EAP Authentication Method Based on Radius Server“. In: *2018 IEEE 18th International Conference on Communication Technology (ICCT)*. 2018, pp. 1324–1328. DOI: 10.1109/ICCT.2018.8600077.

[183] Ewen MacAskill et al. „WikiLeaks publishes 'biggest ever leak of secret CIA documents'“. en-GB. In: *The Guardian* (Mar. 2017). ISSN: 0261-3077. URL: https://www.theguardian.com/media/2017/mar/07/wikileaks-publishes-biggest-ever-leak-of-secret-cia-documents-hacking-surveillance (visited on 2023-04-24).

[184] Stanislav Mahula, Evrim Tan, and Joep Crompvoets. „With blockchain or not? Opportunities and challenges of self-sovereign identity implementation in public administration: Lessons from the Belgian case“. In: *DG.O2021: The 22nd Annual International Conference on Digital Government Research*. DG.O'21. New York, NY, USA: Association for Computing Machinery, June 2021, pp. 495–504. ISBN: 978-1-4503-8492-6. DOI: 10.1145/3463677.3463705. URL: https://doi.org/10.1145/3463677.3463705 (visited on 2023-06-22).

[185] Tim A. Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. „Progressive Web Apps: the Definite Approach to Cross-Platform Development?“ In: *Hawaii International Conference on System Sciences*. 2018.

[186] *Push API - Web APIs | MDN*. en-US. Feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/Push_API (visited on 2023-03-10).

[187] Alexey Melnikov and Wei Chuang. *Internationalized Email Addresses in X.509 Certificates*. RFC 8398. May 2018. DOI: 10.17487/RFC8398. URL: https://www.rfc-editor.org/info/rfc8398.

[188] *The Mempool Open Source Project™*. original-date: 2019-07-21T14:42:52Z. Feb. 2023. URL: https://github.com/mempool/mempool (visited on 2023-02-08).

[189] Ilya Mironov and Lintao Zhang. „Applications of SAT Solvers to Cryptanalysis of Hash Functions“. en. In: *Theory and Applications of Satisfiability Testing - SAT 2006*. Ed. by Armin Biere and Carla P. Gomes. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 102–115. ISBN: 978-3-540-37207-3. DOI: 10.1007/11814948_13.

[190] Q. ai-Powering a Personal Wealth Movement. *What Really Happened To LUNA Crypto?* en. Section: Money. Sept. 2022. URL: https://www.forbes.com/sites/qai/2022/09/20/what-really-happened-to-luna-crypto/ (visited on 2023-07-28).

[191] *Using IndexedDB - Web APIs | MDN*. en-US. Feb. 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Using_IndexedDB (visited on 2023-03-10).

[192] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

[193] Madjid Nakhjiri and Mahsa Nakhjiri. „Front Matter“. In: *AAA and Network Security for Mobile Access*. John Wiley & Sons, Ltd, 2005, pp. i–xxi. ISBN: 9780470017463. DOI: https://doi.org/10.1002/0470017465.fmatter. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470017465.fmatter. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470017465.fmatter.

[194] Madjid Nakhjiri and Mahsa Nakhjiri. „Remote Access Dial-In User Service (RADIUS)“. In: *AAA and Network Security for Mobile Access*. John Wiley & Sons, Ltd, 2005. Chap. 6, pp. 127–145. ISBN: 9780470017463. DOI: https://doi.org/10.1002/0470017465.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470017465.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470017465.ch6.

[195] Madjid Nakhjiri and Mahsa Nakhjiri. „The 3 “A”s: Authentication, Authorization, Accounting“. In: *AAA and Network Security for Mobile Access*. John Wiley & Sons, Ltd, 2005. Chap. 1, pp. 1–23. ISBN: 9780470017463. DOI: https://doi.org/10.1002/0470017465.ch1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470017465.ch1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470017465.ch1.

[196] *Namecoin*. URL: https://www.namecoin.org/ (visited on 2023-05-30).

[197] Arvind Narayanan et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. USA: Princeton University Press, 2016. ISBN: 0691171696.

[198] Nelson Wang Nelson Sam Kessler and Danny. *BNB Chain Halts After 'Potential Exploit' Drained Estimated $100M in Crypto*. en. Section: Business. Oct. 2022. URL: https://www.coindesk.com/business/2022/10/06/binance-linked-bnb-price-falls-close-to-4-on-hack-rumors/ (visited on 2022-10-19).

[199] Network RADIUS SARL. *RADIUS System Components | FreeRADIUS Documentation*. URL: https://networkradius.com/doc/3.0.10/concepts/introduction/components.html (visited on 2023-03-17).

[200] *How authentication protocols work*. en. Feb. 2022. URL: https://networkradius.com/articles/2022/02/20/how-authentication-protocols-work.html (visited on 2023-03-01).

[201] Dr. Clifford Neuman et al. *The Kerberos Network Authentication Service (V5)*. RFC 4120. July 2005. DOI: 10.17487/RFC4120. URL: https://www.rfc-editor.org/info/rfc4120.

[202] Matthew Newton. *Getting Started - freeRADIUS*. 2018. URL: https://wiki.freeradius.org/guide/Getting%20Started (visited on 2023-04-17).

[203] Stephan Niedermeier. *Die Geschichte von Ethereum*. de-DE. Dec. 2017. URL: https://ethblog.de/geschichte-von-ethereum/ (visited on 2023-08-07).

[204] *openwisp-radius*. original-date: 2018-08-06T08:07:38Z. Apr. 2023. URL: https://github.com/openwisp/openwisp-radius (visited on 2023-04-09).

[205] *OpenZeppelin/openzeppelin-contracts*. original-date: 2016-08-01T20:54:54Z. May 2023. URL: https://github.com/OpenZeppelin/openzeppelin-contracts (visited on 2023-05-25).

[206] Alexander Papageorgiou et al. „DPKI: A Blockchain-Based Decentralized Public Key Infrastructure System“. In: *2020 Global Internet of Things Summit (GIoTS)*. 2020, pp. 1–5. DOI: 10.1109/GIOTS49054.2020.9119673.

[207] Moritz Platt et al. „The Energy Footprint of Blockchain Consensus Mechanisms Beyond Proof-of-Work“. In: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. ISSN: 2693-9371. Dec. 2021, pp. 1135–1144. DOI: 10.1109/QRS-C55045.2021.00168.

[208] Joseph Poon and Thaddeus Dryja. *The bitcoin lightning network: Scalable off-chain instant payments*. en. 2016.

[209] Arun Pratap and Preeti Saxena. „An Analytical and Experimental Study of AAA Model with Special Reference to RADIUS and TACACS+“. In: *IJCA* 169.9 (July 2017), pp. 6–10. ISSN: 09758887. DOI: 10.5120/ijca2017914870. URL: http://www.ijcaonline.org/archives/volume169/number9/sikarwar-2017-ijca-914870.pdf (visited on 2023-03-15).

[210] *Proof of Transfer | Stacks Docs*. en. URL: https://docs.stacks.co/docs/stacks-academy/proof-of-transfer (visited on 2023-08-04).

[211] *Proof-of-stake (PoS)*. en. URL: https://ethereum.org (visited on 2023-08-04).

[212] *Pulse · smartcontractkit/chainlink*. URL: https://github.com/smartcontractkit/chainlink/pulse/monthly (visited on 2023-06-12).

[213] Muhammad A. Razi, J. Michael Tarn, and Faisal A. Siddiqui. „Exploring the failure and success of DotComs“. In: *Information Management & Computer Security* 12.3 (Jan. 2004). Publisher: Emerald Group Publishing Limited, pp. 228–244. ISSN: 0968-5227. DOI: 10.1108/09685220410542598. URL: https://doi.org/10.1108/09685220410542598 (visited on 2023-06-05).

[214] *React Native · Learn once, write anywhere*. en. 2023. URL: https://reactnative.dev/ (visited on 2023-03-10).

[215] Carl Rigney. *RADIUS Accounting*. RFC 2059. Jan. 1997. DOI: 10.17487/RFC2059. URL: https://www.rfc-editor.org/info/rfc2059.

[216] R[onald] L. Rivest, A[di] Shamir, and L[eonard M.] Adleman. „A method for obtaining digital signatures and public-key cryptosystems“. In: *CACM* 21.2 (Feb. 1978). (This is the "RSA paper".), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342.

[217] Yacine Rouizi. *Working with AJAX in Django*. en. Aug. 2021. URL: https://testdriven.io/blog/django-ajax-xhr/ (visited on 2023-05-22).

[218] Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. „Evaluating Progressive Web App Accessibility for People with Disabilities“. en. In: *Network* 2.2 (June 2022). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, pp. 350–369. ISSN: 2673-8732. DOI: 10.3390/network2020022. URL: https://www.mdpi.com/2673-8732/2/2/22 (visited on 2023-03-10).

[219] Sayak Saha Roy et al. *Demystifying NFT Promotion and Phishing Scams*. 2023. arXiv: 2301.09806 [cs.CR].

[220] Allan Rubens et al. *Remote Authentication Dial In User Service (RADIUS)*. RFC 2058. Jan. 1997. DOI: 10.17487/RFC2058. URL: https://www.rfc-editor.org/info/rfc2058.

[221] Jaffer Sadhique. *How to install build-essential?* Forum post. Oct. 2017. URL: https://askubuntu.com/q/398489 (visited on 2023-09-22).

[222] Joseph A. Salowey and Henry Haverinen. *Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)*. Request for Comments RFC 4186. Num Pages: 92. Internet Engineering Task Force, Jan. 2006. DOI: 10.17487/RFC4186. URL: https://datatracker.ietf.org/doc/rfc4186 (visited on 2023-03-03).

[223] Philipp Sandner. *The Green Bitcoin Theory: How are Bitcoin, Electricity Consumption and Green Energy Related?* en. May 2022. URL: https://philippsandner.medium.com/the-green-bitcoin-theory-how-are-bitcoin-electricity-consumption-and-green-energy-related-b541b23424ab (visited on 2023-08-07).

[224] Philipp Sandner. *Three Ways How Blockchain Technology Is Already Changing The Lives Of Thousands Of Workers In Emerging Countries*. en. Section: Forbes Digital Assets. Aug. 2022. URL: https://www.forbes.com/sites/philippsandner/2022/08/31/three-ways-how-blockchain-technology-is-already-changing-the-lives-of-thousands-of-workers-in-emerging-countries/ (visited on 2023-06-06).

[225] Manfred Scherr, Verena Schwan, and Manuel Hoffrichter. *FISBOX als SaaS-Lösung - Digitale Verwaltung out of the Box*. de. URL: https://hzd.hessen.de/fisbox (visited on 2023-05-19).

[226] Jessica Schilling, Johnny Matthews, and et al. *What is IPFS? | IPFS Docs*. URL: https://docs.ipfs.tech/concepts/what-is-ipfs/#decentralization (visited on 2022-08-15).

[227] Anna-Lena Schlitt et al. „Cyberkriminalität: US-Ermittler finden Lösegeld nach Pipeline-Hackerangriff". de-DE. In: *Die Zeit* (June 2021). ISSN: 0044-2070. URL: https://www.zeit.de/politik/ausland/2021-06/hackerangriff-usa-pipeline-colonial-loesegeld-fbi-ermittler-bitcoin?utm_referrer=https%3A%2F%2Fwww.ecosia.org%2F (visited on 2023-06-06).

[228] SDL. *What is the maximum size of a transaction?* Forum post. Sept. 2014. URL: https://bitcoin.stackexchange.com/q/1823 (visited on 2022-08-25).

[229] Johannes Sedlmeir et al. „The transparency challenge of blockchain in organizations". en. In: *Electronic Markets* 32.3 (Sept. 2022), pp. 1779–1794. ISSN: 1422-8890. DOI: 10.1007/s12525-022-00536-0. URL: https://doi.org/10.1007/s12525-022-00536-0 (visited on 2023-04-08).

[230] Silvia Semenzin, David Rozas, and Samer Hassan. „Blockchain-based application at a governmental level: disruption or illusion? The case of Estonia". In: *Policy and Society* 41.3 (Sept. 2022), pp. 386–401. ISSN: 1449-4035. DOI: 10.1093/polsoc/puac014. URL: https://doi.org/10.1093/polsoc/puac014 (visited on 2023-06-22).

[231] A. Sghaier Omar and O. Basir. „Capability-Based Non-fungible Tokens Approach for a Decentralized AAA Framework in IoT". en. In: *Blockchain Cybersecurity, Trust and Privacy*. Ed. by Kim-Kwang Raymond Choo, Ali Dehghantanha, and Reza M. Parizi. Advances in Information Security. Cham: Springer International Publishing, 2020, pp. 7–31. ISBN: 978-3-030-38181-3. DOI: 10.1007/978-3-030-38181-3_2. URL: https://doi.org/10.1007/978-3-030-38181-3_2 (visited on 2022-05-09).

[232] Shaheed212004. *Django Rest Framework vs Django Ninja*. Reddit Post. Jan. 2022. URL: www.reddit.com/r/django/comments/rx2av8/django_rest_framework_vs_django_ninja/ (visited on 2023-04-11).

[233] Rakesh Sharma. *Can Lightning Network Solve Bitcoin's Scalability Issues?* en. July 2022. URL: https://www.investopedia.com/tech/bitcoin-lightning-network-problems/ (visited on 2023-08-07).

[234] Mark D. Sheldon. „Preparing Auditors for the Blockchain Oracle Problem". In: *Current Issues in Auditing* 15.2 (May 2021), P27–P39. ISSN: 1936-1270. DOI: 10.2308/CIIA-2021-007. URL: https://doi.org/10.2308/CIIA-2021-007 (visited on 2023-04-08).

[235] Na Shi et al. „A blockchain-empowered AAA scheme in the large-scale HetNet". en. In: *Digital Communications and Networks* 7.3 (Aug. 2021), pp. 308–316. ISSN: 2352-8648. DOI: 10.1016/j.dcan.2020.10.002. URL: https://www.sciencedirect.com/science/article/pii/S235286482030273X (visited on 2022-05-03).

[236] Haya Shulman. *Stalloris: RPKI downgrade attack*. en-US. June 2022. URL: https://blog.apnic.net/2022/06/15/stalloris-rpki-downgrade-attack/ (visited on 2023-06-01).

[237] Bundesamt fur Sicherheit in der Informationstechnik. *BSI - Technische Richtlinie: Kryptographische Verfahren: Empfehlungen und Schlussellaengen*. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=2. [(Betrachtet am 27.01.2022)]. 2021.

[238] Daniel Simon, Dr. Bernard D. Aboba, and Pasi Eronen. *Extensible Authentication Protocol (EAP) Key Management Framework*. RFC 5247. Aug. 2008. DOI: 10.17487/RFC5247. URL: https://www.rfc-editor.org/info/rfc5247.

[239] K. Smith. *PPP Authentication Protocols*. Request for Comments 1334. Internet Engineering Task Force, Oct. 1992. URL: https://tools.ietf.org/html/rfc1334.

[240] Solana. *Skalierbare Blockchain-Infrastruktur: Milliarden von Transaktionen | Solana: Build crypto apps that scale*. de. URL: https://solana.com/de (visited on 2022-10-19).

[241] Mladen Stanke and Mile Sikic. „Comparison of the RADIUS and Diameter protocols". In: *ITI 2008 - 30th International Conference on Information Technology Interfaces*. ISSN: 1330-1012. June 2008, pp. 893–898. DOI: 10.1109/ITI.2008.4588529.

[242] *Leading cause of ransomware infection 2020*. en. 2020. URL: https://www.statista.com/statistics/700965/leading-cause-of-ransomware-infection/ (visited on 2023-03-09).

[243] Ted Stevenot. *How many bitcoin are there?* en-US. June 2022. URL: https://unchained.com/blog/how-many-bitcoin-are-there/ (visited on 2022-08-23).

[244] Péter Szilágyi. *EIP-225: Clique proof-of-authority consensus protocol*. en. June 2017. URL: https://eips.ethereum.org/EIPS/eip-225 (visited on 2023-08-04).

[245] Michael Szydlo. „Merkle Tree Traversal in Log Space and Time". en. In: *Advances in Cryptology - EUROCRYPT 2004*. Ed. by Christian Cachin and Jan L. Camenisch. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 541–554. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_32.

[246] Shigeaki Tanimoto et al. „Quantifying Cost Structure of Campus PKI“. In: *2011 IEEE/IPSJ International Symposium on Applications and the Internet* (July 2011). Conference Name: 2011 IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT) ISBN: 9781457705311 Place: Munich, Germany Publisher: IEEE, pp. 315–320. DOI: 10.1109/SAINT.2011.60. URL: http://ieeexplore.ieee.org/document/6004177/ (visited on 2023-04-07).

[247] . *Timeline of Certificate Authority Failures - SSLMate*. URL: https://sslmate.com/resources/certificate_authority_failures (visited on 2023-04-07).

[248] Josef Tischmacher. *Was sind Parachains?* de. July 2022. URL: https://blockchainwelt.de/parachains/ (visited on 2022-11-28).

[249] Andrew Tobin and Drummond Reed. *The Inevitable Rise of Self-Sovereign Identity*. https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf. (Accessed on 07/13/2021). June 2017.

[250] Muoi Tran et al. „A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network“. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 894–909. DOI: 10.1109/SP40000.2020.00027.

[251] Wortschatz - Leipzig University. *Wortschatz – deu_news_2021 – Dezentralisation*. URL: https://corpora.uni-leipzig.de/de/res?corpusId=deu_news_2021&word=Dezentralisation (visited on 2022-10-19).

[252] Wortschatz - Leipzig University. *Wortschatz – deu_news_2021 – Verteilung*. URL: https://corpora.uni-leipzig.de/de/res?corpusId=deu_news_2021&word=Verteilung (visited on 2022-10-19).

[253] Blesson Varghese et al. „Realizing Edge Marketplaces: Challenges and Opportunities“. In: *IEEE Cloud Computing* 5 (Nov. 2018), pp. 9–20. DOI: 10.1109/MCC.2018.064181115.

[254] JP Vergne. „Decentralized vs. Distributed Organization: Blockchain, Machine Learning and the Future of the Digital Platform“. en. In: *Organization Theory* 1.4 (Oct. 2020). Publisher: SAGE Publications Ltd, p. 2631787720977052. ISSN: 2631-7877. DOI: 10.1177/2631787720977052. URL: https://doi.org/10.1177/2631787720977052 (visited on 2022-10-16).

[255] Leon Waidmann. *4 Skalierungslösungen, die Ethereum 2021 Mainstream-tauglich machen*. de-DE. Jan. 2021. URL: https://www.btc-echo.de/news/4-skalierungsloesungen-die-ethereum-2021-mainstream-tauglich-machen-107856/ (visited on 2022-11-28).

[256] Guizhou Wang et al. „A Systematic Overview of Blockchain Research“. en. In: *Journal of Systems Science and Information* 9.3 (June 2021). Publisher: De Gruyter, pp. 205–238. ISSN: 2512-6660. DOI: 10.21078/JSSI-2021-205-34. URL: https://www.degruyter.com/document/doi/10.21078/JSSI-2021-205-34/html (visited on 2023-08-23).

[257] *What is a zero-knowledge proof?* en-US. URL: http://zkp.science/ (visited on 2023-08-03).

[258] *RADIUS*. en. Page Version ID: 1143963913. Mar. 2023. URL: https://en.wikipedia.org/w/index.php?title=RADIUS&oldid=1143963913#History (visited on 2023-03-17).

[259] Wikipedia. *CAP-Theorem*. de. Page Version ID: 222904512. May 2022. URL: https://de.wikipedia.org/w/index.php?title=CAP-Theorem&oldid=222904512 (visited on 2022-08-06).

[260] P. Williams, L. Johansson, and R. Callon. *Clarifications and Extensions to the Generic Security Service Application Program Interface (GSS-API) for the Use of Channel Bindings*. RFC 5554. Internet Engineering Task Force, May 2009. DOI: 10.17487/RFC5554. URL: https://www.rfc-editor.org/rfc/rfc5554.txt.

[261] Wirtschaftsinformatik des Fraunhofer-Instituts für Angewandte Informationstechnik FIT. *Entwicklung einer datenschutzkonformen Blockchain-Lösung im deutschen Asylprozess*. de. https://www.bamf.de/SharedDocs/Anlagen/DE/Digitalisierung/blockchain-whitepaper.pdf?__blob=publicationFile&v=10. 2019.

[262] Lilith Wittmann. *Mit dem Personalausweis zum Onlineshopping: Wie selbstbestimmt sind "selbstbestimmte Identitäten"?* en. May 2022. URL: https://lilithwittmann.medium.com/mit-dem-personalausweis-zum-onlineshopping-wie-selbstbestimmt-sind-selbstbestimmte-identit%C3%A4ten-f096a5bdd55a (visited on 2023-03-14).

[263] Dr Gavin Wood. *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER*. en. Oct. 2022.

[264] Pieter Wuille, Jorge Timon, and et. al. *Bitcoin Core integration/staging tree*. original-date: 2010-12-19T15:16:43Z. Aug. 2022. URL: https://github.com/bitcoin/bitcoin/blob/3c098a8aa0780009c11b66b1a5d488a928629ebf/src/consensus/tx_verify.cpp (visited on 2022-08-25).

[265] Karl Wüst and Arthur Gervais. „Do you Need a Blockchain?" In: *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. 2018, pp. 45–54. DOI: 10.1109/CVCBT.2018.00011.

[266] *Xamarin | Open-source mobile app platform for .NET*. en-US. 2023. URL: https://dotnet.microsoft.com/en-us/apps/xamarin (visited on 2023-03-10).

[267] Peter E. Yee. *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 6818. Jan. 2013. DOI: 10.17487/RFC6818. URL: https://www.rfc-editor.org/info/rfc6818.

[268] Eun-Jun Yoon, Wan-Soo Lee, and Kee-Young Yoo. „Secure PAP-Based RADIUS Protocol in Wireless Networks". en. In: *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*. Ed. by De-Shuang Huang, Laurent Heutte, and Marco Loog. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2007, pp. 689–694. ISBN: 978-3-540-74282-1. DOI: 10.1007/978-3-540-74282-1_77.

[269] Liudmila Zavolokina et al. „Leitfaden Blockchain in der kantonalen Verwaltung Zürich". de. In: (Feb. 2021).

[270] Xinrui Zhang et al. „Time-manipulation Attack: Breaking Fairness against Proof of Authority Aura". In: *Proceedings of the ACM Web Conference 2023*. WWW '23. New York, NY, USA: Association for Computing Machinery, Apr. 2023, pp. 2076–2086. ISBN: 978-1-4503-9416-1. DOI: 10.1145/3543507.3583252. URL: https://dl.acm.org/doi/10.1145/3543507.3583252 (visited on 2023-08-04).

[271] Qiheng Zhou et al. „Solutions to Scalability of Blockchain: A Survey". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 16440–16455. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2967218.

Appendix A

Zusammenfassung / German Summary

Einleitung

Das Aufkommen der Blockchain-Technologie hat in den vergangenen Jahren große öffentliche Aufmerksamkeit erregt und wurde von der Forschungsgemeinschaft ausgiebig untersucht [256]. Die bisherigen Forschungsarbeiten liefern zwar vielversprechende Ergebnisse, konzentrieren sich aber auf Anwendungen und Entwicklungen im Finanzbereich oder auf die Realisierung wirtschaftlicher Geschäftsfälle. Aber auch wenn es eine Minderheit an Forschungspublikationen gibt, sind technische Verbesserungen für technische Systeme wie Internet of Things (IoT) und digitale Identitäten gezeigt worden.
Diese Arbeit zielt darauf ab, das wissenschaftliche Verständnis über den Kontext finanzieller und wirtschaftlicher Gewinne hinaus zu erweitern, indem sie untersucht, ob und wie Blockchain auf ein sich ausbreitendes Problem für zunehmend digitalisierte Gesellschaften angewendet werden kann: wie kann die öffentliche Verwaltung die Technologie nutzen, um die Verarbeitungseffizienz zu verbessern und sich vor modernen Fäden durch zentralisierte bösartige Bedrohungsakteure zu schützen? Blockchain bietet eine mögliche Antwort auf diese Frage, indem es eine dezentrale und sichere Lösung für eine widerstandsfähige öffentliche Verwaltung bietet. Die Untersuchung, wie Blockchain auf Verwaltungsprozesse angewendet werden kann, einschließlich einer umfassenden Analyse, der der Technologie innewohnenden Probleme, der Implementierungshürden und der potenziellen Sicherheitsvorteile, wird ein Schwerpunkt dieser Forschung sein.
Zur Veranschaulichung der untersuchten Konzepte wird ein Design Science Research (DSR)-Konzept verwendet. Das DSR ist eine Forschungsmethodik, die zur Erstellung und Bewertung von IT-Artefakten verwendet wird, mit dem Ziel, identifizierte organisatorische Probleme zu lösen. In diesem Zusammenhang kann ein Artefakt Software, Technologie oder Methoden sein. Es wird eine Implementierung des RADIUS-Protokolls auf einer Blockchain entwickelt, um zu zeigen, wie ein solches System nahtlos in die bestehenden IT-Systeme der öffentlichen Verwaltungen integriert werden kann und welche potenziellen Herausforderungen sich daraus ergeben. Das Forschungsprojekt wurde in Zusammenarbeit mit dem HZD entwickelt. Als IT-Dienstleister des Landes Hessen ist der HZD ein wesentlicher Akteur bei der digitalen Transformation von Verwaltungsprozessen. In einer technologisch immer komplexer werdenden Welt ist er ständig auf der Suche nach innovativen Lösungen, um Effizienz, Sicherheit und Zuverlässigkeit zu gewährleisten.
In der Zeit der Kooperationsvereinbarungen des Forschungsvorhaben mit der HZD ebbte das Blockchain-Aufsehen ab. In diese Aufsehen-Phase, die durch zahlreiche Projekte des Initial Coin Offering (ICO) ausgelöst wurde, hatte man zunächst den Eindruck, dass eine Blockchain in vielen Bereichen anwendbar ist und sich als vorteilhaft erweist. Doch erst das Scheitern der vielen ICO-Projekte gab den Anstoß, der Forschungsfrage nachzugehen. Ist eine Blockchain in der Lage, die Effizienz und Sicherheit von Prozessen in der öffentlichen Verwaltung zu erhöhen?
Öffentliche Verwaltungen sind in ihren Strukturen und Abläufen sehr komplex. Da es sich um eine Arbeit aus dem Bereich der Informatik handelt, sollte der Fokus auf einer technischen Umsetzung liegen. Soziologische und ökonomische Überlegungen werden daher nur kurz angerissen und bedürfen weiterer Forschung,

neben dieser Arbeit.
Die Blockchain-Technologie hat im Jahr 2017 aufgrund des Aufsehens, der zu einer Vielzahl von Projekten führte, große Aufmerksamkeit erregt. Eines der Versprechen war und ist die Dezentralisierung. Die Fähigkeit, Prozesse ohne einen Trusted Third Party (TTP) zu implementieren, klingt vielversprechend. Ob diese Eigenschaft gehalten werden kann, und ob diese Eigenschaft benötigt wird, ist Gegenstand dieser Arbeit.
Weit weniger bekannt, aber weitverbreitet ist das sogenannte RADIUS-Protokoll. Es ermöglicht die Ausführung von Aufgaben über verschiedene Netze hinweg, die sich unter dem Akronym AAA zusammenfassen lassen. Ein unter Hochschulmitarbeitern und Studenten weithin bekanntes Beispiel ist das eduroam-Netzwerk. Dieser Zugang zur WLAN-Infrastruktur ermöglicht es, sich an einer Universität zu registrieren, sich aber mit dieser Registrierung an allen an eduroam teilnehmenden Universitäten authentifizieren zu können. Die Weiterleitung von Authentifizierungsanfragen wird über einen TTP realisiert. Es stellt sich also sofort die Frage, ob man ein solches System mit einer dezentralen Technologie, wie z. B. Blockchains, umsetzen kann. Ferner stellen sich die Fragen der Sinnhaftigkeit einer solchen Portierung.

Motivation

Wie bereits erwähnt, ist die HZD an technologischen Innovationen und Lösungen interessiert, die seine eigenen Prozesse effizienter machen. Da es sich bei dieser Organisation um einen IT-Dienstleister für öffentliche Verwaltungen handelt, bietet sie eine geeignete Plattform für die Durchführung von Forschungsarbeiten in diesem Bereich. Aus Sicht der öffentlichen Verwaltung ist es interessant zu untersuchen, ob eine dezentrale Architektur die föderalen Systeme bei ihrer Digitalisierung unterstützen oder nutzen kann. Überdies hat sich ein weiteres Projekt, nämlich das hessische WLAN, als interessante Schnittstelle für die Implementierung eines RADIUS-Systems erwiesen. Daher ist es naheliegend, diese Forschungsbereiche zu kombinieren.
Weitere Ideen ergeben sich aus den bisherigen Überlegungen. So stellt sich unter anderem die Frage, ob die Implementierung des RADIUS-Systems auf einer Blockchain um die Fähigkeit zur Implementierung eines SSO erweitert werden kann. Dabei würden die für die Implementierung von AAA verwendeten Methoden zum Einsatz kommen. Ferner stellt sich die Frage, ob eine Implementierung eines SSO auch auf physischer Ebene durch die Integration von Türschlössern und Handvenenscannern möglich wäre. Die Authentifizierung einer Person müsste dann nur einmal erfolgen, möglicherweise über eine Blockchain. Die Zugangsberechtigung würde dann nicht nur für die betreffende Person, sondern auch für die von ihr mitgeführte Hardware gelten.
Schließlich ist noch die Frage der Dezentralisierung zu klären. Wie funktioniert die Dezentralisierung, und ist sie wünschenswert? Es empfiehlt sich, die Vor- und Nachteile sowie die Integrationsmöglichkeiten anhand des oben beschriebenen Beispiels darzustellen und zu diskutieren.

Problembeschreibung

Die oben beschriebene Motivation lässt vermuten, dass es viele mögliche Anwendungen gibt, die hier untersucht werden. Doch welche Probleme müssen angegangen werden?
Da sind zum einen die öffentlichen Verwaltungen. Diese sind in Deutschland föderal strukturiert und dienen in erster Linie dem Schutz der Demokratie. Im Zuge der Nachforschung und Implementierung für diese Arbeit wurde jedoch eine Schwierigkeit deutlich. Für jedes Bundesland werden immer wieder zahlreiche Ansätze simultan untersucht. Durch die Eigenständigkeit der 16 Bundesländer oder auch der jeweiligen Kommunen ergibt sich zudem oft eine sehr heterogene Systemarchitekturlandschaft. Absprachen untereinander sind sehr kompliziert und aufgrund der Vielzahl der möglichen Standorte nicht immer möglich. Überschneidungen ergeben sich somit in regelmäßigen Fällen.
Im Gegensatz zu der eben beschriebenen Struktur der Bundesrepublik Deutschland ist eduroam hierarchisch aufgebaut. Das eduroam-Projekt wird mit dem RADIUS-Protokoll realisiert. Es existiert also ein TTP für die Weiterleitungsberechtigung. Auch in anderen Anwendungen des Protokolls gibt es einen TTP. Diese

Autoritätssysteme können jedoch gefährdet sein, wenn die oberste zentrale Autorität ausfällt. Dabei ist es unerheblich, ob das Versagen unbeabsichtigt war oder von einer böswilligen Entität absichtlich herbeigeführt wurde. Diese zentralisierte Gefährdung wird auch als Single Point of Failure (SPOF) bezeichnet. Wenn eine dezentralisierte Architektur das Problem lösen kann, sollte sie in Betracht gezogen werden.
Es gibt viele Arten von Blockchain-Technologien. Jede Lösung hat ihre eigenen Stärken und Schwächen, je nach ihren eigenen Spezifikationen. Es gibt jedoch auch Aspekte, die derzeit allen aktuellen Blockchain-Systemen unterlegen sind, obwohl es unterschiedliche Systeme für unterschiedliche Anwendungen gibt. Dazu gehören das Transparenz-, das Orakel- und das Skalierungsproblem. Auf diese Probleme wird später noch eingegangen. Mögliche Lösungen und Implikationen sollen diskutiert und aufgezeigt werden.

Forschungsziele

Diese Arbeit befasst sich mit der Frage nach den möglichen Auswirkungen der Blockchain-Technologie auf die öffentliche Verwaltung. Dazu wird eine beispielhafte Blockchain-Anwendung anhand einer RADIUS-Implementierung dargestellt und umgesetzt. Aus den beschriebenen Motivationen und Problemstellungen lassen sich weitere Fragen ableiten, die im Vorfeld der Forschung relevant sind und untersucht werden sollten. Folglich kann es zu Überschneidungen bei der Beantwortung der Fragen kommen.

- **Was kann eine Blockchain leisten, und wo liegen ihre Grenzen?** Viele Blockchain-Projekte sind gescheitert. Darauf deutet ein Blick auf die Anzahl der Kryptowährungen mit ihren Kursentwicklungen hin. Gerade die vielen gescheiterten Blockchain-Projekte sollten ein Indiz für einen fragwürdigen Einsatz der Technologie sein. Was können Blockchains, und wo stoßen sie an ihre Grenzen?

- **Wie kann eine Blockchain in bestehende Systeme integriert werden?** Die in dieser Arbeit vorgestellten Anwendungen können meist in hybrider Form realisiert werden. Dies ist auch sinnvoll, da ein bestehendes System nicht komplett ersetzt werden muss. Der mögliche Ansatz, ein Blockchain-System in ein bestehendes System einzubetten, soll untersucht und beantwortet werden.

- **Was sind die Hintergründe für die Einführung einer Blockchain in öffentlichen Verwaltungen?** Sobald die technische Machbarkeit bewiesen ist, stellt sich die Frage, warum man sich darum bemüht.

- **Ist die Blockchain-Technologie in der Lage, die Demokratie zu fördern?** Die erwähnte föderale Struktur in Deutschland bringt eine Verteilung der Macht und eine Gewaltenteilung mit sich. Ist ein dezentrales IT-System in der Lage, solche föderalen Strukturen zu unterstützen?

- **Was sind die Voraussetzungen für eine erfolgreiche Blockchain-Implementierung?** Auch hier zeigen die gescheiterten Projekte, dass Blockchains schwierig zu implementieren sind. Falls entsprechenden Punkte eines sinnvollen Einsatzes identifiziert wurden, was sind möglichen Komplikationen? Wie kann ein Projekt erfolgreich realisiert werden?

Grundlagen

Für die Bearbeitung dieser Arbeit sind zwei Technologien wesentlich. Das ist zum einen das Remote Authentication Dial-In User Service (RADIUS) Protokoll. Dieses ermöglicht die Umsetzung von Authentication, Authorization and Accounting (AAA) Aufgaben. Diese Aufgaben fallen häufig in netzwerkübergreifende Authentifizierungen an. Die zweite wesentliche Technologie sind die Blockchains und DLTs. Ihr Ziel ist es im Wesentlichen ein Zustand über Transaktionshistorien über größere Netzwerke hinweg auf Integrität zu sichern. Bekanntestes Beispiel ist der Bitcoin (BTC). Beim BTC wird versucht, eine digitale Ware oder Währung zu realisieren.

Das Remote Authentication Dial-In User Service (RADIUS) ist ein Netzwerkprotokoll, das die Aufgaben der Authentifizierung, Autorisierung und Abrechnung von Benutzern in einem Netzwerk umsetzt und ist als zentralisiertes Client-Server-Modell implementiert. Dabei sendet der Benutzer in der Regel über einen Client eine Kombination aus Benutzernamen und Passwort an den RADIUS-Server. Dieser Server prüft die verfügbaren Daten und autorisiert den Benutzer entsprechend für den Zugang zum Netzwerk. Falls erforderlich, werden zusätzliche Informationen zu Abrechnungszwecken vom Server gesammelt und verwaltet. Eine wichtige Funktion von RADIUS ist die netzübergreifende Nutzung der Methoden, die als Roaming bezeichnet wird.

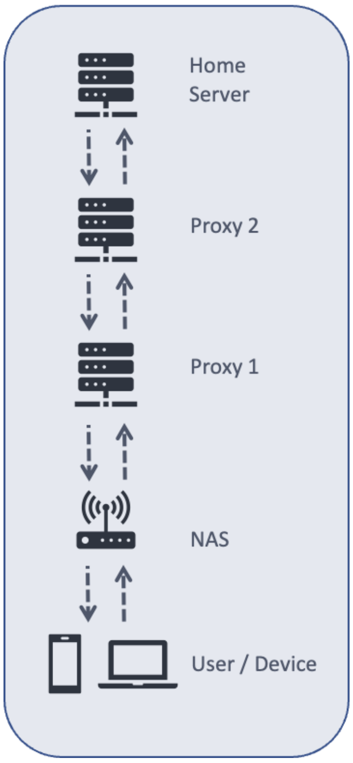


Abbildung A.1: Darstellung der RADIUS-Rollen im Roaming-Konzept.

Durch die Implementierung der AAA-Funktionen kann RADIUS für Netzbetreiber von enormer Bedeutung sein. Erstens implementiert es einen vordefinierten Benutzerzugang zum Netzwerk. Nur autorisierte Benutzer mit den entsprechend definierten Rechten werden in das Netzwerk gelassen. Damit wird der Anwendung ein großer Teil der Verantwortung für eine sichere und integre Implementierung übertragen. Ein solches System ist aber auch essenziell für eine eventuell notwendige Abrechnung, für die erfolgreiche Abwicklung der Benutzerabrechnung mit Integrität und damit möglicherweise für die Sicherstellung der Abrechnung des Netzbetreibers. Für Implementierungen des RADIUS-Protokolls wurden bereits viele Standardisierungen in Form von Request for Comments (RFC)-Dokumenten erstellt. Eine übersichtliche Liste von RFC-Dokumenten, die das RADIUS-Protokoll betreffen, findet sich auf der Wikipedia-Seite [258].

Besonders spannend ist in dem RADIUS Protokoll die Möglichkeit der Weiterleitung von Anfragen. So ist es möglich, sich mit seinen Anmeldeinformationen aus dem Heimnetzwerk in fremde Netzwerke zu authentifizieren. Beispielanwendung dafür sind Mobilfunknetze und das eduroam Netzwerk.

Seit der Einführung von Bitcoin, mit dem Papier "Bitcoin: A peer-to-peer electronic cash system"[192], hat sich die Technologie erheblich weiterentwickelt. Im Jahr 2009 wurde eine Implementierung und Einführung einer Instanz der vorgestellten Technologie durchgeführt, wie in [33] dokumentiert. Neben den Weiterentwicklungen von Bitcoin, aus denen auch einige sogenannte Forks (dazu später mehr) hervorgegangen sind, haben sich neue Blockchain-Technologien entwickelt. Diese zielen darauf ab, die Skalierbarkeit zu verbessern, indem eine neuartige Datenstruktur verwendet wird, die von linearen und verketteten Blöcken abweicht. Daher wurde in der Literatur ein neuer verallgemeinernder Begriff identifiziert, nämlich Distributed Ledger Technology (DLT). Durch die Verallgemeinerung schließt DLT auch Blockchains mit ein. Dennoch ist der Begriff Blockchain weiterhin in der Literatur zu finden. Einerseits, weil der Begriff vertrauter ist. Andererseits aber auch, weil die anderen Technologien den Blockchains in ihren Grundprinzipien sehr ähnlich sind. Daher wird in der Literatur der Begriff Blockchain in einer Weise verwendet, die DLTs integriert. Auch in dieser Arbeit gehen wir so vor.

Eine Blockchain kann als eine Liste von Datenblöcken verstanden werden, die in sich selbst verkettet sind. Die Verkettung wird durch die Verwendung der ID des vorherigen Blocks im aktuellen Block selbst erzeugt. Die notwendigen Informationen für einen Block werden in einer kryptografischen Hash-Funktion kodiert, und die daraus resultierende Ausgabe wird als Identifikationsnummer des neuen Blocks verwendet. Diese wird dann im folgenden Block verwendet. Durch diese Verkettung wird eine Datenstruktur geschaffen, die es ermöglicht, minimale Änderungen schnell zu erkennen. Dies geht jedoch auf Kosten der Veränderbarkeit der Daten. Möchte eine Person ein Datum ändern oder streichen, so wird der Hashwert des Blocks entsprechend dem genannten Datum geändert. In der Folge ändern sich die Hashwerte aller folgenden Blöcke.

Je früher das Datum in der Blockchain steht, desto mehr Blöcke müssen neu berechnet werden. Für handelsübliche Computer ist dies in der Regel keine schwierige Aufgabe. Hash-Werte können in Bruchteilen von Sekunden berechnet werden. Deshalb wird diese Datenstruktur bei der Anwendung von Blockchains noch mit einem Konsens versehen. Die Ziele des Konsenses sind die Sicherstellung der Datenintegrität innerhalb der eigenen Datenstruktur und die Synchronisation der Daten innerhalb des verteilten Netzwerks. Kein Knoten im Netzwerk darf die Daten verarbeitet haben. Dies würde zu Inkonsistenzen führen und das Netzwerk gefährden. Der Konsens ist daher von großer Bedeutung, vorrangig beim Vergleich verschiedener Knotenpunkte. Bitcoin (BTC) verwendet mit PoW ein relativ teures Verfahren. Dies führt immer wieder zu dem Argument, Bitcoin verbrauche zu viel Energie [80, 18]. Es gibt jedoch auch Argumente, die für den Energiebedarf und Konsensverfahren sprechen, die deutlich weniger Energie verbrauchen. Für unsere Daten, die derzeit in der Blockchain-Datenstruktur enthalten sind, ist es zwingend erforderlich, dass der Konsens bei allen Änderungen eingehalten wird. Neben der Übereinstimmung der Hash-Werte sind also noch andere Faktoren wichtig, die eine schnelle Änderung erschweren oder sogar unmöglich machen können. Wir haben eine Datenstruktur, die als äppend-onlyStruktur bezeichnet wird.

Konzeption und Methodik

Um die in der Einleitung gestellten Forschungsfragen zu beantworten, wird ein DSR-Ansatz als Forschungsmethodik verwendet. In der Wirtschaftsinformatik und verwandten Disziplinen ist der DSR-Ansatz eine Forschungsmethode, die dazu dient, innovative und praktische Lösungen für reale Probleme zu entwickeln und gleichzeitig das theoretische Wissen zu erweitern. Der DSR-Ansatz wird oft als Brücke zwischen Theorie und Praxis gesehen, da er darauf abzielt, sowohl praktische Lösungen zu entwickeln, als auch theoretisches Wissen zu erweitern. [78]
Durch die enge Zusammenarbeit mit dem Forschungspartner HZD, mit dem Ziel, eine Eigenentwicklung mit Schnittstellen zu HZD-Systemen zu realisieren, ergibt sich eine intensive Verknüpfung von Theorie und Praxis im Forschungsprojekt. Da dies in DSR explizit unterstützt und gefordert wird, ist diese Forschungsmethodik auf den weiteren Prozess dieser Arbeit abgestimmt.
Die an die Arbeit angepasste Vorgehensweise besteht aus den folgenden Schritten:

- Problemidentifikation: Die in der Einleitung gestellten Forschungsfragen verteilen sich auf mehrere Aspekte. *Kann das RADIUS-Protokoll auf Blockchain verwendet werden?* und *Evaluierung von Blockchain-Technologien für öffentliche Verwaltungen*. Für letzteren Aspekt werden der Entwicklungsstand verschiedener historischer und aktueller Projekte dargestellt und mögliche Stärken und Schwächen aufgezeigt.

- Definition der Ziele: Zunächst wird der aktuelle Stand der Entwicklungen dargestellt. Der Fokus liegt dabei auf der Weiterentwicklung von RADIUS-Protokollen, sowie der Weiterentwicklung von Blockchains und DLTs. Auch die bestehende Kombinationsforschung wird näher beleuchtet. Das zweite Ziel ist es, ein Framework zu entwerfen und zu implementieren, das das RADIUS-Protokoll auf Blockchain realisieren kann. Die Ergebnisse der Forschungsstudie werden mit denen der Weiterentwicklung des RADIUS-Protokolls und DLTs verglichen und erörtert. Dabei werden die Fragen nach der Anwendung von Blockchain für die öffentliche Verwaltung beantwortet.

- Design und Entwicklung: Um die definierten Ziele zu erreichen, wird zunächst eine systematische Literaturrecherche durchgeführt. Hierfür werden verschiedene Quellen genutzt. Dazu gehören die Datenbanken Google Scholar, IEEE, Springer, ACME und Elicit. Letztere ist ein KI-gestützter Dienst, der den Nutzern hilft, geeignete wissenschaftliche Literatur zu finden. Blockchain für eduroam, Blockchain für Radius und Blockchain in der öffentlichen Verwaltung waren einige der verwendeten Suchanfragen. Darüber hinaus wurde eine Auswahl an Literatur zu den jeweiligen technologischen Weiterentwicklungen getroffen. Es wurden Quellen verwendet, die einen Bezug zur eigenen

Forschung haben und den Kriterien für gute Arbeiten entsprechen. Zu diesen Kriterien gehören die Anzahl der Zitate, der Zeitpunkt der Veröffentlichung und die Anzahl der Konferenzberichte.

Auf die Klassifizierung folgt der Entwurf und die Implementierung des RADIUS-Protokolls auf der Blockchain. Dieser Entwurf und die Implementierung werden AAA-me genannt. Die Messungen von Authentifizierungen mit AAA-me wird dann in einem Laborexperiment mit einem RADIUS-System verglichen.

- Demonstration: Die Implementierung von AAA-me und die Durchführung des AAA-me Experiments folgen auf die Konzeption.

- Evaluation: Abschließend wird alles zusammen evaluiert und in einen Kontext gebracht.

Es gibt zwei weitere Aspekte von DSR. Der erste zusätzliche Aspekt ist die Kommunikation; die Forschungsergebnisse werden mit Experten und Interessenvertretern kommuniziert und diskutiert. Die Ergebnisse dieser Diskussionen fließen in den Entwicklungsprozess ein. Iterative Diskussionen mit dem HZD und einem weltweiten Forschungsnetzwerk namens Bloxberg fanden während der Entwicklung kontinuierlich statt. Letzteres wurde genutzt, um den universitären Forschungsaustausch zu erleichtern und um zusätzliche Perspektiven zu erhalten. Der zweite zusätzliche Aspekt von DSR ist die Verfolgung der Forschungsergebnisse. Dies geschieht durch die Veröffentlichung dieser Dissertation.

AAA-me

Das Ziel von AAA-me ist es, die Integration bestehender RADIUS-Systeme, die Entwicklung von RADIUS-Systemen und die Verbindung zu Blockchain-Systemen zu erleichtern. Somit wird es sowohl für eine hybride als auch für eine vollständige Implementierung geeignet sein. Außerdem wird erwartet, dass Blockchain-Anwendungen, die gemeinhin als Smart Contracts bezeichnet werden, überwacht und kontrolliert werden können. Eine Erweiterung hiervon könnte die Einbindung einer Blockchain-Programm-IDE zur Erstellung von Blockchain-Programmen sein.
Für jede der drei Schnittstellenimplementierungen muss mindestens eine Blockchain-Infrastruktur vorhanden sein. Daher ermöglicht das Framework, sich mit einer Blockchain-Instanz zu verbinden. Die Voraussetzungen für eine solche Verbindung werden durch die jeweilige Blockchain-Technologie und -Instanz festgelegt. Damit der Nutzen von AAA-me möglichst breit gefächert ist, nutzt das System eine Vielzahl von Blockchain-Technologien. Zunächst sind dies ETH, BTC und Hyperledger Indy. Andere Technologien werden derzeit evaluiert und können hinzugefügt werden.
Weitere Aufgaben im Kontext der Blockchain-Schnittstellen sind die Überwachung des aktuellen Status des Netzwerks, die Bereitstellung von Schnittstellen für die Interaktion mit der jeweiligen Blockchain-Instanz sowie das Abrufen von Daten von verschiedenen Knoten und deren Vergleich untereinander. Der letzte Schritt besteht darin, sicherzustellen, dass es nicht zu einer Unterbrechung des Netzwerks oder einem Angriff kommt. Um dies zu erreichen, muss eine konstant aktualisierte Liste von Knoten erstellt und gepflegt werden. Es gibt eine vordefinierte Anzahl von Knoten, mit denen eine Verbindung besteht. Die aus der Blockchain abgefragten Informationen werden miteinander verglichen.
Es wäre zu teuer, einen separaten Knoten für verschiedene Blockchain-Projekte innerhalb von AAA-me zu entwickeln. Die Erstellung und Verwaltung eines Netzwerkelements sind extrem aufwendig und aufgrund der erforderlichen Sicherheitsrichtlinien nicht praktikabel. Daher kommen für AAA-me nur Schnittstellen zu Knoteninstanzen infrage. Um dies zu erreichen, wird versucht, so viele Schnittstellen wie möglich zu implementieren. Dies impliziert, dass eine Vielzahl von Blockchain-Technologien für die Realisierung und Integration von AAA-me-Funktionalitäten ermöglicht wird. Dies erfordert den Aufbau modularer Strukturen.

Neben der Kontrolle über Blockchain-Instanzen müssen wir auch die Kontrolle über Blockchain-Programme (Smart Contracts) haben. Dazu gehören, wie bei den Blockchain-Instanzen, die Überwachung, die Bereitstellung von Schnittstellen und die Kontrolle dieser Programme. AAA-me stellt eine Auswahl an vordefinierten Authentifizierungsschemata als Templates zur Verfügung, sodass diese auch mit dem Framework auf der jeweiligen Blockchain-Instanz verwendet werden können.
Zusammenfassung der AAA-me-Aufgaben:

1. Blockchain-Instanzen
    - Überwachung
    - Schnittstellenbereitstellung (Dokumentation, Anpassungen, ...)
    - Knotenlistenverwaltung
   - Blockchain-Programme (Smart Contracts)
     - Überwachung
     - Kontrolle und Interaktion
     - Schnittstellenbeschreibung
     - Blockchain-Programme über verschiedene Blockchain-Instanzen und Technologien hinweg.
     - Bereitstellung und Wartung
   - Authentifizierung
     - Methoden Instandhaltung
     - Dokumentation
2. RADIUS Systeme
    - Überwachung
    - Steuerung und Interaktion
    - Schnittstellenbeschreibung

Zusammenfassend kann gesagt werden, dass AAA-me als Konfigurator konzipiert und entwickelt wird. Nachdem Sie Ihre Einrichtung konfiguriert haben, werden Sie sie voraussichtlich nicht mehr benötigen. Um den Funktionsumfang zu erweitern, werden verschiedene Überwachungsmöglichkeiten angeboten. So gewinnt die Software mit minimalem Mehraufwand erheblich an Bedeutung.

Architecture

Für eine effiziente Realisierung von AAA-me werden verschiedene Libraries und Frameworks verwendet. Dies beschleunigt die Entwicklung und ermöglicht es, eine Implementierung sogar im kleinen Team voranzubringen. Für AAA-me wurden im Wesentlichen für drei Module entsprechend Libraries und Frameworks genutzt. In der Grafik A.2 wird ein Gesamtüberblick über die genutzten Technologien gegeben.
Das erste Modul ist Django. Hiermit wird ein Webserver bereitgestellt und implementiert, das im Wesentlichen eine Übersicht über die oben erwähnten Aufgaben geben soll. Zusätzlich können weitere Funktionalitäten wie Application Programming Interface (API) Schnittstellen oder Progressive Web Apps (PWAs) implementiert werden.
Die Integration von AAA-me in bestehende oder neu aufgebaute RADIUS Systeme wird mit dem Framework OpenWISP RADIUS implementiert. Dieses Framework wendet wiederum das Tool freeRADIUS

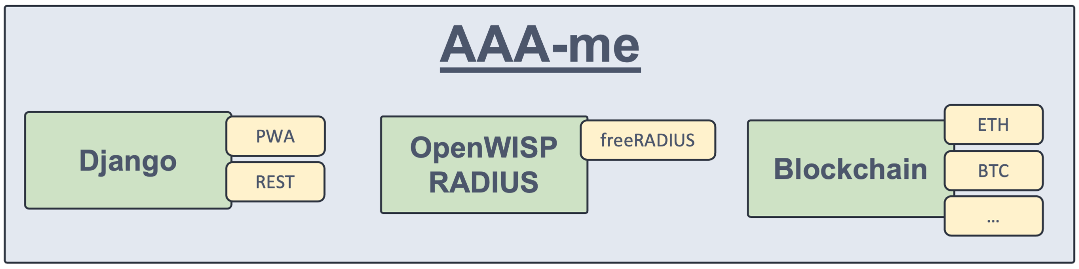


Abbildung A.2: Gesamtübersicht über die verwendeten Frameworks

an, um zu einem RADIUS Client oder Server eine Verbindung aufzubauen. Auch kann hiermit eine Konfiguration durchgeführt werden, um auf die entsprechenden Systeme zugreifen zu können. Zusätzlich gibt es eine Schnittstelle zur Datenbank des RADIUS Servers, sodass unmittelbar auf diese eingegriffen werden kann.
Das letzte Modul beschreibt die Interaktionen mit den Blockchain-Technologien. Hier werden die Schnittstellenbeschreibungen definiert und umgesetzt. Möchte man eine neue Blockchain-Technologie integrieren, so muss an dieser Stelle eine neue Beschreibung hinzu programmiert werden.

Experiments

Befürworter von Blockchain-Technologien sehen in der Technologie eine Möglichkeit, den sogenannten Single Point of Failure (SPOF) zu reduzieren. Dieser besagt, dass ein System gefährdet ist, wenn eine zentrale Instanz, z. B. zur Steuerung oder Verwaltung eines Systems, vom System getrennt oder anderes ausgeschaltet wird. Dabei ist es unerheblich, ob diese Abschaltung mutwillig oder ungewollt auftreten. Um diese Behauptung prüfen zu können, wird in dieser Arbeit zwei Untersuchungen durchgeführt. Die erste Untersuchung versucht durch simultane häufige Authentifizierungsanfragen die Resilienz gegen Denial of Service (DoS) Angriffe zu messen. Die zweite Untersuchung ist analytischer Natur und prüft die Angriffsvektoren für Man-in-the-Middle (MITM) Angriffe. Beide genannten Angriffstypen sind die wesentlichen Angriffe in SPOF.
Zur Untersuchung der DoS Resilienz wird ein Demo-Set-up vorgeschlagen, um die Aufgaben eines RADIUS Systems auf jedem der drei verschiedenen Umsetzungen zu untersuchen. Zwei der Systeme sind Blockchain-gestützt und mit AAA-me konfiguriert. Das andere System ist ein freeRADIUS-System, das für den Betrieb auf Roaming-Basis konfiguriert wurde. In diesen Experimenten soll die Übertragung von Client-Anfragen an einen Server mit RADIUS simuliert werden. Zu diesem Zweck werden Python-Skripte erstellt, die einer Anfrage an den Server ähneln. Es stehen drei Konfigurationen zur Verfügung, die es einem, zwei oder vier Clients erlauben, gleichzeitig mit dem Server zu interagieren. Die Aufrechterhaltung einer Verbindung wird nicht geprüft. Daher wird für jede Authentifizierungsanfrage nur eine Authentifizierungsanfrage gestellt. Um die Vergleichbarkeit mit Blockchain-Konfigurationen zu gewährleisten und Ungleichheiten aufgrund von Verzögerungen bei Internetverbindungen o. Ä. zu vermeiden, wird ein lokaler Knoten erstellt. Der Zusammenbruch des RADIUS-Servers ist bei diesen Experimenten nicht zu erwarten, da zu wenige Authentifizierungsversuche gleichzeitig stattfinden dürften. Eine Verzögerung des Authentifizierungsprozesses sollte jedoch in den Daten erkennbar sein.
FreeRADIUS Password Authentication Protocol (PAP)-Authentifizierungen werden im ersten Teil untersucht. Dabei soll ein möglicher negativer Einfluss von AAA-me nachgewiesen oder widerlegt werden. In einem zweiten Schritt werden die Interaktionen mit der Blockchain untersucht, wobei die Kategorien und Konfigurationen weitgehend gleich bleiben, um eine Vergleichbarkeit herzustellen.
Es gibt einige Einschränkungen, die sich aus den oben genannten Testszenarien ergeben. Der Großteil der Untersuchungen wird auf einem MacBook in einer lokalen virtuellen Umgebung durchgeführt. Eine Aus-

führung über das Internet findet nicht statt. Außerdem sind die Konfigurationen nicht immer an reale Situationen angepasst, sondern um ein Verhalten gezielt untersuchen zu können. Im Fall der Prävention bedeutet dies, dass absichtlich zahlreiche Anfragen gleichzeitig generiert werden, ohne dass die Auslastung überprüft oder das System ausgeglichen wird. Da jedoch sowohl Blockchain- als auch Nicht-Blockchain-Untersuchungen gleich strukturiert sind, können Erkenntnisse gewonnen werden. Diese Erkenntnisse können durch nachfolgende Untersuchungen weiter verifiziert oder widerlegt werden.
Jedes Experiment wird zweimal durchgeführt. Mit und ohne AAA-me Konfiguration neben RADIUS. Zudem werden zu den jeweiligen Authentifizierungsanfragen noch einmal in gleicher Anzahl Anfragen mit falschen Daten durchgeführt. In gleicher Reihenfolge werden die Tests für die Blockchain Instanzen durchgeführt:

1. FreeRADIUS Local Authentications (10, 100, 1.000, 10.000, 100.000 Anfragen)
2. FreeRADIUS Roaming Authentications (10, 100, 1.000, 10.000, 100.000 Anfragen)
3. FreeRADIUS Roaming Authentications mit 2 RADIUS Clients (10,100,1.000,10.000, 50.000, 100.000 Anfragen)
4. FreeRADIUS Roaming Authentications mit 4 RADIUS Clients (10,100,1.000,10.000, 25.000, 100.000 Anfragen)

Ergebnisse und Diskussion

Die Ergebnisse der Untersuchungen zwischen reinen RADIUS Konfigurationen und diesen mit AAA-me ergaben exakt gleiche Werte. Daher kann davon ausgegangen werden, dass AAA-me keine negativen, wie auch positiven Auswirkungen auf die Netzwerkkonfigurationen hat. Es verbleibt die Unterscheidung von RADIUS Systemen in klassischer Konfiguration im Vergleich zur Konfiguration mit Blockchain Systemen. Dafür können zwei Tabellen herangezogen werden.

Tabelle A.1: Ergebnisse des Experiments zum Zeitaufwand mit FreeRADIUS mit AAA-me-Einrichtung in Sekunden.

Authentication AAA-me	**10**	**100**	**1,000**	**10,000**	**100,000**
Local	2.05	13.43	129.22	1,302.63	12,925.54
Hop, 1 Client	4.00	26.49	255.06	2,561.45	25,508.22
Hop, 2 Clients 192.168.10.11	12.05	26.52	255.27	2,553.73	25,470.09
Hop, 2 Clients 192.168.10.21	4.07	26.50	255.19	2,555.81	25,480.89
Hop, 4 Clients 192.168.10.11	4.06	26.48	254.66	2,550.69	25,466.87
Hop, 4 Clients 192.168.10.21	4.11	26.56	255.16	2,549.37	25,466.87
Hop, 4 Clients 192.168.10.22	4.11	26.58	255.08	2,549.77	25,516.82
Hop, 4 Clients 192.168.10.23	4.14	26.59	255.13	2,546,47	25,516.94

Die Tabelle A.1 zeigt die gemessenen Zeiten im RADIUS Set-up mit AAA-me Konfiguration. Wie zu erwarten steigt durch komplexere Set-ups die notwendige Zeiten zur Verarbeitung der Anfragen. In der Beschreibung bedeutet Hop, dass eine Roaming Verbindung eingestellt wurde, der Nutzer seine Anfrage somit über einen Proxy an ein weiter entferntes Netzwerk übertragen hatte. Weiter entfernt bedeutet in diesem Kontext die Verbindung über einen Proxy. Dabei unterscheiden sich die Zeiten, bis auf einer Ausnahme, nicht von den jeweiligen Maschinen.
In der Tabelle A.2 findet man die gemessenen Zeiten für die Konfiguration mit Blockchains. Dafür wurde ein Smart Contract installiert, der eine ähnliche Passwortauthentifikationsmethode umsetzt, wie sie im RADIUS Set-up verwendet wurde. Dies wurde durch die notwendige Vergleichbarkeit der Studien durchgeführt. Der entsprechende Smart Contract darf so nicht in Produktivsystemen verwendet werden, da hier personenbezogene und sensible Daten öffentlich im Ledger gespeichert werden.

Tabelle A.2: Ergebnisse des Experiments zum Zeitaufwand mit Blockchain-Einrichtung in Sekunden.

Authentifizierung mit Blockchain	**10**	**100**	**1,000**	**10,000**	**100,000**
Local Node, HTTP	0.09	0.91	9.37	91.81	1,011.09
Local Node, WS	0.08	0.76	15.82	100.38	1,144.39
Remote Node, LAN, HTTP	0.10	1.03	9.91	96.28	1,020.63
Remote Node, LAN, WS	0.08	0.78	15.30	88.58	1,226.26
Remote Node, Internet, HTTP	0.17	1.13	11.98	126.55	1,180.25
Remote Node, Internet, WS	0.26	2.31	19.54	89.92	681.03

Deutlich wird im Vergleich beider Tabellen, dass die Konfiguration mit der Blockchain um ca. den Faktor 13 besser ist, als die klassische Konfiguration. Schaut man sich die expliziten Daten der Zeitmessungen genauer an, stellt man fest, dass dies primär in den Authentifizierungsanfragen mit falschen Informationen geschuldet ist. Hier hat die Blockchain-Lösung eine deutlich schnellere Umsetzung gezeigt. Aber auch in den Anfragen mit richtigen Benutzernamen und Passwort-Kombinationen konnte die Blockchain-Lösung effizienter ausführen.
Die in der Tabelle rot markierten Stellen sind der Websocket-Schnittstelle zur Blockchain Node geschuldet. Diese hat eine Überlastübungsmechanik integriert, die bei zu vielen Anfragen automatisch den Zugriff schließt. Dadurch konnten zwar schnellere Zeiten gemessen werden, die Outputs waren jedoch fast alle mit einer Fehlermeldung behaftet. Daher können die rot markierten Zeiten nicht in die Bewertung des Vergleichs einbezogen werden.

Potenzial gegen MITM Angriffe

In der Abbildung A.3 können die verschiedenen Interaktionsschnittstellen vom RADIUS System mit einer Blockchain betrachtet werden. Betrachtet man jedoch aktuelle Schutzmaßnahmen gegen MITM Angriffe, könnte man schlussfolgern diese Methoden auf einer Blockchain zu implementieren. Dies liegt daran, dass Gegenmaßnahmen gegen MITM in der Regel von einer Verteilung der Autorität und von kryptografischen

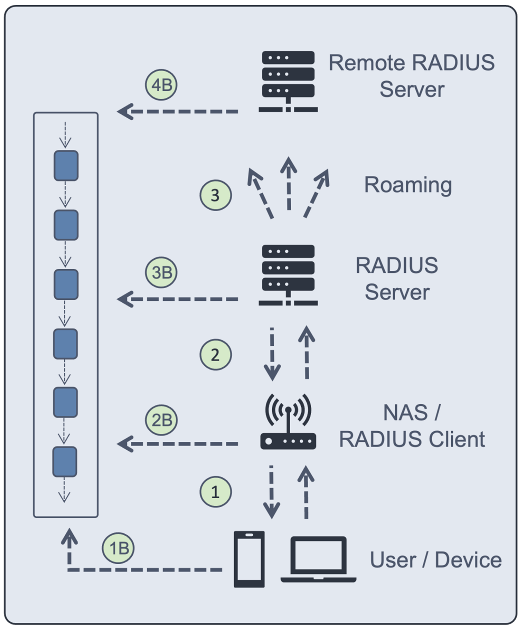


Abbildung A.3: Schnittstellen von RADIUS und Blockchain-Systemen.

Zertifikaten abhängen. Blockchain unterstützt diese Maßnahmen maßgeblich, durch ihre inhärente Dezentralisierung und der möglichen Anwendung als eine Public Key Infrastructure (PKI).
Mit Blick auf die Grafik A.3 stellt man jedoch fest, dass ein potenzieller Angreifer deutlich mehr Angriffsmöglichkeiten hat. Sollte dieser es schaffen, die Interaktion zur Blockchain zu manipulieren, so stünde die RADIUS Instanz schlecht dar.
Eine Blockchain kann demnach nicht allein als Lösung für das MITM Problem angesehen werden. Dazu ist sie selbst zu anfällig für Angriffe, z. B., wenn die Verteilung der Knoten zu gering ist. Dies wird jedoch gerade bei privaten Blockchain-Instanzen angenommen. So wurde beispielsweise in der Arbeit von [89] gezeigt, dass die Verteilung der Knoten essenziell für die Vermeidung von Angriffen auf die ETH-Blockchain ist. Die gleiche Bedeutung der Knotenverteilung wird in der Arbeit von [250] erwähnt, in der ein schwer zu entdeckender Partitionsangriff auf BTC durchgeführt wird. Daher wird eine signifikante Anzahl von Knoten als Gegenmaßnahme genannt.
Trotzdem kann eine Blockchain genutzt werden, um die stark ausgeprägten P2P-Elemente zu nutzen und eine MITM-resistentere Umgebung zu schaffen. Die Transparenz und Unveränderlichkeit der Daten sind zwei Eigenschaften, die einen Angriff nahezu unmöglich erscheinen lassen, zusätzlich zur Dezentralisierung. Darüber hinaus enthält die Technologie häufig kryptografische Elemente in ihrer Implementierung, die in eigene Systeme eingebettet werden können. Aufgrund dieser Modularität scheint die Implementierung von Blockchain eine geeignete Option zu sein, um MITM-Angriffe zu verhindern. Dennoch gibt es noch zahlreiche unbeantwortete Fragen, die in weiteren Untersuchungen bestätigt werden müssen.

Umsetzung in der Hessischen Zentrale für Datenverarbeitung (HZD)

Um eine Schnittstelle der beiden Fragen vom Anwendungspotenzial für öffentliche Verwaltungen und der Implementierung des RADIUS Protokolls zu erhalten, wurde in einer Laborumgebung ein weit genutztes System der HZD genutzt und mit AAA-me eine Blockchain zur Interaktion konfiguriert. Dieses System ist die FISBOX, welches als standardisierte und anpassungsfähige Software-as-a-Service (SaaS) Lösung für Fachinformationssysteme dient. Es ist ein zentraler Bestandteil des Onlinezugangsgesetzes (OZG) und ein wichtiger Bestandteil der Digitalisierungsstrategie des Landes Hessen. Die bereits hohe Verfügbarkeit der FISBOX® innerhalb des HZD in zahlreichen Projekten macht es möglich, durch eine durchdachte und sichere

Implementierung und Gestaltung eine Schnittstelle zur Blockchain mit AAA-me bereitzustellen. So können alle Projekte, die bisher auf der FISBOX® liefen, auf die Blockchain portiert werden. Darüber hinaus kann die FISBOX® durch weitere Implementierungen um die Blockchain erweitert werden, was ihren Funktionsumfang deutlich erhöhen würde. Die hohe Akzeptanz ermöglicht einen geringeren Entwicklungsaufwand bei größerem Integrationspotenzial. Erfolgreich wurde am Beispiel eines Smart Garden Forschungsprojekts diese Integrationsmöglichkeit gezeigt.

In Hessen wird derzeit ein Netz von drahtlosen Netzzugangspunkten aufgebaut, die es den Mitarbeitern, aber auch Gästen ermöglichen, über die bereitgestellten WLAN-Router auf das Internet zuzugreifen. Voraussetzung für den Zugang ist lediglich eine WLAN-fähige Hardware. Sie müssen sich bei der Verbindung zum Netz über ein Captive Portal authentifizieren. Mit dem Aufbau einer Verbindung erklärt sich der Nutzer mit den Nutzungsbedingungen einverstanden. Das Fehlen einer vollständigen Authentifizierung und die Anforderung, dass alle Personen, denen Zugang gewährt wird, die Vereinbarungen unterschreiben müssen, lassen ein RADIUS-Protokoll als übertrieben erscheinen. Auch andere Bereiche des Anwendungsbereichs von AAA werden von RADIUS bereitgestellt, sind aber hier nicht erforderlich.
Da es mit FreeRADIUS, der von AAA-me verwendeten RADIUS-Protokollimplementierung, leicht möglich ist, ein Captive Portal einzurichten, ist eine technische Realisierung möglich und damit denkbar. Dennoch bleibt der Gedanke, dass für die Umsetzung bereits eine Technologie verwendet wird und die Verwendung von RADIUS an dieser Stelle nicht notwendig ist.

Als letzte Integrationsmöglichkeit wurde die Anwendung des RADIUS-Protokolls durch AAA-me für die physische SSO-Funktionalität erforscht. Diese Zertifikate ermöglichen eine passwortlose Authentifizierung, indem die entsprechenden geheimen Schlüssel auf der Hardware gespeichert werden. Bei einem Challenge-Response- oder ZKP-Verfahren wird der Besitz dieses Schlüssels überprüft. Im Ledger selbst können durch dieses Verfahren zusätzliche Verknüpfungen hergestellt werden, die den Besitz oder die Berechtigungen erleichtern. So kann die von einem Unternehmen zur Verfügung gestellte Hardware dem Mitarbeiter durch Zertifikate zugewiesen werden. Wird dem Mitarbeiter der Zugang gewährt, erhält auch die mit ihm verknüpfte Hardware automatisch den Zugang. Die Integrationsmöglichkeiten von FreeRADIUS mit bestehenden Benutzerverwaltungssystemen, wie Active Directory oder LDAP, ermöglichen den Zugriff auf der physischen Ebene. AAA-me ruft dann Informationen aus dem Hauptbuch ab und speichert sie in den entsprechenden Systemen.
Bei öffentlichen Verwaltungen könnte dies dazu führen, dass die Anfrage nach Zugang zu einem Gebäude einer anderen Verwaltung gleichzeitig die Berechtigung für die Hardware der Internet-Infrastruktur liefert. Für IT-Prüfungen, wie sie unter anderem das BSI durchführt, ließe sich dieser Ansatz schnell und effizient umsetzen.

Zusammenfassend kann festgehalten werden, dass die Integration bzw. Implementierung einer Blockchain in allen drei Bereichen möglich ist. Die Notwendigkeit der Integration ist jedoch noch zu untersuchen. Über die Bedeutung der Dezentralisierung wird in späteren Kapiteln noch mehr gesagt. Ein Grund für eine Blockchain kann mit dem bereits gesehenen Potenzial einer erhöhten Resilienz gegenüber SPOFs gegeben werden. Neben dem Verbesserungspotenzial birgt die Blockchain-Technologie aber auch Gefahrenpotenziale. Dazu können Nebenwirkungen gehören, die heute nur schwer zu finden sind. Bei einem vollständig öffentlichen Blockchain-System besteht die Gefahr der Überlastung. Eine Reihe von Maßnahmen zur Bewältigung der entsprechenden Bedrohungen ist nicht nur nützlich, sondern auch zwingend erforderlich, um die Sicherheit der Daten zu gewährleisten. Die Idee eines Distributed Ledger erscheint für die öffentliche Verwaltung vielversprechend. Allerdings muss noch viel Zeit und Mühe investiert werden, um das System gegen Bedrohungen zu schützen und es sicherer zu machen.

Appendix B

JSON Interpretation of a Bitcoin Transaction

Shown here is a JSON representation of a validated Bitcoin transaction from block 477230 [39]. The ID from this transaction is the hash value and can be found in line 52. This transaction was validated on 07/23/2017 in the mentioned block. The time is given in line 45 as the Unix timestamp.
Essential information can be divided into three areas of a transaction. We have the input information from line three to 18, which refers to the address whose bitcoins are to be sent. In the output part in lines 22-41 the addresses are described on which the bitcoins are sent proportionally. As a rule, a so-called change account is included there. This account serves as a residual money recovery because the bitcoins from an address cannot be sent proportionally. So that one does not lose the change, one sends this to a separate address. For security reasons, this is not the same address itself, although that would be technically possible. The last information area is split between lines 2, 23-25 and 46-53 in the JSON representation. Line two defines the version of the transaction and thus the possible transaction type. Lines 23 to 25 provide information about the block that validated the transaction. Lines 46 to 53 give metadata about the transaction itself.

```json
{
    "ver":1,
    "inputs":[
        {
            "sequence":4294967294,
            "witness":"",
            "prev_out":{
                "spent":true,
                "tx_index":269632701,
                "type":0,
                "addr":"1GLctvTi81GDYZF5F6nif2MdbxUnAGHATZ",
                "value":4920000,
                "n":5,
                "script":"76a914a83fc0fb6edecb5289ef3e333520e4235bdf8a8788ac"
            },

            "script":"473044022036be6403aeb4e0e6fd54720b328d9d81bea32fb79684da0288743668fb5ef3ee
            02202023a71ef7217061fb9b4f35a05143de71447032e5a35b39c3d14b3210bad10b
            0121032725846bb7bc2e47b7b5a50670d77c8268f4d7f3243bdcf1b22174a67faaf528"

        }
    ],
    "weight":900,
    "block_height":477230,
    "relayed_by":"178.79.179.49",
    "out":[
        {
            "spent":true,
            "tx_index":270018614,
            "type":0,
            "addr":"1AG24pctoCpEMfSvEKfUZqaGYnz8ey8BfF",
            "value":3744000,
            "n":0,
            "script":"76a914659042e01e864e2f29641ea3a213c51a956d33c788ac"
        },
        {
            "spent":true,
            "tx_index":270018614,
            "type":0,
            "addr":"18Gj3unkApi17yh24JF6WwhN635SfABFMj",
            "value":1018920,
            "n":1,
            "script":"76a9144fc238bcda3f884ff6ce8d9feeb89b50dfd3da8888ac"
        }
    ],
    "lock_time":477228,
    "size":225,
    "double_spend":false,
    "time":1500839662,
    "tx_index":270018614,
    "vin_sz":1,
    "hash":"b657e22827039461a9493ede7bdf55b01579254c1630b0bfc9185ec564fc05ab",
    "vout_sz":2
}
```

Code B.1: Transaction on Bitcoin in JSON

Appendix C

Bitcoin Script Example

Table C.1: Verification process of an Bitcoin address (P2PKH). The direction is from top to down. The stack is at the beginning empty. The script shows the actual by Bitcoin nodes broadcasted scripts if a transaction is verified. After step one the script is putting the signature and the public key into this stack. In step 2 the public key is added another time and will be hashed in step 3 with the 'OP_Return' command. In step 4 the broadcasted hash value is added to the stack which will be verified with the generated hash value in step 5. In step 6 the signature will be verified. At the end all items of the stack are dropped and the result will be placed.

Source: [122]

Step	Stack	Script	Description
1	Emtpy	<sig><pubKey>OP_DUP OP_HASH160<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
2	<sig><pubKey>	OP_DUP OP_HASH160<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
3	<sig><pubKey> <pubKey>	OP_HASH160<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
4	<sig><pubKey> <pubKeyHashA>	<pubKeyHash>OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
5	<sig><pubKey> <pubKeyHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
6	<sig><pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
7	true	Emtpy	Signature is checked for top two stack items.

There is a second table as an example in chapter 2.5 Verification process of a Bitcoin address (P2SH) on page 18, showing a Pay-to-Script-Hash (P2SH) transaction script.

Appendix D

Private Ethereum Network

Ethereum is enabled and facilitated by numerous client software solutions. Moreover, those clients facilitate the effortless implementation of a private network. The advantages and disadvantages of private blockchain networks are discussed in 6.4.3 6.4.3 discussed. Nonetheless, particularly in the domain of research and testing, the advantages are evident.
An overview of used Ethereum (ETH) client software can be found in 2.2.3 2.2.3. Since the change from Proof-of-Work (PoW) to Proof-of-Stake (PoS), there are two versions of clients. The consensus and execution clients. For this work, only the Execution Clients are interesting, since they already support two consensus methods of type PoW and Proof-of-Authority (PoA) out of the box. Hence, one software is sufficient to establish a private network. The most widely used client software is Geth, and therefore also the one with the best documentation. Therefore, this software was utilized for the execution of the thesis. The name comprises the components Go and Ethereum, as this particular version was written in Go.
The PoW supported by the client "Geth" is also called “ethhash”. This version is different from other PoW systems, such as that of Bitcoin. It is hoped that this will result in a certain degree of autonomy, which may also result in a heightened level of security. To achieve this objective, Ethereum aimed to employ ASIC-resistant processes, i.e., consensus processes that are incapable of being accelerated by hardware. This increases the need for graphics cards, as this hardware was the most effective on the Ethereum network. However, all current Ethereum clients are capable of this version of PoW, allowing participation in the main network or a private network.
The PoA version is called Clique. This is different from other PoA systems, such as Authority Round (Aura). This has significant implications for potential development decisions, since these clients cannot interact with other networks. Therefore, if the decision favors PoA, the explicit procedure would have to be chosen, and thus software would have to be chosen. However, client diversification suffers as a result. An explanation of the respective consensus procedures can be found in the chapter 2.2.2 Consensus Procedures.

D.1 Installation and setup

```
sudo apt install software-properties-common
sudo add-apt-repository -y ppa:ethereum/ethereum

sudo apt-get update
sudo apt-get install ethereum
```

A complete list and description of all installation methods can be found on the Geth website [91]. It also lists the hardware and operating system requirements that are supported. The described variant below is valid for Ubuntu 22.04 with ARM64 architecture.

D.2 Set up a private network

No matter which consensus is chosen, the procedure in both implementations is very similar. Now, two variants are exemplarily shown to set up a private network. A PoW and a PoA network are set up. Here, many inputs are characterized variably with < variable >.
No matter which consensus is chosen, the procedure in both implementations is very similar.

1. Create a Genesis file to define the Genesis block.
2. Initialize the database from the client with the Genesis block.
3. Create or import a wallet (private-public key pair).
4. Start the node.
5. Connect Node to the P2P network, respectively to the boot node(s).
6. Start mining/signing.

The genesis files are explained in the chapter D.2.4 Genesis-Files, and examples are provided. There is also an explanation in the developer documentation, where a private network is explained. Furthermore, you can find a troubleshooting site, which can help you resolve issues.
It is advisable to exercise caution when selecting the path name, as Geth will create a folder if it does not already exist. It is a frequent occurrence that upon altering the directory and initiating the start command from memory, a new uninitialized folder is created and initiated from that location. In this case, you would be out of sync on your private network and unable to connect to the P2P network.
The next two sections will discuss how to set up the respective networks.

D.2.1 Initialize an Ethash Node

A PoW network is to be set up. First of all, it should be clear where the nodes are to be distributed. If they are on the same machine, you have to pay attention to the port distribution and folder directories.
In the following, we assume that one has created a folder for the administration of the Geth node (e.g., ETH) and that one is located in it. The necessary Genesis file is also located there. In this directory, you then call:

```
geth init −−datadir ./data ./<name−of−gensis>.json
```

The init command creates a folder data where the blockchain data is stored. The first block, the so-called Genesis block, is already located here.
In addition to the data directory, a keystore directory is also created. This is where the data for managing the accounts (private-public key pairs) administered by Geth is stored.
Later, a geth.ipc file will be created in the root directory, which is needed later to manage the node.

For an Ethash system, an account does not have to be created and managed in Geth. It is sufficient to have an address available later on. We therefore begin with two different node types. A boot node would be one of the nodes. This is a reliable instance that can serve as a starting point for the Peer-to-Peer (P2P) network. Other peers are then exchanged and linked. The other node would be a member node. This is fed with the bootnode information to get a direct start into the network.
It is important to know that the port on TCP and UDP must be enabled for all nodes. The default setting is 30303, but it can be set to any value with the –port flag.

- Bootnode:

```
geth --datadir ./data/ --networkid 37 --identity '<NAME>' --ethstats <Identity>:<WS_SECRET>
     --http --http.port <PORT> http.api "<APIs>" --authrpc.port <PORT>
```

- Membernode

```
geth --datadir ./data/ --networkid 37 --identity '<NAME>' --ethstats <Identity>:<WS_SECRET>
     --http --http.port <PORT> http.api "<APIs>" --authrpc.port <PORT> --bootnodes <ENODE
    or ENR>
```

Explanation of the flags in the geth calls:

- datadir
  Specifies the path for storing key and blockchain data. If the folder at the end of the path does not exist, it will be created. This can quickly lead to lack of connectivity of the nodes and should be considered.

- networkid
  Here, the ID from the genesis file is named. This is important for Geth so that at the beginning, when the nodes are looking for each other, they can quickly determine whether the nodes match each other or not. The default value is 1 and represents the main Ethereum layer. There are lists where you can voluntarily list your chains [61]. It is important to note that for the nodes to be successfully linked, the genesis state must still be absolutely the same.

- identity
  Geth shares information about your node, such as which version you are using. Additionally, you can set an alias if you want to see your node recognized on the network.

- port
  The default port for exchanging data and locating nodes is 30303. If you want to use a different one, you can set it with the –port flag. It is important to note that both TCP and UDP are affected and must be enabled in the firewall.

- http
  The HTTP flag sets the release of an interface. If you want to control your node from outside your machine, it must be released. Even if other services, such as the Web IDE Remix or the block explorer Blockscout would like to be used, the interface should be released. This makes so-called Remote Procedure Call (RPC) calls possible, which the service providers or one needs. Also, the Web3 library (available for various programming languages) needs these RPC methods to execute transactions or to omit information. For security, one can set which interfaces are to be made available. A non-exhaustive list due to constant changes and further development of Geth is "txpool, ethash, eth, net, web3, admin".

- ethstats
  These are the credentials for network monitoring. The chapter D.3 Eth-Netstats Monitoring will go into more detail.

- authrpc.port
  Nodes communicate via RPC calls. Port 8551 is used by default. If you want to run multiple nodes on one machine, you have to enter a new value here.

- bootnodes
  This command is intended for member nodes. With this command, you can enter a list of nodes to which your node should initially connect and download the blockchain data. Subsequently, additional nodes are sought. The bootnode serves as a so-called springboard into the P2P-network to make oneself known. Thus, one can use the so-called enode. However, it is also possible to utilize a so-called bootstrap node record. The encoding for this data begins with "enr" and is a base64-encoded string.

If you execute one of the two commands, a geth.ipc file is created in the data directory of the initialization. You can use this file to call a JavaScript console to control the node. Settings that can also be a flag in the call of the node, like starting or stopping the mining. If one would rather not call the console, a command can also be executed directly with the flag –exec.
In addition, there is the possibility, if one has provided the HTTP interface, to access the console remotely. For this purpose, the address of the machine is used.

```
geth attach ./data/geth.ipc --exec <COMMAND>
geth attach http://<HTTP-RPC-IP>:<HTTP-RPC-PORT> --exec <COMMAND>
```

For a network, it is important that blocks be minted. It is therefore necessary to connect to the node in the console as described above. If you have not already done so, you must set the Coinbase address. This is the address to which the reward will be sent when a block is found. After the Coinbase has been set, you can start mining. If you would rather not mine with full CPU power, you can set the number of threads.

```
miner.setEtherbase("<Adresse>")
miner.start(2) # Mine with 2 Threads
```

D.2.2 Initialize a Clique Node

To set up a private Ethereum network using the Clique consensus method, analogous commands are needed as described above. However, there are a few small things that need to be considered:

1. If no addresses have been created yet, you have to create an account. The public address is needed to define a first signer in the Genesis file (step 2). An account can be created with the following command:

   ```
   geth --datadir ./data account new
   ```

   The command will create a directory with all keys to be managed by geth. A key file containing the secret key will then also be created there. This file should be protected with a password, which will be requested when it is called.

2. The Genesis file is structured differently. See the section D.2.5. Clique - PoA.

3. At least one node must start the account defined with the public key in the Genesis file. To accomplish this, you need a file that contains the private key (without formatting, just copy-and-paste). This can then be imported with the following command:

   ```
   geth account import keyfile
   ```

4. Now the initialization and the start of the node can take place analogously to Ethash. One exception is the node that has imported the public key of the Genesis file as an account. The account must be activated when the node is started.

```
geth <other−flags> −−unlock <Public−Address> −−mine

# In Console:
personal.unlockAccount(eth.accounts[0], "<PASSWORD>", <TIME−UNLOCKED>)
```

The mine flag at the end starts the mining and thus the creation of new blocks. Mining can also be started in the console. The time for which an account remains unlocked is specified in seconds.

5. Since you have an unlocked account in the PoA variant, it makes sense to add another flag if you want to release the HTTP interface. Otherwise, it remains locked.

```
geth <other−flags> −−allow−insecure−unlock
```

D.2.3 Set up a fork

Updates to the consensus layer are referred to as forks. See the chapter 2.2.2 Updates and Forks for more details. In order to execute such a fork, it is imperative that a vote be conducted among all participants. So-called Improvement Proposals are currently used in both Bitcoin and Ethereum (Bitcoin Improvement Proposals (BIP) and Ethereum Improvement Proposals (EIP)) . The procedure here in the test set up is not essential.We anticipate a coordinated update. This update was implemented, and with the introduction of a new version of the client software, it is theoretically accessible for all nodes. However, in order for the network to coordinate the start, the Genesis file must be adjusted. Most updates receive a name. This name must be specified in the configuration with a block time that is in the future. In the genesis file that serves as an example in code D.1, the Constantinople update is set to block time 200,000 in line 8. If this time is in the future, and you are ahead of the current block with the update, then re-calling the initialization can unlock the update. All nodes would be required to perform this task in order to maintain consensus. It is inevitable that the blockchain state of a node will differ from the network over time if it does not apply the update. This is called a fork.

D.2.4 Genesis-Files

Ethash - PoW

The referenced code D.1 describes the origination file for an Ethash network. You will notice that there are corresponding blocks that have to perform different tasks.
The first block extends from line two to line eleven. The configuration of the chain is described, which means that the consensus procedure is defined and that the respective updates are scheduled. As mentioned in the blockchain chapter 2.2 Blockchain Technologies, the time flow in blockchain systems is represented in blocks. For example, the update EIP 150 in line five from the example code D.1 is activated in block 1000. Starting in block 1000, all changes described in EIP 150 will be active.
In line 10, the consensus algorithm is chosen. If one wants to build a PoS-system, one needs a so-called consensus client, which is necessary for the consensus tasks. Furthermore, however, an executing client such as Geth is required. The advantages of a PoS-system compared to a PoA-system are debated. Therefore, the work is limited to PoW and PoA systems.
The parameters for the genesis block are defined on lines 12 to 19. Numerous parameters pertain to the structure of a typical block, thus requiring default configurations for their formation. The genesis file allows you to define the genesis block, and a check of these parameters is not checked by a node. However, all defaults must be kept starting from the next block. A comprehensive account of all essential parameters can

be obtained from the yellow paper [263]. There are all the specifications in it, which are necessary, should one try to develop a client.
The nonce along with the mixhash must satisfy conditions to provide proof of work. This ailment is referred to as the Difficulty in line 14. The mixhash is supposed to be smaller than a certain threshold. With an alternating nonce (incremental), the miner manages to find a suitable nonce for a mixhash over time. The value of the Difficulty should not be set too high at the beginning, otherwise the first nodes may need too long to find a suitable block. Furthermore, an adjustment is made per block if the average time is under or exceeds.
The coinbase field described in line 15 describes an address. The reward for finding the block goes to this address. At this point, the miner will enter an address that the user can also manage, i.e., own the private keys to. For the Genesis block, you can also choose an address that does not have a private key. In that case, the first reward would not be distributed.

```
1  {
2    "config": {
3      "chainId": 37,
4      "homesteadBlock": 0,
5      "eip150Block": 1000,
6      "eip155Block": 2000,
7      "eip158Block": 5000,
8      "byzantiumBlock": 100000,
9      "constantinopleBlock": 200000,
10     "ethash": {}
11   },
12   "nonce": "0x63687269616d7565",
13   "mixhash": "0x69756c69696167756b0000000000000000000000000000000000000000000000",
14   "difficulty": "0x100",
15   "coinbase": "0x29FF4F5004BeCe6Bdeb47DD761D1782Ee18E21d9",
16   "timestamp": "0x0",
17   "parentHash": "0
        x0000000000000000000000000000000000000000000000000000000000000000",
18   "extraData": "0x",
19   "gasLimit": "0x8000000",
20   "alloc": {
21     "0x89ad0c7a4a544b2c75bf6cef730fd80b5a1feb38": {"balance": "
          1000000000000000000000"},
22     "0xe4CC5920f23E8d8726F48d05b8262be66a04FABd": {"balance": "300000" },
23     "0xf41c74c9ae680c1aa78f42e5647a62f353b7bdde": {"balance": "400000" },
24     "0xf54DFE141A3a02BD6a0a85d50360B044449C9aB5": {"balance": "10123012398" }
25   }
26 }
```

Code D.1: Example Genesis-File for Ethash PoW

The timestamp is defined in the Unix standard and attempts to establish the time on all nodes. This requirement is necessary in order to perform the difficulty adjustment per block.
The parent hash typically originates from the preceding block. Nevertheless, since the Genesis block does not have a predecessor, the 0 is usually predefined here. This is the sole block that is capable of possessing such a parent hash.
Besides the consensus choice, the extraData field is the main difference between Ethash and Clique. In Ethash, this field is solely capable of incorporating additional information into the genesis state. An analogy to Bitcoin, where a New York Times headline was written into the first block, here one can provide a

proof that the block was not created earlier. In order to rule out pre-production, proof would be necessary. A potential fraudster could define a genesis state but have already calculated hundreds of subsequent blocks. After a certain time, this person could then publish their longer chain on the network and change all previously validated blocks to non-valid ones. In the instance used for the thesis, no validation had to be considered. Thus, the value is set to 0. For a clique network, the address of a signer would be required in this field.
The maximum gas limit specifies how large a block may become. This value can be changed by the miners at a later time. As a miner, you can define a gas limit yourself. Each miner takes care of this, and with each block that's minted, the sum of the old and new values is compared, and the limit is adjusted. The adjustment has a maximum or minimum value that is determined in relation to the current value.
The last block starts in the code D.1 at line 20 and defines the so-called allocation. This concept is commonly referred to in the literature as pre-mine, which assigns the credits to the designated value prior to the discovery of a block. Therefore, these credits are created independently of the mining process and are often criticized. Numerous projects utilize this function to provide developers with credit, which is highly contingent on the value of the respective tokens or coins. This block is optional and can also be omitted.

D.2.5 Clique - PoA

```
{
  "config": {
    "chainId": 108,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "clique": {
      "period": 25,
      "epoch": 30000
    }
  },
  "difficulty": "1",
  "gasLimit": "8000000",
  "extradata": "0x0000000000000000000000000000000000000000000000000000000000000000
      ↪  e4CC5920f23E8d8726F48d05b8262be66a04FABd0000000000000000000000000
      ↪ 00000000000000000000000000000000000000000000000000000000000000000
      ↪ 0000000000000000000000000000000000000000",
  "alloc": {
    "0xe4CC5920f23E8d8726F48d05b8262be66a04FABd": {"balance": "300000" },
    "0x29FF4F5004BeCe6Bdeb47DD761D1782Ee18E21d9": {"balance": "400000" }
  }
}
```

Code D.2: Example Genesis-File for Clique PoA

A detailed explanation of the structure of the genesis file can be found in the section D.2.4 Ethash - PoW. Here, only the differences betweenPoW and PoA genesis files are explained. The primary distinctions between

the two genesis files pertain to the configuration and the extraData field. Logically, the configuration of the used consensus must be set to clique. There exist two parameters that necessitate definition within it. The period is the time between two blocks in seconds, and the epoch is there for the division of the signers in the interval to protect against cheaters. For larger networks with a high number of participants, the default value is 30000.
In line 20, the extraData field is essential. The address or addresses of the signers must be contained. To accomplish this, it is necessary to add 64 zeros in the front and 130 zeros in the back. The quantity of addresses is not restricted. It is possible to create a Genesis file using generators, such as puppeth. But also templates, like the genesis D.2 below, can be taken and replaced by copy-and-paste with your address.
This address must be stored as an account on at least one node. Either you had previously established an account prior to generating the genesis file and subsequently copied the address out, or alternatively, you imported the private key by utilizing the appropriate geth function.

D.3 Eth-Netstats Monitoring

If you have set up a private network, it is necessary to monitor it. ETH offers Eth-Netstats for this purpose. You can easily install this and have your node connected as Geth flags. This is one of many possible applications. You can also easily pull the data from your node and build your own representation. However, since Eth-Netstats is strongly integrated into Geth and provides a comprehensive representation of important information, it is recommended to use it as well. One should still note that the tool has not been updated for a while and therefore security gaps can appear. For a productive system, it would be advisable to build up an own monitoring, preferably integrated in own security systems.
Prerequisites and install Eth-Netstats:

```
sudo apt install nodejs npm git
git clone https://github.com/ethercore/eth-netstats
cd eth-netstats
npm install
```

Now you can start your instance. For security reasons, it is recommended to specify a port and a password for connecting. This can be done directly in the startup call of the instance. For this, you have to be in the directory of Eth-Stats:

```
   PORT=<Port> WS_SECRET=<Secret> npm start
```

In order for the custom Geth node to connect to the Eth-Netstats instance, the following flags must be set on the startup command:

```
   geth <other-flags> --ethstats <Identity-Name>:<WS_Secret>@<IP>:<Port>
```

D.4 Blockscout Blockexplorer

In addition to monitoring, an explorer can prove to be a valuable tool when seeking to conduct a thorough examination and investigation of a blockchain network. An Eth-Stats monitoring tool displays the nodes, whereas Blockscout looks at and displays the transactions. It contains important information about all transactions validated in the block, transaction types, and the associated byte code of deployed smart contracts. One can thus gain insight into the data and the current state of the blockchain through an explorer, and this information is crucial in the development of further smart contracts or in monitoring the proper execution of those already implemented.

Furthermore, there are many other explorers that could be used. Blockscout provides a convenient method for the quick setup of its own instance using Docker containers, which can also communicate with its own private blockchain. Consequently, it proves to be advantageous in my particular scenario as well.
The following installation instructions can be found on the project's GitHub site [36].
It is possible for the explorer to run on its own machine and connect to a remote node via RPC connection, or you can choose to run the explorer on a separate machine. Compared to separate hosting, the latter solution has the advantage of better performance, since Explorer can get the data directly via the Java-Script console and does not need to connect over the Internet.
To install it, you download the GitHub project to your computer. In the project, there is a folder with Docker Compose files. In this folder, you will find another folder envs where environment variables can be defined. Here you can enter the access to the node and a description of the network.

```
git clone https://github.com/blockscout/blockscout

# If you want to change environment variables:
nano blockscout/docker—compose/envs/common—blockscout.env

# Now you can start the docker compose to run the system
cd blockscout/docker—compose/
docker compose up ——build
```

You can make many settings to the Blockscout Explorer using an Environment Variable file. Below is an excerpt of this file. There you can set the connection to the node, the images, and connections to other services. An overview of the variables can be found in the following link [37].

```
# DOCKER_TAG=
ETHEREUM_JSONRPC_VARIANT=geth
ETHEREUM_JSONRPC_HTTP_URL=http://<IP—or—Hostname:Port>/
DATABASE_URL=postgresql://postgres:@host.docker.internal:7432/blockscout?ssl=false
ETHEREUM_JSONRPC_TRACE_URL=http://<IP—or—Hostname:Port>/
NETWORK=Ethereum
SUBNETWORK=<Name—of—your—Chainporject>
LOGO=/images/blockscout_logo.svg
LOGO_FOOTER=/images/blockscout_logo.svg
ETHEREUM_JSONRPC_TRANSPORT=http
ETHEREUM_JSONRPC_DISABLE_ARCHIVE_BALANCES=false
IPC_PATH=
NETWORK_PATH=/
API_PATH=/
SOCKET_ROOT=/
BLOCKSCOUT_HOST=
BLOCKSCOUT_PROTOCOL=
# SECRET_KEY_BASE=
# CHECK_ORIGIN=
PORT=4000
COIN_NAME=<Your—Coin—Name>
[...]
```

Appendix E

Private Bitcoin Network

Bitcoin, as the first blockchain project, was copied by many other projects and thus served as a template for future blockchain projects. Nonetheless, it is not an effortless task to obtain a copy of such software by oneself. In contrast to Ethereum, where a new chain can be created by utilizing a Genesis.json file, it is necessary to extract the program code from Bitcoin and modify all configurations to ensure that the compilation remains functional. The degree of complexity involved in this matter is exceedingly high. Fortunately, there exists a function within the Bitcoin core referred to as regtest. With this, the developer is able to create a local instance with function calls. This instance is not comparable to a private chain with Ethereum and should therefore not be preferred over this solution. However, a regtest network can be used to simulate a private network. This provides testing and development opportunities. If one wishes to remain with the Bitcoin technology, one must go through recompilation. Also, there are no other consensus procedures in Bitcoin, besides the Proof-of-Work (PoW) as in Ethereum. In regtest mode, only a PoW can be simulated, but this does not come close to the security of the PoW or other consensus procedures. Bitcoin remains a system that was designed as a currency. Further adjustments should be operated with other technologies.

E.1 Bitcoin Regtest Network-Node

There are several ways to set up a node. As with any program, the opportunity of compiling the program code is possible. In the spirit of decentralization, this should also be considered, especially considering the implications that a buggy or compromised piece of software could have. There are also pre-compiled software packages. Under an Ubuntu system, any PPA entries are present. However, Docker containers can also be found and used to get the Node running. Below is one way to use a PPA solution on Ubuntu [221]

```
sudo apt-get install build-essential
sudo apt-get install libtool autotools-dev autoconf
sudo apt-get install libssl-dev
sudo apt-get install libboost-all-dev
sudo add-apt-repository ppa:luke-jr/bitcoincore
sudo apt-get update
sudo apt-get install bitcoind # This will install just the bitcoin daemon tool. If you want to
    have a GUI, you need additional installations.
```

To start a node in the regtest, you can add the command addition (flag) -regtest. However, this makes the further calls cumbersome. Also, further changed parameters would have to be changed with each function call. In order not to have to bother with the further parameters, one can store the appropriate values into a configuration file. This is under Linux systems in a folder, which must be created first. Also, the configuration file must be created in this directory.

In the Bitcoin repository, the file gen-bitcoin-conf.sh can be found. The following command can be executed to create a sample config:

```
mkdir ~/.bitcoin

# For generating a template bitcoin.conf file, you can use the line below.
BUILDDIR=$PWD bitcoin/contrib/devtools/gen-bitcoin-conf.sh

# Create a username and password for rpc-connection. Note the given output.
bitcoin/share/rpcauth/rpcauth.py <Username> <Password>

# You can just create a file with nano or other text editors
nano ~/.bitcoin/bitcoin.conf
```

Before editing the configuration file, you should first create an authentication. You enter a username and password and the output is the hash of the password to a given salt (random value as protection against so-called reverse hash tables). This line is written down and added to rpcauth (line 8 in the code E.1). The script to create it can be found in the Bitcoin repository, as well as the configuration generator. Running it with username and password will generate the required information.

```
1  regtest=1 # State, that we want to start in regtest mode.
2  daemon=1 # We dont want to see the console of the bitcoin Daemon
3  server=1 # We want that the node is accessable remotly
4  listen=1 # We want that the node can get input from other nodes
5  txindex=1 # We want to be able to connect other services and nodes with this node. This is
       neccessary.
6  rpcuser=<USER-NAME>
7  rpcpassword=<USER-PASSWORD>
8  rpcauth=<USER-NAME>:<GIVEN-SALT>$<HASH-of-PASSWORD>
9  rpcallowip=0.0.0.0
10 # Set a explicit defined IP or a block of IPs for connection.
11 rpcallowip=194.163.134.0/24
12
13 # [wallet]
14 # A fee rate (in BTC/KB) that will be used when fee estimation has insufficient data
15 fallbackfee=0.0005
16
17 # [network]
18 # Specify your public IP address.
19 externalip=<Public-IP>:<Port|18443>
20
21 [regtest]
22 # add a connection to a server
23 addnode=<IP-Remote-Node>:<Remote-Port|18443>
24
25 rpcbind=0.0.0.0
26 zmqpubrawblock=tcp://<Public-IP>:<Port|28332>
27 zmqpubrawtx=tcp://<Public-IP>:<Port|28333>
```

Code E.1: Example bitcoin configuration file

With the configuration file described in the code E.1, we can start a node in regtest mode that connects directly to a remotely started node.Additional security measures for connections to our node via Remote Procedure Call (RPC) are entered. IPs have been set and should be set even more restrictively in the event of a startup. For testing and development purposes, these settings are sufficient. The accesses defined in lines 22 and 23 are needed for alternative services like the Lightning network. If Lightning is not to be used, these two lines can be removed.
With the following commands, you can start the node now.

```
bitcoind
```

E.1.1 Explorer and Monitoring

Compared to the solution mentioned for Ethereum, here we discuss a possibility that solves monitoring and explorer functionalities in one. Mempool.space can be used to monitor the network as well as view transaction details. The setup can be done on other machines as well, provided shares have been set in the respective Bitcoin nodes. Docker also makes it straightforward to set up. [188]

```
# First get the project on your machine by cloning the repository.
git clone https://github.com/mempool/mempool.git

# Changing some settings.
nano /mempool/docker/docker-compose.yml

# Run the docker compose
docker compose up
```

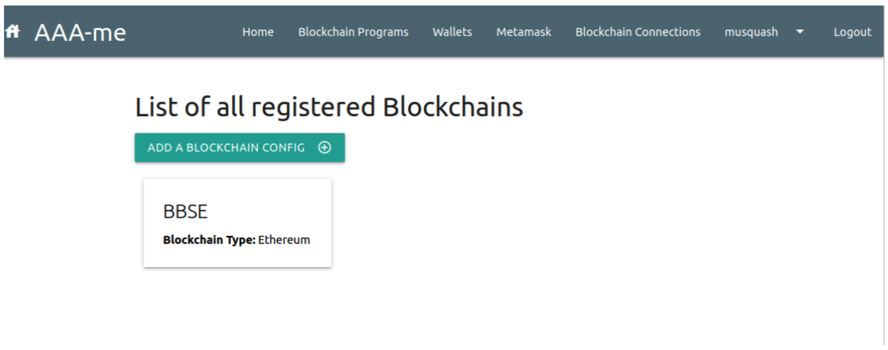
In the mentioned docker compose file, the corresponding data must be adjusted. Further settings are also possible, or necessary, if you want to connect an Electrum server to get a wider range of functions.

```
  api:
    environment:
      MEMPOOL_BACKEND: "none"
      CORE_RPC_HOST: "<Your-Remote-IP"
      CORE_RPC_PORT: "<Port>"
      CORE_RPC_USERNAME: "<Username-from-Bitcoin.conf>"
      CORE_RPC_PASSWORD: "<Password-from-Bitcoin.conf>"
```

Appendix F

AAA-me Screenshots

This appendix shows pictures of the program AAA-me. These pictures have been adapted to the code descriptions so that an immediate representation of the program can be shown.

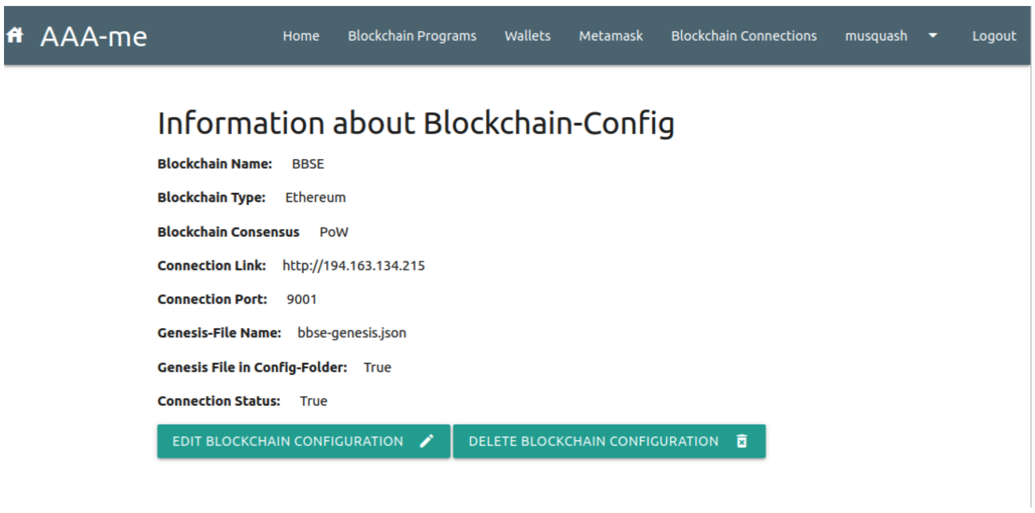


List of all registered Blockchains

BBSE

Blockchain Type: Ethereum

(a) Overview of Blockchain Connections



Information about Blockchain-Config

Blockchain Name:	BBSE
Blockchain Type:	Ethereum
Blockchain Consensus	PoW
Connection Link:	http://194.163.134.215
Connection Port:	9001
Genesis-File Name:	bbse-genesis.json
Genesis File in Config-Folder:	True
Connection Status:	True

(b) Detail View of a Blockchain Connection

Figure F.1: Blockchain Connections

Appendix G

Setting up Virtual Machines with Virtual Networks

This section describes how to set up a virtual network. This is used to simulate several networks to enable remote authentication. These remote authentications can then be examined.
An Apple 14" Macbook with M1 Max CPU, 64 GB main memory and a 2TB SSD is used. Host operating system was Ventura 13.2.1.
Apple's Parallels software was used for parallelization, in the Pro license.
The network was built purely on functionality. This means that this is not a handout for secure networks. For that, other security settings have to be taken into account.
The iptables package was used for routing. However, the default under Server 22.04 is nftables. Since the documentation for iptables exceeds that of nftables in number and comprehensiveness, iptables was retained for this tutorial.

G.1 Setting up a Virtual Network

This section describes how the virtual network is set up. Three Virtual Machines (VMs) (router, client, server) with different functions are set up for each virtual network, simulating a company. An Ubuntu Server 22.04 ARM version is used as the virtual operating systems. This has the advantage that significantly fewer resources must be provided for the respective machines. If necessary, a GUI can be installed afterwards. Images of other operating systems can also be installed to simulate connecting other devices.
An example architecture of the two company networks can be seen in the figure G.1.
In the virtual network, the three VMs should each fulfill the tasks of their names. The server is a Remote Authentication Dial-In User Service (RADIUS) server and the client can be any client. Two company networks are simulated, identified by the invented names BENBU and ARA. The steps are analogous for both networks. Therefore, only an exemplary and general setup is described here. At the end you will find the configurations of all machines.

1. The first step is to create a network in the virtualization that can connect the machines. In Parallels, the 'Host-Only' mode is considered, since a system is created that is isolated from other networks. Parallels takes care of the DHCP functions for us. The IP addresses for the networks can be arranged. It is 192.168.10.0/24 for the fictional company BENBU and 192.168.20.0/24 for the fictional company ARA for my purposes.

2. The total of six VMs must be set up. To do this, download the image from the provider and use the installation wizard of the virtualization software. It is important to have an Internet connection to complete the installation successfully. To do this, we connect the VM to a network with Internet

access for installation. There is a possibility to clone the VM in many virtualization solutions. In this case, the image of the instance is copied. If you then change the MAC address(es) of the machine, you can save the installation of all other machines.

3. After the installation, the network must be configured. First the net-tools package should be installed. This will give you the necessary access and control over the network cards. When setting up the VM router, one more network card must be set. This can be done in the configurations. The appropriate company network (BENBU or ARA) is then selected here. So that the VM starts automatically with the necessary network settings, netplan is used. The configuration file is located in / etc / netplan / by default and can be configured with the example described below.

```
sudo apt install net−tools

ifconfig # overview of active networks
ifconfig −a # overview of all networks (on and off)
```

Configuration for client and server:

```
# Setting up the netplan configuration:
sudo nano /etc/netplan/*.yaml

# Extract from netplan−config:
network:
    ethernets:
        <Interface−WAN>:
            dhcp4: true
            gateway4: <IP−Virtual−Network−Router>
            nameservers:
                addresses: [8.8.8.8, 8.8.4.4]
    version: 2
    renderer: networkd

# After saving activate netplan configuration:
sudo netplan apply
```

Following is the netplan configuration for the router:

```
sudo nano /etc/netplan/*.yaml

# Config of netplan:
network:
    ethernets:
        <Interface−WAN>:
            dhcp4: true
            dhcp4−overrides:
                route−metric: 100
        <Interface−VLAN>:
            dhcp4: true
            dhcp4−overrides:
                route−metric: 200
            routes:
            - to: 0.0.0.0/24
              via: <WAN-IP-Router>
            nameservers:
                addresses: [8.8.8.8, 8.8.4.4]
    version: 2
    renderer: networkd

# After saving activate netplan configuration:
sudo netplan apply
```

4. After the networks have been set up, the port shares and forwarding (natting) must be set up on the router VMs. The general release of port forwarding for IPv4 must also be allowed.

```
    # For port forwarding uncomment the line with net.ipv4.ip_forward=1
    sudo nano /etc/sysctl.conf
```

```
    sudo iptables -t nat -A POSTROUTING -o <Interface-WAN-Router> -j MASQUERADE
    sudo iptables -A FORWARD -i <Interface-WAN-Router> -o <Interface-LAN-Router> -m state
          --state RELATED,ESTABLISHED -j ACCEPT
    sudo iptables -A FORWARD -i <Interface-LAN-Router> -o <Interface-WAN-Router> -j
         ACCEPT
```

Now pinging on the other networks as well as pinging internal pages should be possible. An overview of the set routes of the respective machines is possible with `route -n`.

G.2 Example Configuration for Virtual Networks used for Experiments

The configurations with the respective defined IPs and network interfaces from the experiments used are now described here. This can help to better understand the settings to simplify own setups. At this point it should be repeated that these networks have no security settings. Accordingly, these are not sufficient instructions for networks that are to be used.
First, the networks have to be set in Parallels. To accomplish this, go to the network settings (discoverable via the settings of an explicit VM) and add another interface with the plus. The type must be "Host-Only". This restricts the host to DHCP functions, i.e., it is disconnected from the Internet. In the IP settings, you can then enter the address range you want to set for your network. For this, I have the address ranges 192.168.10.0/24 for BENBU and 192.168.20.0/24. The address 192.168.10.0 or 192.168.20.0 is the network prefix, which identifies all hosts or devices in this network, while the /24 prefix is the subnet mask, which identifies which part of the IP address is the network part and which is the host part.
The /24 prefix indicates that the first 24 bits of the IP address form the network identification prefix, while the last 8 bits form the host identification prefix. This means that up to 256 hosts or devices can be connected in the address range 192.168.10.0 to 192.168.10.255.

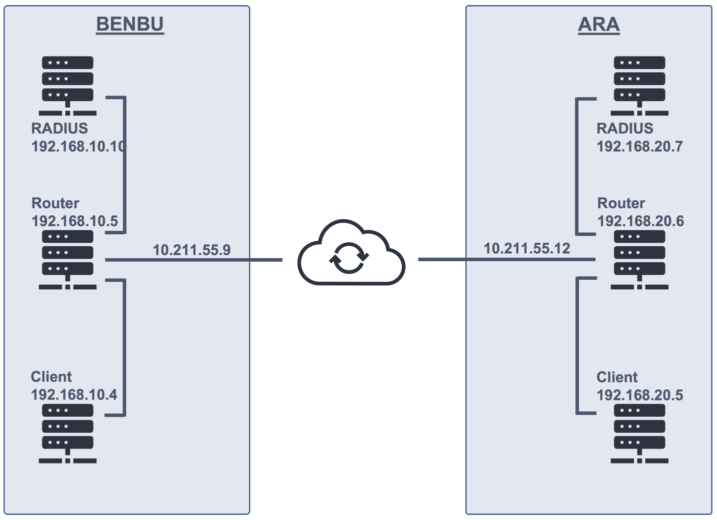


Figure G.1: Virtual network architecture - On the left is the fictitious BENBU network, on the right is ARA. All IPs have been given by a DHCP Server.

Virtual BENDU network

BENBU Client (IP: 192.168.10.4):

```
sudo apt install net−tools
# open netplan configuration:
sudo nano /etc/netplan/00−installer−config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
ethernets:
   enp0s5:
      dhcp4: true
      gateway4: 192.168.10.5
      nameservers:
         addresses: [8.8.8.8, 8.8.4.4]
version: 2
renderer: networkd

# Apply netplan configuration
sudo netplan apply
```

BENBU RADIUS (IP: 192.168.10.10):

```
sudo apt install net−tools
# open netplan configuration:
sudo nano /etc/netplan/00−installer−config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
ethernets: enp0s5:
   dhcp4: true
   gateway4: 192.168.10.5
```

```
      nameservers:
          addresses: [8.8.8.8, 8.8.4.4]
  version: 2
  renderer: networkd

# Apply netplan configuration
sudo netplan apply
```

BENBU Router (IP: 192.168.10.5, WAN: 10.211.55.9):

```
sudo apt install net-tools
# open netplan configuration:
sudo nano /etc/netplan/00-installer-config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
    ethernets:
        enp0s5:
            dhcp4: true
            dhcp4-overrides:
                route-metric: 100
        enp0s6:
            dhcp4: true
            dhcp4-overrides:
                route-metric: 200
            routes:
            - to: 0.0.0.0/0
              via: 10.211.55.9
            nameservers:
                addresses: [8.8.8.8, 8.8.4.4]

    version: 2
    renderer: networkd

# Apply netplan configuration
sudo netplan apply

# For port forwarding uncomment the line with net.ipv4.ip_forward=1
sudo nano /etc/sysctl.conf

# Enabling Natting and port forwarding
sudo iptables -t nat -A POSTROUTING -o enp0s5 -j MASQUERADE
sudo iptables -A FORWARD -i enp0s5 -o enp0s6 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i enp0s6 -o enp0s5 -j ACCEPT
```

Virtual ARA network

ARA Client (IP: 192.168.20.5):

```
sudo apt install net—tools
# open netplan configuration:
sudo nano /etc/netplan/00—installer—config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
ethernets: enp0s5:
   dhcp4: true
   gateway4: 192.168.20.6
   nameservers:
      addresses: [8.8.8.8, 8.8.4.4]
version: 2
renderer: networkd

# Apply netplan configuration
sudo netplan apply
```

ARA RADIUS (IP: 192.168.20.7):

```
sudo apt install net—tools
# open netplan configuration:
sudo nano /etc/netplan/00—installer—config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
ethernets: enp0s5:
   dhcp4: true
   gateway4: 192.168.20.6
   nameservers:
      addresses: [8.8.8.8, 8.8.4.4]
version: 2
renderer: networkd

# Apply netplan configuration
sudo netplan apply
```

ARA Router (IP: 192.168.20.6, WAN: 10.211.55.12):

```
sudo apt install net—tools
# open netplan configuration:
sudo nano /etc/netplan/00—installer—config.yaml
#edit netplan config:
# This is the network config written by subiquity
network:
   ethernets:
      enp0s5:
         dhcp4: true
```

```
            dhcp4-overrides:
                route-metric: 100
        enp0s6:
            dhcp4: true
            dhcp4-overrides:
                route-metric: 200
            routes:
            - to: 0.0.0.0/0
              via: 10.211.55.12
            nameservers:
                addresses: [8.8.8.8, 8.8.4.4]

    version: 2
    renderer: networkd

# Apply netplan configuration
sudo netplan apply

# For port forwarding uncomment the line with net.ipv4.ip_forward=1
sudo nano /etc/sysctl.conf

# Enabling Natting and port forwarding
sudo iptables -t nat -A POSTROUTING -o enp0s5 -j MASQUERADE
sudo iptables -A FORWARD -i enp0s5 -o enp0s6 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i enp0s6 -o enp0s5 -j ACCEPT
```

Make iptables rebootable

As mentioned at the beginning of the section, iptables is not preconfigured. Thus, after a reboot, all rout settings would be removed again. To preserve the settings after a reboot, iptables-save must be installed and set up.

```
sudo mkdir /etc/iptables
sudo apt-get install iptables-persistent

# Write current iptables into rules file.
sudo sh -c "iptables-save > /etc/iptables/rules.v4"


sudo nano /etc/network/if-pre-up.d/iptables

# Fill in the following in the file:
#!/bin/sh
/sbin/iptables-restore < /etc/iptables/rules.v4

# Set file executable
sudo chmod +x /etc/network/if-pre-up.d/iptables
```

Appendix H

Setting up Roaming in RADIUS

This appendix describes how to set up roaming with freeRADIUS. This requires at least two freeRADIUS servers that act differently and must be set up in advance. For this, you can use the attachment G Setting up Virtual Machines with Virtual Networks to set up two virtual networks on which a freeRADIUS server can be installed.

H.1 Install freeRADIUS

Detailed and up-to-date installation instructions can be found in the source [202].

```
sudo apt-get install freeradius
```

H.2 Roaming

This tutorial is based on the two fictitious company networks from appendix G Setting up Virtual Machines with Virtual Networks. So all information for the own networks must be adapted accordingly.
Roaming is the option of RADIUS that the servers can communicate with each other. In this case, the locally controlled RADIUS server takes over the function of an RADIUS client and forwards the authentication request according to its setting and data specified by the user.

In order for two FreeRADIUS servers to establish a connection and permit roaming queries, it is imperative to establish a server-to-server connection, commonly referred to as a "proxy" or "Federated RADIUS". Here are the basic steps:

1. Client file Configuration: Open the file `/etc/freeradius/3.0/clients.conf` and add their information about the remote FreeRADIUS server machine. Here is an example from the BENBU RADIUS server:

   ```
       client ara_radius {
          ipaddr = 192.168.20.7
          secret = testing123
          nastype = other
       }
   ```

   Accordingly, the file must be set up on the other server ARA, to build a connection to BENBU.

2. Proxy file Configuration: Open the file `/etc/freeradius/3.0/proxy.conf` and add their information about the remote FreeRADIUS server machine. Here again an example from the BENBU RADIUS server

```
    realm ara_radius {
        authhost = 192.168.20.7:1812
        accthost = 192.168.20.7:1813
        secret = testing123
    }
```

3. Configuration of the Server Assignment: Open the file `/etc/freeradius/3.0/sites-enabled/default` and add information about the remote FreeRADIUS server machine there.

```
    pre-proxy {
        if (Packet-Src-IP-Address == 192.168.20.7) {
            update control {
                Proxy-To-Realm := "ara_radius"
            }
        }
    }
```

4. Reboot FreeRADIUS:

```
sudo systemctl restart freeradius
```

5. Testing a Roaming Request:

```
radtest <Username> <Password> <IP-Address of the other server> <Auth-Port> <shared
    secret>
```

H.3 Roaming with Hops

The most widespread RADIUS application is the eduroam network (see 2.7.1). This allows members of academic and research institutions to log in with their user data at participating sites worldwide. In this network, RADIUS servers are connected in series. This can be seen schematically in the figure H.1. This section describes how such a system was simulated for the experiments.
For the tutorial, we will assume three servers. We will call these A, B and C. The ARA network is used for A and accordingly the BENBU network for B. The abbreviation is used for a cleaner and denser code illustration.

1. Server A Configuration: Open `/etc/freeradius/3.0/clients.conf` in the home directory of freeradius and modify it as follows to allow Server B to communicate with Server A:

```
client server-B {
    ipaddr = 192.168.10.10 # IP-Address_Server_B
    secret = shared_secret
    nastype = other
}
```

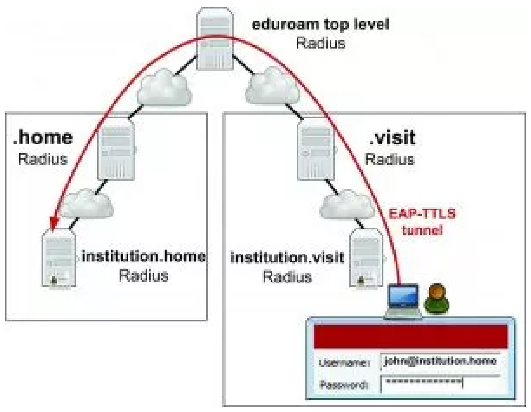


Figure H.1: Authentication Process in eduroam. [143]

Domains Description: Afterward, a domain description must be set as proxy. Open the file `/etc/freeradius/3.0/proxy.conf` for that, on Server A.

```
realm benbu.com {
    auth_pool = server—B—pool
    acct_pool = server—B—pool
}

home_server_pool server—B—pool {
    type = fail—over
    home_server = server—B
}

home_server server—B {
    type = auth+acct
    ipaddr = 192.168.10.10 # IP—Address_Server_B
    port = 1812
    secret = shared_secret
    response_window = 20
    max_outstanding = 65536
}
```

2. Server B Configuration: Open `/etc/freeradius/3.0/clients.conf` on Server B. Here you must allow the connection to A and C:

```
client server—a {
ipaddr = 192.168.20.7 # IP—Adresse_von_Server_A
secret = shared_secret_A
nastype = other
}
client server—c {
    ipaddr = 192.168.30.3 # IP—Adresse_von_Server_C
    secret = shared_secret_C
    nastype = other
}
```

Server B is intended to be the trusted root instance for forwarding authentication request to the corresponding domain.

Domains Description: On server B, two domain descriptions must be set as proxy. Open the file `/etc/freeradius/3.0/proxy.conf` for that, on Server B.

```
realm clara.com {
auth_pool = server−c−pool
acct_pool = server−c−pool
}

home_server_pool server−c−pool {
    type = fail−over
    home_server = server−c
}

home_server server−c {
    type = auth+acct
    ipaddr = 192.168.30.3 # IP−Adresse_von_Server_C
    port = 1812
    secret = shared_secret_C
    response_window = 20
    max_outstanding = 65536
}
```

3. Server C Configuration: Open `/etc/freeradius/3.0/clients.conf` in the home directory of freeradius and modify it as follows to allow Server B to communicate with Server C:

```
client server−b {
    ipaddr = 192.168.10.10 # IP−Address_Server_B
    secret = shared_secret
    nastype = other
}
```

The configuration shown allows authentication on the network of server A through a prior registration on server C.