Effiziente Signaturerzeugung durch Server-Unterstützung und Vorberechnung

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Mathematik
der Johann Wolfgang Goethe–Universität
Frankfurt am Main

von
Johannes Merkle
aus Kelkheim

Frankfurt 2000
(D F 1)

vom Fachbereich Mathematik der

Johann Wolfgang Goethe–Universität als Dissertation angenommen

Dekan: Prof. Dr. R. Bieri

Gutachter: Prof. Dr. C. P. Schnorr, Prof. Dr. M. Sieveking

Datum der Disputation: 4. Februar 2000

Vorwort

Es steht außer Zweifel, daß digitale Signaturen schon bald zu unserem Alltag gehören werden. Spätestens mit dem Inkrafttreten des Gesetzes zur digitalen Signatur (siehe [BMB]) sind sie zu einem wichtigen Instrument in der Telekommunikation geworden. Dabei kommt der Verwendung von Chipkarten eine wichtige Bedeutung zu: In ihnen lassen sich die sensiblen Daten (z.B. der geheime Schlüssel) auslesesicher aufbewahren; gleichzeitig können sie bequem mitgeführt werden. Aus diesen Gründen erlebt die Verwendung von Chipkarten zur Erzeugung von digitalen Signaturen zur Zeit einen enormen Aufschwung. Problematisch ist jedoch der oft unverhältnismäßig große Berechnungsaufwand für die Erzeugung von digitalen Signaturen.

Ziel dieser Arbeit ist es, Methoden zu entwickeln und/oder zu untersuchen, welche die Berechnung digitaler Unterschriften wesentlich beschleunigen. Dabei spiegelt sich die Zweiteilung der in der Praxis hauptsächlich verwendeten Typen von Signaturverfahren in der Struktur der Arbeit wider. Der erste Teil dieser Arbeit untersucht Verfahren zur effizienten Berechnung von RSA–Unterschriften. Dabei entstanden die Untersuchungen in den Abschnitten 3.2.3 und 3.2.4 in Zusammenarbeit mit R. Werchner und der Inhalt der Abschnitte 3.1 – 3.2.4 ist bereits in [MW98] veröffentlicht. Im zweiten Teil entwickeln wir Verfahren zur effizienteren Generierung von Unterschriften, die auf dem diskreten Logarithmus basieren, und untersuchen deren Sicherheit. Dabei entstanden die Untersuchungen in den Abschnitten 4.2 (bis auf 4.2.2) und 4.3.1 in Zusammenarbeit mit C. P. Schnorr und sind teilweise in [MS98] zusammengefaßt.

Obwohl diese Arbeit eine mathematische Abhandlung darstellt, versuchen wir, die praktische Anwendung nicht aus den Augen zu verlieren. So orientieren sich die betrachteten Verfahren stets an den durch die verfügbare Technologie gegebenen Rahmenbedingungen. Darüber hinaus richten wir unser Augenmerk weniger auf das asymptotische Verhalten der betrachteten Verfahren, als vielmehr auf konkrete, für die Anwendung relevante Beispiele.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Digitale Signaturverfahren zählen zu den asymmetrischen kryptographischen Verfahren: Jeder Benutzer besitzt ein Paar Schlüssel – einen geheimen und einen öffentlichen[1]. Mit dem geheimen Schlüssel kann er seine Daten derart versiegeln, daß ihre Urheberschaft und Unverfälschtheit jederzeit mit Hilfe des öffentlichen Schlüssels überprüft werden kann. Digitale Signaturen ermöglichen somit einen wirksamen Schutz gegen Manipulation von Daten und der Vorspiegelung falscher Identitäten bei einer Kommunikation über offene Netze sowie den Abschluß verbindlicher Vereinbarungen im elektronischen Geschäftsverkehr.

Die Sicherheit der wichtigsten digitalen Signaturverfahren beruht auf der Komplexität von zwei verschiedenen Problemen der algorithmischen Zahlentheorie: Das RSA-Verfahren [RSA78] sowie die Verfahren von Fiat und Shamir [FS87], von Rabin [Rab79], von Ong und Schnorr [OS90] und von Guillou und Quisquater [GUQ92] beruhen auf dem Problem, große Zahlen in ihre Primfaktoren zu zerlegen. Dieses Problem nennt man Faktorisierungsproblem. Die Verfahren von ElGamal [ElG85], Schnorr [Sch89, Sch91] und der Digital Signature Standard (DSS) [NIS94] beruhen dagegen auf dem Problem, in einer endlichen Gruppe G zu gegebenen $g, y \in G$ eine ganze Zahl z mit $g^z = y$ zu finden – sofern eine solche existiert. Dieses Problem ist als die Berechnung des diskreten Logarithmus bekannt. Wir bezeichnen im folgenden die auf diesem Problem beruhenden Verfahren als DL-Verfahren.

Die Erzeugung einer digitalen Signatur ist jedoch sehr rechenintensiv. In vielen Situationen stellt dieser große Berechnungsaufwand ein Problem dar. Ein Beispiel hierfür sind Chipkarten: selbst mit Hilfe eines kryptographischen Koprozessors benötigen sie z.B. für die Berechnung einer RSA–Unterschrift 0.5 - 1 Sekunde (siehe [HP98] oder [NM96]). In Anwendungen wie Homebanking oder E-Cash, wo oft mehrere Signaturen auf einmal geleistet werden müssen, wird der Berechnungsaufwand unverhältnismäßig groß. In diesen Anwen-

[1] Der öffentlichen Schlüssel sollte im Idealfall jedermann zugänglich sein

dungen benötigt man daher Methoden, welche die Erzeugung digitaler Unterschriften wesentlich beschleunigen. Auf der anderen Seite sind solche Methoden auch aus theoretischer Sicht interessant, weil sie die Entwicklung neuer Sicherheitsmodelle und Analysemethoden erfordern.

Bei den meisten der genannten Verfahren besteht der größte Teil des Berechnungsaufwandes in einer Exponentiation in einer endlichen Gruppe G. Während bei den DL-Verfahren die Basis konstant und der Exponent zufällig ist, hängt bei RSA die Basis von der zu unterschreibenden Nachricht ab und der Exponent ist konstant – es ist der geheime Schlüssel. Bei vielen Verschlüsselungs und Authentifikationsverfahren besteht der größte Aufwand ebenfalls in einer Exponentiation. Beispiele hierfür sind die Verschlüsselungsverfahren von ElGamal [ElG85], die RSA-Verschlüsselung [RSA78], sowie die Authentikationsverfahren von Schnorr [Sch89, Sch91], von Guillou und Quisquater [GUQ92] und von Ong und Schnorr [OS90].

Wir untersuchen in dieser Arbeit Verfahren zur effizienten Erzeugung von digitalen Signaturen. Dabei messen wir die Effizienz der betrachteten Verfahren durch die Anzahl der benötigten Multiplikationen.

1.1 Bekannte Ansätze zur effizienten Signaturerzeugung

Eine Exponentiation $a \rightarrow a^b$ wird üblicherweise durch fortlaufendes Quadrieren und Hinzumultiplizieren von a berechnet. Dieses Verfahren ist als *„square and multiply"* bekannt und benötigt $\lfloor \log b \rfloor + \mathrm{wt}(b) - 1$ Multiplikationen, wobei $\mathrm{wt}(b)$ die Anzahl der Einsen in der Binärdastellung von b ist. (Mit log bezeichnen wir immer den Logarithmus zur Basis 2.) Wird – wie bei den DL-Signaturen – b zufällig aus einem Intervall $(0, N)$ gewählt, so benötigt eine Exponentiation mit der „square and multiply"-Methode im Durchschnitt $1.5 \log N$ Multiplikationen. Bei RSA dagegen hängt die Anzahl der Multiplikationen allein von der Größe des geheimen Schlüssels ab. M. Wiener zeigte in [Wie90], daß ein geheimer RSA-Schlüssel mit weniger als $(\log |G|)/4$ Bits aus dem öffentlichen Schlüssel effizient rekonstruiert werden kann. Es wird deshalb allgemein empfohlen, die Bitlänge des geheimen Schlüssels nicht viel kleiner als $\log |G|$ zu wählen (siehe [MvOV97, f.288]).

Es gibt verschiedene Ansätze, um die Exponentiation a^b bei der Signatur-Erzeugung effizienter zu berechnen.

Ein Ansatz besteht in einer geschickteren Nutzung der während der Berechnung entstehenden Zwischenergebnisse: Während das „square and multiply"-Verfahren in jedem Rechenschritt nur das jeweils letzte Zwischenresultat und den Basiswert a verwendet, nutzen sogenannte *Additionsketten-Verfahren* beliebige zuvor berechnete Zwischenergebnisse. Eine Additionskette der Länge t ist eine Sequenz $x_1, \ldots, x_t$ von Zwischenergebnissen, wobei $x_1 = 1$ und $x_t = b$ gilt und es für alle $i > 1$ Indices $j_1, j_2 < i$ gibt mit $x_i = x_{j_1} + x_{j_2}$. Zu einer gegebenen Additionskette der Länge t zu b läßt sich die Exponentiation a^b mit $t-1$ Multiplikationen berechnen: In Schritt $i > 1$ berechnet man $g^{x_i} = g^{x_{j_1}} g^{x_{j_2}}$.

Ein Nachteil von Additionsketten-Verfahren ist ihr Speicherbedarf; alle Zwischenergebnisse, die in der weiteren Berechnung noch benötigt werden, müssen temporär gespeichert werden. Auf der anderen Seite ist die Länge einer Additionsketten zu einem Exponenten b mindestens $\log(b) + \log \mathrm{wt}(b) - 2.13$ (siehe [Sch75]), d.h. selbst mit einer optimalen Additionskette ist die Exponentiation höchstens 1.5 mal effizienter als das „square and multiply"-Verfahren

Die Erzeugung einer RSA-Signatur läßt sich durch Ausnutzung der Struktur der zugrunde liegenden Gruppe – die Gruppe der invertierbaren Elemente in $\mathbb{Z}_n$, wobei n das Produkt zweier großer Primzahlen p und q ist – beschleunigen. Aus dem Chinesischen Restsatz folgt, daß jedes Element x aus dieser Gruppe durch seine Anteile $x_p := x \bmod p$ und $x_q := x \bmod q$ eindeutig bestimmt ist. Falls p und q bekannt sind, läßt sich das Element aus diesen Anteilen sehr effizient berechnen: man setzt $x = x_p w_p + x_q w_q$ mit $w_p := q(q^{-1} \bmod p)$ und $w_q := p(p^{-1} \bmod q)$. Man kann daher zuerst $a^b \bmod p$ und $a^b \bmod q$ berechnen und dann daraus $a^b \bmod n$ zusammensetzen. Dieses Verfahren hat zwei Vorteile: zum einen kann man bei der Berechnung von $a^b \bmod p$ (bzw. $a^b \bmod q$) vor jedem Rechenschritt die Zahlen modulo p (bzw. modulo q) reduzieren und somit stets nur mit $\log p$ (bzw. $\log q$) Bit langen Zahlen rechnen. Zum anderen gilt nach dem Satz von Fermat $a^b = a^{b \bmod p-1} \bmod p$ und $a^b = a^{b \bmod q-1} \bmod q$. Man kann daher die Exponentiation mit den wesentlich kürzeren Exponenten $b \bmod p$ bzw. $b \bmod q$ ausführen. Falls man die Werte w_p und w_q vorher berechnet und abgespeichert hat, ist diese Berechnungsmethode etwa vier mal effizienter als die naive Methode.

Eine Möglichkeit die Exponentiation bei der Erzeugung von DL-Signaturen zu beschleunigen ist die Verwendung von *Precomputation*: Dabei wird eine Anzahl von Gruppenelementen fest gespeichert und der Exponentialwert a^b bei jeder Signaturerzeugung aus diesen Werten zusammengesetzt. Falls zum Beispiel – wie bei den DL-Verfahren – die Basis a konstant und der Exponent aus einem Intervall $(0, N)$ ist, kann man für eine Konstante $c \geq 2$ die Werte $a, a^c, a^{c^2}, \ldots, a^{c^k}$ mit $k := \lceil \log_c N \rceil - 1$ speichern und dann die Exponentiation durch $a^b = \prod_i a^{b_i c^i}$ berechnen, wobei $b_0 \ldots b_k$ die c-adische Darstellung von b ist, d.h. $b = \sum_i b_i c^i$ gilt. Ein solcher Ansatz benötigt zwar zusätzlichen Speicher, ist jedoch oft wesentlich effizienter als z.B. Additionsketten-Verfahren. In Abschnitt 4.1 stellen wir einige weitere Verfahren zur Exponentiation mit Precomputation für DL-Signaturen vor.

1.2 Neue Ansätze zur effizienten Signaturerzeugung

In dieser Arbeit untersuchen wir zwei völlig andere Ansätze zur effizienten Erzeugung digitaler Signaturen. Beide unterscheiden sich in einem wesentlichen Punkt von den oben beschriebenen Methoden: Der für die Signaturerzeugung benötigte Aufwand hängt nicht von der Größe der verwendeten Gruppe, sondern allein von der gewünschten Sicherheit (bezüglich des von uns verwendeten Sicherheitsmodells) ab. In vielen Fällen erreichen wir mit diesen Ansätzen wesentlich effizientere Verfahren zur Exponentiation. Auf der anderen Seite hängt bei diesen Ansätzen die Sicherheit einer Signatur nicht mehr allein von der Sicherheit des Signaturverfahrens, sondern auch von den Parametern des Exponentiationsverfahrens ab. Aus diesem Grund steht bei unseren Untersuchungen stets die Analyse der Sicherheit der vorgestellten Verfahren im Vordergrund. Diese Analysen basieren auf einem eingeschränkten Berechnungsmodell: das Modell der generischen Algorithmen. Dieses Modell wurde (in verschiedenen Formulierungen) von Nechaev in [Nec94] und Shoup in [Sho97] eingeführt.

Vereinfacht gesprochen nutzen generische Algorithmen keine speziellen Eigenschaften der Darstellung der Gruppenelemente durch Bitstrings aus. Sein Verhalten hängt zu jedem Zeitpunkt nur von den Kollisionen der zuvor berechneten Gruppenelemente ab. Ein Beispiel für generische Algorithmen ist der „Baby Step, Giant Step"-Algorithmus von Shanks zur Berechnung des diskreten Logarithmus. Wir stellen diesen Algorithmus in Abschnitt 2.2 vor. Siebmethoden wie das Number Field Sieve [LLMP90] sind dagegen nicht generisch.

Der erste Ansatz, den wir in dieser Arbeit untersuchen, ist die Verwendung von *Server-unterstützten RSA Protokollen*. Dabei nutzt ein Anwender (*der Client*) in einem interaktiven Protokoll die Hilfe eines leistungsfähigeren Computers (*der Server*) zur Erzeugung einer RSA-Signatur[2]. Dazu wählt der Client eine zufällige Zerlegung seines geheimen RSA-Schlüssels. Die einzelnen Teile kann er öffentlich machen, nicht aber die Art und Weise, in der aus ihnen der geheime Schlüssel zusammengesetzt ist. Während des Protokolls übergibt der Client dem Server den Modul, die zu unterschreibende Nachricht und die Teile seines geheimen Schlüssels. Mit diesen Werten führt der Server festgelegte Berechnungen aus und sendet die Ergebnisse zurück an den Client. Dieser setzt schließlich die Signatur aus den erhaltenen Ergebnissen zusammen. Server-unterstützte RSA-Protokolle sind so konzipiert, daß der aufwendigste Teil der Signatur-Berechnung vom Server durchgeführt wird.

Es wurden mehrere effiziente Server-unterstützte RSA-Protokolle veröffentlicht (z.B. [MKI89, MILY93, BQ95, KS93]); für typische Parameter ist die (vom Client) benötigte Anzahl von Multiplikationen bei diesen Verfahren um ein vielfaches niedriger als beim RSA-

[2] In vielen Anwendungen ist ein solcher Server – z.B. in Gestalt eines Kartenterminals – anwesend

Verfahren. Für keines dieser Protokolle konnte jedoch die Sicherheit bewiesen werden. In Kapitel 2 beweisen wir die ersten unteren Schranken für die Komplexität von Angriffen gegen Server-unterstütze RSA Protokolle. Wir verwenden dabei das Modell der generischen Algorithmen.

In einigen Fällen sind unsere unteren Schranken scharf, d.h. es gibt Angriffe durch generische Algorithmen, deren Komplexität unsere Schranken erreichen. Außerdem präsentieren wir eine neue Art von Angriffen gegen eine in [LL95] vorgeschlagene Variante von Server-unterstützten RSA Protokollen. Diese Angriffe basieren auf der effizienten Lösung von linearen Gleichungssystemen mittels Gitterbasenreduktion.

Unser zweiter Ansatz beschleunigt die Erzeugung von DL-Unterschriften: die *pseudozufällige Generierung von Präsignaturen.* Unter Präsignaturen verstehen wir Paare (r, g^r), wobei g ein Generator der Gruppe G ist. Wir entwickeln und untersuchen Verfahren, die aus einer Anzahl von gespeicherten Präsignaturen (r_i, g^{r_i}) in pseudozufälliger Weise neue Präsignaturen (r, g^r) erzeugen. Dabei wird der Exponent r jeweils durch eine zufällige Summe der r_i und der Exponentialwert g^r als das (der Summe) entsprechende Produkt der g^{r_i} berechnet. Die gespeicherten Präsignaturen werden zu Anfang zufällig gewählt und teilweise bei jeder Berechnung einer neuen Präsignatur transformiert. Die Verteilung der erzeugten Präsignaturen hängt bei diesen Verfahren sowohl von der zufälligen Wahl der gespeicherten Präsignaturen als auch von der zufälligen Wahl der Summe der r_i ab.

In analoger Weise kann auch die Erzeugung von Paaren (x, x^b) mit zufälligem x und festem b – wie sie bei Ong-Schnorr-Signaturen benötigt werden – beschleunigt werden. Dazu speichert man eine konstante Anzahl von Paaren (x_i, x_i^b) und berechnet x als zufälliges Produkt der x_i. Der Exponentialwert x^b ergibt sich dann durch entsprechende Produkt der x_i^b.

Bereits in [Sch89] stellte Schnorr einen Generator für pseudozufällige Präsignaturen vor. Leider stellte sich dieser als unsicher heraus (siehe [dR91]). Auch ein in [Sch91] vorgestelltes, verbessertes Verfahren wurde gebrochen (siehe [dR93, dR97]). Eine Vielzahl weiterer Generatoren wurde vom Autor dieser Arbeit in [Mer95] entwickelt und analysiert. Dabei beschränkten sich jedoch die Sicherheitsanalysen auf Computer-unterstützte Heuristiken. Kürzlich stellten Boyko, Peinado und Venkatesan in [BPV98] ebenfalls Generatoren für pseudozufällige Präsignaturen bzw. Paare (x, x^b) vor. Sie behaupteten, daß bei Verwendung dieser Generatoren die Signaturverfahren von ElGamal, Schnorr, Ong und Schnorr sowie das DSS sicher seien. Leider haben die Autoren die Beweise der entsprechenden Theoreme bis heute nicht veröffentlicht. Darüber hinaus sind die Generatoren aus [BPV98] weniger effizient als unsere Verfahren.

Wir entwickeln und analysieren in dieser Arbeit verschiedene Varianten eines Generators für pseudozufällige Präsignaturen für beliebige zyklische Gruppen. Bei den typischen Größenordnungen der DL-Signaturverfahren sind diese Verfahren erheblich effizienter als eine Exponentiation mit Precomputation oder Additionsketten-Verfahren. Insbesondere führen wir ein Maß SEC für die Sicherheit dieser Verfahren ein und leiten von diesem Maß untere Schranken für Angriffe durch generische Algorithmen ab. Unsere Analysen komplettieren wir dann durch explizite Schranken für den Wert von SEC.

Beim Einsatz unserer Generatoren auf elliptischen Kurven ergeben sich zusätzliche Gesichtspunkte. So kann man in diesen Gruppen z.B. sehr effizient Elemente invertieren. Indem wir diese Eigenschaft ausnutzen, erhalten wir wesentlich effizientere Generatoren. Wir behandeln die Generatoren für elliptische Kurven in Abschnitt 4.4 detailliert.

Kapitel 2

Grundlagen

2.1 Digitale Signaturen

Wir behandeln in diese Arbeit vor allem die beiden wichtigsten Signaturverfahren: RSA und DL-Verfahren. Da sich die DL-Verfahren untereinander sehr ähnlich sind, stellen wir von diesen nur das Schnorr-Signaturverfahren vor.

2.1.1 Das RSA-Signaturverfahren

Das RSA-Verfahren wurde 1978 von R. Rivest, A. Shamir und L. Adleman in [RSA78] veröffentlicht. Es kann sowohl zur Chiffrierung als auch für digitale Signaturen verwendet werden. Wir betrachten hier das Signaturverfahren.

Jeder Anwender A wählt sich zwei große verschiedene Primzahlen p und q und setzt $n := pq$. Außerdem wählt er sich einen geheimen Schlüssel $d \in \mathbb{Z}^*_{\phi(n)}$. Hierbei bezeichnet ϕ die Euler'sche Funktion, d.h. $\phi(n) = (p-1)(q-1)$. Zu diesem d berechnet er sich dann den öffentlichen Schlüssel $e = d^{-1} \bmod \phi(n)$. Schließlich veröffentlicht er n und e.

Wenn A nun eine Nachricht $m \in \mathbb{Z}^*_n$ unterschreiben möchte, berechnet er $y = m^d \bmod n$ und veröffentlicht y als die *Signatur von m*. Jeder Anwender kann nun mit Hilfe des öffentlichen Schlüssels die Korrektheit dieser Signatur überprüfen: Es gilt $y^e = m^{de} = m^{1+k\phi(n)} = m \bmod n$.

Bemerkung. Die Abbildung $\mathrm{Sig} : x \to x^d \bmod n$ ist multiplikativ. Aus den Signaturen für m_1 und m_2 kann man durch Multiplikation die Signatur für $m_1 m_2$ berechnen. Außerdem stimmen die Signaturen von zwei Nachrichten überein, falls diese sich um ein Vielfaches von n unterscheiden. Aus Sicherheitsgründen sollte deshalb die Nachricht vor dem Unterschreiben mit einer *kollisionsresistenten Hashfunktion h* „gehasht“ werden, d.h. es sollte $h(m)$ anstelle von m unterschrieben werden. (Eine Hashfunktion ist eine effizient berechenbare

Funktion, die Bitstrings beliebiger Länge in Bitstrings einer konstanten Länge abbildet. Eine Hashfunktion heißt kollisionsresistent, falls es nicht möglich ist, zwei Bitstrings zu finden, die auf denselben String abgebildet werden.) Diese Vorgehensweise wird auch vom PKCS-Standard in [PKC93] vorgeschlagen.

Die Sicherheit des RSA–Verfahrens beruht auf der Schwierigkeit des *Faktorisierungsproblems*, dem Problem, große Zahlen n in ihre Primfaktoren zu zerlegen. Dieses Problem der algorithmischen Zahlentheorie gilt als schwer berechenbar. Für $n \geq 2^{750}$ gilt das Faktorisierungsproblem mit heutigen Methoden als nicht lösbar.

Der Aufwand für die Erzeugung einer RSA-Signatur hängt vom geheimen Schlüssel d ab. Wiener zeigte in [Wie90], daß das RSA-Verfahren gebrochen werden kann, wenn d weniger als $(\log n)/4$ Bits besitzt. In vielen Anwendungen wählt man dagegen einen kleinen öffentlichen Schlüssel, um die Signaturen effizient verifizieren zu können. In diesem Fall besitzt d fast immer ungefähr $\log n$ Bits. Damit ist die Berechnung einer RSA-Signatur sehr aufwendig.

2.1.2 Das Schnorr-Signaturverfahren

Wir stellen hier das Schnorr-Signaturverfahren [Sch89] in der allgemeinen Form für beliebige zyklische Gruppen vor.

Sei G eine zyklische Gruppe mit primer Ordnung q und $g \in G$ ein Generator. Außerdem sei $h : \{0,1\}^* \to \{0,1\}^t$ eine kollisionsresistente Hashfunktion ($t < \log q$ ist ein Sicherheitsparameter des Schemas).

Jeder Anwender A wählt einen privaten Schlüssel $s \in \{1,\ldots,q\}$ und berechnet daraus den öffentlichen Schlüssel $v = g^{-s}$. Um eine Nachricht m zu unterschreiben, wählt er ein zufälliges $r \in \{1,\ldots,q\}$ und berechnet $x := g^r$. Außerdem setzt er $y := r + se \bmod q$ mit $e := h(x,m)$. Das Paar (e,y) ist nun *die Signatur von* m. Jeder Anwender kann die Gültigkeit der Signatur verifizieren, indem er $\hat{x} := g^y v^e$ berechnet und testet, ob $h(\hat{x},m) = e$ gilt.

Die Sicherheit des Schnorr-Signaturverfahrens beruht auf der Schwierigkeit, in der Gruppe G den *diskreten Logarithmus*, d.h. zu gegebenem $z \in G$ eine ganze Zahl a mit $g^a = z$, zu berechnen. Darüber hinaus hängt die Sicherheit des Verfahrens auch vom Sicherheitsparameter t und der gewählten Hashfunktion ab. In vielen Gruppen gilt die Berechnung des diskreten Logarithmus als schwer. Im ursprünglichen Verfahren wird eine zyklische Untergruppe G von $\mathbb{Z}_p^*$ der Ordnung q verwendet. Für $p \geq 2^{750}$ und $q \geq 2^{160}$ gilt der diskrete Logarithmus in dieser Gruppe mit heutigen Methoden als nicht berechenbar.

Den größten Teil des Berechnungsaufwandes stellt die Erzeugung der zufälligen Präsignatur (r,x) dar. Wir werden in dieser Arbeit Verfahren zur effizienten Berechnung von pseudozufälligen Präsignaturen entwickeln.

2.2 Generische Algorithmen

Der Begriff *generischer Algorithmus* wurde von Shoup ([Sho97]) eingeführt. Ein generischer Algorithmus erhält seine Eingaben aus einer Gruppe G und führt auf den Gruppenelementen als einzige Operationen die Multiplikation/Division in der Gruppe und den Test auf Gleichheit aus. Generische Algorithmen können deshalb als Blackbox-Algorithmen mit Multiplikation und Division als Blackbox-Operationen betrachtet werden.

Shoup zeigte im Modell der generischen Algorithmen untere Schranken für das Berechnen des diskreten Logarithmus, das Lösen des Diffie-Hellmann-Problems ([DH76]) und das Brechen des Schnorr Authentifikationsverfahrens ([Sch89],[Sch91]). Zuvor hatte bereits Nechaev in [Nec94] für spezielle generische Algorithmen untere Schranken für die Berechnung des diskreten Logarithmus bewiesen. Maurer und Wolf [MW98] zeigen untere Schranken für generische Reduktionen vom Diffie-Hellmann-Problem zum Problem des diskreten Logarithmus. Schnorr untersucht in [Sch98] die Unvorhersagbarkeit von diskreter Log-Bits für generische Algorithmen. Andere Arten von Blackbox-Algorithmen wurden von Boneh und Lipton in [BL96] und Babai und Szemerédi in [BS84] betrachtet. In dieser Arbeit basieren die Sicherheitsanalysen ebenfalls auf dem Modell der generischen Algorithmen.

Ein einfaches Beispiel für generische Algorithmen ist der „Baby Step, Giant Step"-Algorithmus von Shanks ([Sti95, 165f.]) zur Berechnung des diskreten Logarithmus in einer zyklischen Gruppe G. Dabei nehmen wir an, die Ordnung von G sei prim und setzen $k := \lceil\sqrt{|G|}\rceil$. Der Algorithmus erhält als Eingabe einen Generator g und ein beliebiges Element $x \in G$ und gibt ein s mit $x = g^s$ aus. Dazu berechnet er für $i = 0, \ldots, k-1$ die Werte $y_i := g^i$ (Baby Step) und für $j = 0, \ldots, k-1$ die Werte $z_i = x/g^{jk}$ (Giant Step). Danach sucht er eine Kollision $y_{i_0} = z_{j_0}$ und gibt $s_0 := j_0k + i_0$ aus[1]. Nach Konstruktion der y_i und z_j gilt dann $g^{s_0} = x$. Da $k^2 \geq |G|$ gilt, läßt sich jedes $s \in [0, |G|)$ als $s = jk + i$ mit $i, j \in [0, k-1]$ ausdrücken. Daher existiert immer eine Kollision und der Algorithmus ist immer erfolgreich.

Wir geben nun eine formale Definition für generische Algorithmen. Diese dürfen außer von der zugrunde liegenden Gruppe G noch von zusätzlichen Parametern abhängen. Diese Parameter definieren zusammen mit der Gruppe und einer Relation zwischen den Eingaben und den Ausgaben das Berechnungsproblem, auf das sich die Erfolgswahrscheinlichkeit bezieht. Diese wird über die Verteilung der Eingaben genommen.

Definition 2.1 (generischer Algorithmus) *Sei eine endliche Gruppe G, $n \in \mathbb{N}$ und eine*

[1] Im Modell der Turing-Maschine kann er eine Kollision in $O(k \log k)$ Schritten finden, indem er zuerst die beiden Listen der y_i bzw. z_j sortiert. Im generischen Modell vernachlässigen wir den Aufwand für das Auffinden von Kollisionen (siehe Definition 2.1)

Menge von öffentlichen Parametern gegeben. Ein generischer Algorithmus *auf G ist ein deterministischer Algorithmus $\mathcal{A}$ mit Eingaben $x_1, \ldots, x_n$ aus G. Die Berechnung von $\mathcal{A}$ ist auf die folgende Weise in Schritte aufgeteilt.*

Im Schritt ν berechnet $\mathcal{A}$ ein Element $x_\nu := x_1^{a_1} \cdots x_n^{a_n} \in G$.[2] *Dabei hängen die Exponenten $a_1, \ldots, a_n$ allein von ν, der Kollisionsmenge $\mathcal{CO}_{\nu-1} := \{(i,j) \mid x_i = x_j,\ 1 \leq i < j < \nu\}$ und den öffentlichen Parametern ab. Analog dazu hängt die Ausgabe $y \in \{0,1\}^*$, die der Algorithmus nach t Schritten ausgibt, allein von der Kollisionsmenge $\mathcal{CO}_t$ und den öffentlichen Parametern ab.*

Die Länge *eines generischen Algorithmus ist die Anzahl seiner Schritte. Für eine gegebene Verteilung $\mathcal{D}$ der Eingaben und eine Relation R auf $\{0,1\}^* \times G^n$ sei die* Erfolgswahrscheinlichkeit *von $\mathcal{A}$ zur Berechnung von R definiert als* $\mathrm{Ws}_{\mathcal{D}}[(y, x_1, \ldots, x_n) \in R]$.

Es bedeutet keine Einschränkung der Allgemeinheit, nur deterministische Algorithmen zu betrachten: Wir können bei einem probabilistischen Algorithmus die (in Bezug auf die Erfolgswahrscheinlichtkeit) optimale Folge von internen Zufallsbits fixieren und erhalten einen deterministischen Algorithmus mit mindestens genauso großer Erfolgswahrscheinlichkeit. Eine obere Schranke für die Erfolgswahrscheinlichkeit von deterministischen generischen Algorithmen impliziert deshalb immer auch eine obere Schranke für probabilistische.

Bemerkung. Shoup verwendete in [Sho97] eine andere Definition generischer Algorithmen. In seinem Modell erhält der Algorithmus seine Eingaben $x_1, \ldots, x_n$ aus G kodiert mit einer zufälligen injektiven Funktion $\sigma : G \to S$. Hierbei ist $S \subset \{0,1\}^*$ fest mit $|S| \geq |G|$. Während seiner Berechnung hat der Algorithmus Zugriff auf zwei Orakel $\mathcal{O}^+ : (\sigma(x), \sigma(y)) \to \sigma(x \cdot y)$ und $\mathcal{O}^- : (\sigma(x), \sigma(y)) \to \sigma(x/y)$. Auf den Eingaben, den Orakelantworten sowie den öffentlichen Parametern kann der Algorithmus beliebige Berechnungen ausführen. Die *Länge* des Algorithmus ist die Anzahl seiner Orakelanfragen. Für die Details verweisen wir auf [Sho97].

Im Shoup-Modell kann das Verhalten des Algorithmus nicht nur von den Gleichheitsrelationen sondern auch von der Kodierung der Elemente abhängen. Da diese jedoch zufällig gewählt ist, bedeutet dies für den Algorithmus keinen Vorteil. Alle unsere Beweise lassen sich daher auch auf das Shoup-Modell übertragen.

In [Nec94] zeigte Nechaev das folgende Resultat.

Satz 2.1 (Nechaev 1994) *Sei G eine von $g \in G$ erzeugte endliche zyklische Gruppe und q ein Primteiler von $|G|$. Dann hat jeder generischer Algorithmus $\mathcal{A}$ der Länge t, der für*

[2] In den ersten n Schritten liest $\mathcal{A}$ die Eingaben ein

ein zufälliges $y \in G$ aus g und y den diskreten Logarithmus von y zur Basis g berechnet, höchstens die Erfolgswahrscheinlichkeit $t^2/(2q)$.

Der Einfachkeit halber geben wir den Beweis nur für den Fall $|G| = q$ (d.h. die Gruppenordnung ist prim) an. Für den allgemeinen Fall verweisen wir auf [Nec94] oder [Sho97].

Beweis. Seien $x_1 := g$, $x_2 := y$ die Eingaben des Algorithmus und für $\nu = 2, \ldots, t$ sei $x_\nu \in G$ das von $\mathcal{A}$ im Schritt ν berechnete Element. Dann gibt es $a_1, \ldots, a_t, b_1, \ldots, b_t$ mit $x_\nu = g^{a_\nu} y^{b_\nu}$ für $\nu = 1, \ldots, t$. Wir können ohne Einschränkung der Allgemeinheit $(a_\nu, b_\nu) \neq (a_\mu, b_\mu) \bmod q$ für $\nu \neq \mu$ annehmen. Da $\mathcal{A}$ generisch ist, hängen die Paare (a_ν, b_ν) nur von der Schrittzahl ν und der Kollisionsmenge $\mathcal{CO}_{\nu-1}$ ab.

Wir schätzen zunächst die Wahrscheinlichkeit dafür ab, daß eine Kollision der berechneten Gruppenelemente auftritt. Angenommen, in den Schritten $1, \ldots, j-1$ ist keine Kollision aufgetreten. Falls nun $x_i = x_j$ für ein $i < j$ gilt, erhalten wir

$$a_i + sb_i = a_j + sb_j \bmod q \tag{2.1}$$

mit $s := \log_g(y)$. Auf der anderen Seite gilt aber $\mathcal{CO}_0 = \cdots = \mathcal{CO}_{j-1} = \emptyset$. Da a_i, b_i, a_j, b_j nur von i, j und $\mathcal{CO}_i, \mathcal{CO}_j$ abhängen und s zufällig gewählt ist, gilt (2.1) höchstens mit Wahrscheinlichkeit $1/q$. Durch Summation über $i = 1, \ldots, j-1$ folgt also, daß höchstens mit Wahrscheinlichkeit $(j-1)/q$ in Schritt j eine Kollision auftritt. Indem wir über alle j summieren, erhalten wir, daß $\mathcal{CO}_t \neq \emptyset$ höchstens mit Wahrscheinlichkeit $\binom{t}{2}/q$ gilt.

Es bleibt noch der Fall zu betrachten, daß keine Kollision auftritt. Da die Ausgabe s' von $\mathcal{A}$ nur von t und $\mathcal{CO}_t$ abhängt, ist sie in diesem Fall konstant. Daher ist die Wahrscheinlichkeit, daß $\mathcal{CO}_t = \emptyset$ gilt und die Ausgabe s' korrekt ist, höchstens $1/|G| \leq 1/q$.

□

Falls $|G|$ prim ist, ist die Schranke exakt bis auf den Faktor 2, wie der „Baby Step, Giant Step"-Algorithmus von Shanks (siehe [Sti95, 165f.]) und Pollards Rho–Methode (siehe [MvOV97, 106ff.]) zeigen. Fall darüber hinaus – wie im Schnorr–Signaturverfahren – $G \subset \mathbb{Z}_p^*$ gilt, so sind diese generischen Algorithmen in vielen Fällen die schnellsten bekannten Algorithmen. Es gibt zwar subexponentielle Algorithmen zur Berechnung des diskreten Logarithmus in $\mathbb{Z}_p^*$ (z.B. die Index–Calculus–Methode, siehe [MvOV97, 109ff.]), doch deren Laufzeit hängt stets von p. Falls nun p im Vergleich zu q so groß ist, daß die Laufzeit dieser Algorithmen größer als $\sqrt{q}$ ist, sind die schnellsten Algorithmen zur Berechnung des diskreten Logarithmus generisch.

2.2.1 Erweiterungen des Modells

Generische Algorithmen führen auf den Gruppenelementen nur die Gruppenoperationen Multiplikation und Division aus. Unter Umständen ist es jedoch auch interessant, generische Algorithmen zu betrachten, die zusätzliche Operationen ausführen können. Dabei muß diese Operation in der Praxis nicht notwendigerweise effizient berechenbar sein. So gibt ein Resultat über die Schwierigkeit eines Problems für generische Algorithmen mit zusätzlicher Operation Aufschluß darüber, wie weit sich das Lösen des Problem auf das Berechnen dieser Operation zurückführen läßt.

Boneh und Lipton betrachten z.B. in [BL96] Algorithmen, denen neben Multiplikation, Division und Test auf Gleichheit auch die *Diffie-Hellmann Funktion* $f\colon (g^a, g^b) \to g^{ab}$ zur Verfügung steht. Diese Operation gilt allgemein als nicht effizient berechenbar, ist jedoch aus kryptographischer Sicht interessant, weil die Sicherheit des Diffie-Hellmann Schlüsselaustauschverfahrens ([DH76]) auf der Schwierigkeit dieser Funktion beruht. Boneh und Lipton zeigen, daß unter einer bestimmten zahlentheoretischen Annahme ein solcher Algorithmus den diskreten Logarithmus in einer endlichen zyklischen Gruppe in subexponentieller Zeit lösen kann.

In manchen Anwendungen stehen jedoch tatsächlich zusätzliche Operationen zur Verfügung. So wird sowohl bei RSA als auch beim Verfahren von Chaum und van Antwerpen ([CvA90]) die Signatur zu einer Nachricht m durch $f_s(m) = m^s$ berechnet, wobei s der geheime Schlüssel des Unterzeichners ist. Falls nun ein Angreifer Zugriff auf eine Chipkarte mit einer Implementierung eines dieser Verfahren hat, so kann er die Funktion f_s für beliebige m berechnen. Ein solcher Angriff entspricht einer Adaptively Chosen Message Attack gegen RSA bzw. gegen das Chaum-van Antwerpen-Signaturverfahren. Dieses Szenario läßt sich nun in der folgenden Weise in das generische Modell übertragen.

Sei G eine von $g \in G$ erzeugte endliche zyklische Gruppe und q ein Primteiler der Ordnung von G. Wir nehmen an, ein generischer Algorithmus berechnet für zufälliges $y = g^s \in G$ aus der Eingabe (g, y) den diskreten Logarithmus s. Zusätzlich zu den generischen Operationen kann er – allein abhängig von der Schrittzahl ν und $\mathcal{CO}_\nu$ – die Funktion $f_s : z \to z^s$ berechnen. Jede Anwendung dieser Funktion gilt dabei als ein Schritt.

Der folgende Satz liefert auch in diesem erweiterten Modell eine untere Schranke für die Erfolgswahrscheinlichkeit von Algorithmen zur Berechnung des diskreten Logarithmus. Dabei bezeichnen wir mit $\mathcal{O}^s$ das Orakel, welches zu $z \in G$ die Antwort z^s ausgibt.

Satz 2.2 *Sei G eine von $g \in G$ erzeugte endliche zyklische Gruppe und q ein Primteiler der Ordnung von G. Ferner sei $\mathcal{A}$ ein generischer Algorithmus der Länge t, der für ein zufälliges $y = g^s \in G$ aus g und y den diskreten Logarithmus s berechnet und dabei Zugriff* *auf $\mathcal{O}^s$ hat. Dann hat $\mathcal{A}$ höchstens Erfolgswahrscheinlichkeit $t^3/(3q)$.*

Beweis. Durch vollständige Induktion sieht man, daß die von $\mathcal{A}$ berechneten Gruppenelemente x_ν von der Form $g^{a_{\nu,\nu-1}s^{\nu-1}+\cdots+a_{\nu,1}s+a_{\nu,0}}$ für $a_{\nu,0},\ldots,a_{\nu,\nu-1} \in \mathbb{Z}$ sind. Ohne Einschränkung der Allgemeinheit können wir annehmen, daß $(a_{\nu,0},\ldots,a_{\nu,\nu-1}) \neq (a_{\mu,0},\ldots,a_{\mu,\mu-1})$ für $\nu \neq \mu$ gilt. Da $\mathcal{A}$ generisch ist, hängen $a_{\nu,0},\ldots,a_{\nu,\nu-1}$ nur von $\mathcal{CO}_{\nu-1}$ und ν ab.

Wir schätzen zunächst die Wahrscheinlichkeit dafür ab, daß eine Kollision der berechneten Gruppenelemente auftritt. Angenommen, in den Schritten $1,\ldots,j-1$ ist keine Kollision aufgetreten. Falls nun $x_i = x_j$ für ein $i < j$ gilt, erhalten wir

$$a_{j,j-1}s^{j-1} + \cdots + a_{j,1}s + a_{j,0} = a_{i,i-1}s^{i-1} + \cdots + a_{i,1}s + a_{i,0} \bmod q \ . \tag{2.2}$$

Auf der anderen Seite gilt aber $\mathcal{CO}_0 = \cdots = \mathcal{CO}_{j-1} = \emptyset$. Da $a_{i,0},\ldots,a_{i,i-1},a_{j,0},\ldots,a_{j,j-1}$ nur von i,j und $\mathcal{CO}_i,\mathcal{CO}_j$ abhängen und s zufällig gewählt ist, gilt (2.2) höchstens mit Wahrscheinlichkeit $(j-1)/q$.[3] Durch Summation über alle $1 \leq i < j$ folgt also, daß höchstens mit Wahrscheinlichkeit $(j-1)^2/q$ in Schritt j eine Kollision auftritt. Indem wir über alle j summieren, erhalten wir, daß $\mathcal{CO}_t \neq \emptyset$ höchstens mit Wahrscheinlichkeit

$$q^{-1}\sum_{j=1}^{t}(j-1)^2 \ = \ \frac{t(t-1)(2t-1)}{6q}$$

gilt. Wegen $t \geq 2$ ist das höchstens $\frac{t^3-3}{3q}$.

Falls auf der anderen Seite $\mathcal{CO}_t = \emptyset$ gilt, ist die Ausgabe s' von $\mathcal{A}$ konstant und damit höchstens mit Wahrscheinlichkeit q^{-1} korrekt. Insgesamt ist die Erfolgswahrscheinlichkeit also höchstens $t^3/(3q)$.

□

Wir wissen nicht, wie gut diese Schranke ist. Es ist nicht einmal klar, ob das beschriebene Exponentiationsorakel bei der Berechnung des diskreten Logarithmus überhaupt helfen kann – es ist kein generischer Algorithmus bekannt, der mit diesem Orakel den diskreten Logarithmus effizienter berechnet als der „Baby Step, Giant Step"-Algorithmus. Ein solcher Algorithmus müßte die Gruppenelemente $x_3,\ldots,x_t$ derart berechnen, daß $x_\mu = x_\nu$ für ein Paar $\mu < \nu$ mit Wahrscheinlichkeit größer $\binom{t}{2}/q$ gilt[4]. Dem Beweis von Satz 2.2 entnimmt man, daß es für alle $\nu \leq t$ ein Polynom P_ν vom Grad (höchstens) $\nu - 1$ mit $x_\nu = g^{P_\nu(s)}$ gibt. Der Algorithmus müßte also eine Menge von t paarweise verschiedenen Polynomen P_ν

[3]Da q prim ist besitzt jedes Polynom vom Grad $j-1$ über $\mathbb{Z}_q$ höchstens $j-1$ Nullstellen.

[4]Dabei sollte die Gleichheit nicht unabhängig von s gelten, da der Algorithmus daraus sonst keine Information über s erhält.

wählen, so daß die Differenz-Polynome $P_\nu - P_\mu$ (mit $\mu \neq \nu$) im Durchschnitt mehr als eine (von den Nullstellen der anderen Differenz-Polynome verschiedene) Nullstelle besitzen. Es ist nicht klar, ob eine Menge von Polynomen mit dieser Eigenschaft überhaupt existiert.

2.3 Gitter

Die Gittertheorie und die Gitterbasenreduktion sind mächtige Werkzeuge in der Kryptanalyse. Auch in dieser Arbeit spielen Gitter eine zentrale Rolle. Wir stellen daher in diesem Abschnitt die benötigten Begriffe und Fakten bereit. Für eine detailliertere Einführung verweisen wir auf die Standardwerke von Cassels [Cas71] und von Gruber und Lekkerkerker [GL69] oder auf [Sch95].

Definition 2.2 *Ein* Gitter *ist eine diskrete additive Untergruppe des $\mathbb{R}^n$. Ein linear unabhängiges Erzeugendensystem $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ eines Gitters L mit $\boldsymbol{b}_i \in \mathbb{R}^n$ für $i = 1, \ldots, m$ heißt* Basis *von L. Die Anzahl der Basisvektoren heißt der* Rang *des Gitters.*

Jedes Gitter besitzt eine Basis und der Rang des Gitters ist von der Basis unabhängig. Für ein Gitter L mit Basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ gilt

$$L = \left\{ \sum_{i=1}^{m} x_i \boldsymbol{b}_i \mid x_1, \ldots, x_m \in \mathbb{Z} \right\} .$$

Definition 2.3 *Die* Determinante $\det(L)$ *eines Gitters L mit Basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ ist das m-dimensionale Volumen seiner Grundmasche $\{\boldsymbol{x} = \sum_{i=1}^{m} x_i \boldsymbol{b}_i \mid 0 \leq x_i < 1\}$. Die Determinante eines Gitters ist unabhängig von der Wahl der Basis.*

Das i-te sukzessive Minimum $\lambda_{i,\|\cdot\|}(L)$ *bezüglich einer festen Norm $\|\cdot\|$ ist definiert als die kleinste positive Zahl r, so daß es i linear unabhängige $\boldsymbol{x}_j \in L$ mit $\|\boldsymbol{x}_j\| \leq r$ gibt.*

Der Satz von Minkowski stellt eine Verbindung zwischen der Determinante eines Gitters und den sukzessiven Minima her. Für den Beweis verweisen wir auf [GL69, 123f.].

Theorem 2.3 (Minkowski 1896) *Sei $L \subset \mathbb{Z}^n$ ein Gitter vom Rang n und $V_{n,p}$ das n-dimensionale Volumen von $\{\boldsymbol{v} \in \mathbb{R}^n \mid \|\boldsymbol{v}\|_p \leq 1\}$. Dann gilt*

$$V_{n,p} \prod_{i=1}^{n} \lambda_{i,\|\cdot\|_p} \leq 2^n \det(L) \ .$$

Insbesondere gilt

$$\prod_{i=1}^{n} \lambda_{i,\|\cdot\|_\infty} \leq \det(L) \ .$$

2.3.1 Gitterbasenreduktion

In vielen Anwendungen ist es notwendig, einen (bezüglich einer festen Norm) möglichst kurzen Vektor aus einem gegebenen Gitter L zu berechnen. Dieses Berechnungsproblem gilt im allgemeinen als schwer. In der Maximumsnorm ist das Problem NP-hart (siehe [vE81]), in der euklidischen Norm ist es zumindest NP-hart bezüglich probabilistischer Transformationen (siehe [Ajt97]).

Wir stellen nun einige Verfahren vor, die aus einer gegebenen Gitterbasis eine Basis mit möglichst kurzen Vektoren berechnen. Eine solche Vorgehensweise bezeichnet man als *Gitterbasenreduktion.* Für eine detailliertere Einführung verweisen wir auf [Sch95].

Der *LLL–Reduktions–Algorithmus* wurde von Lenstra, Lenstra und Lovász in [LLL82] veröffentlicht. Er transformiert eine beliebige ganzzahlige Basis eines Gitters L in eine Basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ mit

$$\|\boldsymbol{b}_j\|_2^2 \leq (\delta - 1/4)^{-(m-1)} \; \lambda_{j,\|\cdot\|_2}(L)^2 \;\; ,$$

wobei $1/4 \leq \delta < 1$ ein Parameter des Algorithmus ist. Der LLL–Algorithmus benötigt $O(m^3 n \log M)$ arithmetische Schritte, falls die Länge aller Vektoren der Eingabebasis durch M beschränkt ist.

Der LLL–Algorithmus arbeitet mit ganzzahliger Arithmetik. Bei großen Zahlen ist diese wesentlich aufwendiger als die Gleitpunktarithmetik. Schnorr und Euchner entwickelten in [SE94] eine Variante des LLL–Algorithmus, welche große Teile der Berechnung mit Gleitpunktarithmetik durchführt. Die oben genannten Schranken für die Länge der Ausgabevektoren gelten für dieses Verfahren nicht mehr. In der Praxis zeigt sich jedoch nur ein geringer Unterschied der Länge der Ausgabebasis zwischen der originalen LLL–Reduktion und der Schnorr–Euchner–Variante.

In [Sch87] verallgemeinerte Schnorr den Ansatz von Lenstra, Lenstra und Lovász. Sein *β-Blockreduktions–Algorithmus* berechnet aus einer gegebenen Gitterbasis eine Basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m)$ mit

$$\|\boldsymbol{b}_j\|_2^2 \leq \frac{j+3}{4\delta} \left(\frac{\gamma_\beta}{\delta}\right)^{2\frac{m-1}{\beta-1}} \lambda_{j,\|\cdot\|_2}(L)^2 \;\; ,$$

wobei $\beta \in \{2, \ldots, m\}$ (*die Blockgröße*) und $0 \leq \delta \leq 1$ Parameter des Algorithmus und γ_β die β-te Hermite–Konstante sind (siehe [Sch94]). Die LLL–Reduktion ist in dieser Hierarchie von β-Blockreduktionen als Spezialfall $\beta = 2$ enthalten. Für $\beta > 3$ sind keine polynomialen Laufzeitschranken bekannt. Der in [SE94] angegebene Algorithmus läuft allerdings für $\beta \leq 20$ und $n \leq 66$ in praktikabler Zeit. In [SH95] gaben Schnorr und Hörner einen Algorithmus zur *geschnittenen Blockreduktion* von Gitterbasen an. Diese Variante des Blockreduktions–Algorithmus bricht die Suche nach dem kürzesten Vektor in einem Teilgitter ab, wenn die Wahrscheinlichkeit für das Auffinden eines noch kürzeren Vektors hinreichend klein ist. Für die Abschätzung der Wahrscheinlichkeit verwendet das Verfahren die Gauß'sche Volumenheuristik. Versuche haben gezeigt, daß die geschnittene Blockreduktion wesentlich effizienter kurze Vektoren auffindet als die ungeschnittene (siehe [Hör94]).

2.3.2 Rucksackprobleme

Ein *Rucksackproblem* ist gegeben durch natürliche Zahlen $a_1, \ldots, a_n$ und s. Gesucht sind Zahlen $e_1, \ldots, e_n \in \{0,1\}$ so daß $\sum_{i=1}^n e_i a_i = s$ gilt. Ein solches Tupel $(e_1, \ldots, e_n)$ heißt *Lösung des Rucksackproblems*. Die Sicherheit verschiedener Verschlüsselungsverfahren (z.B. [MH78], [CR88]) beruht auf der Schwierigkeit Lösungen zu Rucksackproblemen zu finden. Coster et al. zeigten in [CJL$^+$92], daß sich aus einem zufälligen Rucksackproblemen mit $a_1, \ldots, a_n < N$ für ein $N \in \mathbb{N}$ ein Gitter konstruieren läßt, dessen kürzester[5] Vektor fast immer[6] eine Lösung des Rucksackproblems liefert, sofern

$$n < 0.9408 \ldots \cdot \log N$$

gilt.

In dieser Arbeit betrachten wir solche Varianten von Rucksackproblemen, bei denen $a_1, \ldots, a_n$ und s aus $\mathbb{Z}_p$ für eine Primzahl p gewählt sind und eine Lösung $(x_1, \ldots, x_n) \in [0, 2^\ell - 1]^n$ mit $\langle \boldsymbol{x}, \boldsymbol{a} \rangle = s \bmod p$ gesucht ist[7]. Ein solches Problem nennen wir *modulares ℓ-Bit Rucksackproblem*. Einen solchen Vektor nennen wir *Lösung des modularen ℓ-Bit Rucksackproblems*.

Das Resultat aus [CJL$^+$92] läßt sich auf den Fall modularer ℓ-Bit Rucksackprobleme übertragen. Wir zeigen, daß sich aus einem zufälligen modularen Rucksackproblem ein Gitter konstruieren läßt, dessen kürzester[5]Vektor fast immer[6]eine Lösung des Rucksackproblems liefert, sofern

$$n < (\ell + 1.0471)^{-1} \log p$$

gilt.

Dieses Resultat ist nicht mit dem aus [CJL$^+$92] vergleichbar. Zum einen betrachten wir nicht nur Lösungen des Rucksackproblems aus $\{0,1\}^n$, sondern solche mit beliebigen Koeffizienten aus $\{0, \ldots, 2^\ell - 1\}$. Zum anderen betrachten wir Gleichungen modulo einer Primzahl. Einige der in [CJL$^+$92] verwendeten Argumentationen lassen sich auf diesen Fall nicht übertragen.

---

[5]In der euklidischen Norm

[6]D.h. mit Wahrscheinlichkeit $1 - 2^{cn}$ für eine Konstante $c < 0$

[7]Mit $\langle \cdot, \cdot \rangle$ bezeichnen wir das Standardskalarprodukt

Satz 2.4 *Sei $K > 2^{\ell-1}\sqrt{n+1}$ und $p > 2^{2\ell}\sqrt{n+1}$. Für festes $\boldsymbol{y} \in [0, 2^\ell)^n \setminus \{\boldsymbol{0}\}$ und festes $s \in [0, p)$ sei $\boldsymbol{a}$ zufällig so aus $[0, p)^n$ gewählt, daß*

$$\sum_{i=1}^{n} y_i a_i = s \bmod p \ . \tag{2.3}$$

Ferner sei $\boldsymbol{x}^0$ der (bezüglich der euklidischen Norm) kürzeste Vektor des von

$$\begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{n+1} \\ \boldsymbol{b}_{n+2} \end{pmatrix} = \begin{pmatrix} 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & Ks \\ 0 & 1 & & & Ka_1 \\ \vdots & & \ddots & & \vdots \\ 0 & & & 1 & Ka_n \\ 0 & 0 & \cdots & 0 & Kp \end{pmatrix}$$

erzeugten Gitters L. Dann sind die Vektoren $\boldsymbol{x}^0 - \langle \boldsymbol{x}^0, \boldsymbol{b}_1 \rangle \boldsymbol{b}_1$ und $\boldsymbol{y}' := (0, y_1, \ldots, y_n, Ks)$ mindestens mit Wahrscheinlichkeit

$$1 - 4\sqrt{n+1}\, 2^{n(\ell+1.0471)}/p$$

über $\mathbb{Z}$ linear abhängig.

Wir können dieses Resultat noch verbessern, indem wir in der Gitterbasis $\boldsymbol{b}_1 = ((2^\ell - 1)/2, \cdots, (2^\ell - 1)/2, Ks)$ wählen. Wir erhalten dann, daß der kürzeste Gittervektor fast immer eine Lösung des Rucksackproblems liefert, sofern $n < (\log_2(2^\ell - 1) + 1.0471)^{-1} p$ gilt. Im Fall $\ell = 1$ erhalten wir also die Konstante 0.6464 gegenüber dem Wert 0.4885 aus Satz 2.4. Im Fall $\ell = 2$ erhalten wir eine verbesserte Konstante 0.38 gegenüber 0.328. Für $\ell > 2$ ist der Unterschied zwischen den beiden Resultaten noch geringer. Wir verzichten daher auf diese Optimierung.

Beweis. Wegen (2.3) gilt $\boldsymbol{y}' \in L$. Wegen $\boldsymbol{y} \neq 0$ folgt daraus

$$\|\boldsymbol{x}^0\|_2 \ = \ \lambda_{1,\|\cdot\|_2} \ \leq \ \|\boldsymbol{y}' - \boldsymbol{b}_1\|_2 \ < \ 2^{\ell-1}\sqrt{n+1} \ .$$

Wegen $\|\boldsymbol{b}_1\|_2 \geq 2^{\ell-1}\sqrt{n+1}$ ist also $\boldsymbol{x}_0$ kein Vielfaches von $\boldsymbol{b}_1$, d.h. $\boldsymbol{x}^0 - \langle \boldsymbol{x}^0, \boldsymbol{b}_1 \rangle \boldsymbol{b}_1$ ist nicht trivial. Somit reicht es aus, zu zeigen, daß es höchstens mit Wahrscheinlichkeit $4\sqrt{n+1}\, 2^{n(\ell+1.0471)}/p$ einen von $\boldsymbol{y}'$ und $\boldsymbol{b}_1$ linear unabhängigen Vektor $\boldsymbol{x} \in L$ mit $\|\boldsymbol{x}\|_2 < 2^{\ell-1}\sqrt{n+1}$ gibt.

Sei $\boldsymbol{x} = \sum \beta_i \boldsymbol{b}_i$ mit $\beta_1, \ldots, \beta_{n+2} \in \mathbb{Z}$ ein solcher Vektor. Wir setzen $\boldsymbol{z} := \boldsymbol{x} - \beta_1 \boldsymbol{b}_1$. Wegen $\|\boldsymbol{x}\|_2 < 2^{\ell-1}\sqrt{n+1}$ gilt $|\beta_1| < \sqrt{n+1}$. Außerdem folgt aus $2^{\ell-1}\sqrt{n+1} < K$ offensichtlich $x_{n+1} = 0$ und damit

$$\sum_{i=1}^{n} \beta_{i+1} a_i = \sum_{i=1}^{n} z_{i+1} a_i = -\beta_1 s \bmod p \ . \tag{2.4}$$

Falls nun $\boldsymbol{z}$ von $\boldsymbol{y}$ linear unabhängig über $\mathbb{Z}$ ist, dann gilt das auch über $\mathbb{Z}_p$. Anderenfalls würde nämlich $z_i/y_i = z_j/y_j \bmod q$ aber $x_i/y_i \neq x_j/y_j$ für ein Paar $2 \leq i,j \leq n+1$ gelten, was wegen $p > 2^{2\ell}\sqrt{n+1}$ und $|z_i| = |x_i - \beta_1 2^{\ell-1}| \leq 2^{\ell}\sqrt{n+1}$ und $|y_i| < 2^{\ell}$ unmöglich ist. Da $\mathbb{Z}_p$ ein Körper ist, erhalten wir für linear unabhängige $\boldsymbol{z}$ und $\boldsymbol{y}$

$$\begin{aligned}
\mathrm{Ws}[\,(2.4)\,] &= \mathrm{Ws}_{\boldsymbol{a}}[\,(2.4) \mid (2.3)\,] \\
&= \frac{\mathrm{Ws}_{\boldsymbol{a}}[\,(2.4) \text{ und } (2.3)\,]}{\mathrm{Ws}_{\boldsymbol{a}}[\,(2.3)\,]} \\
&\leq \frac{1}{p} \ .
\end{aligned}$$

Hierbei bezeichnet $\mathrm{Ws}_{\boldsymbol{a}}$ die Wahrscheinlichkeit über eine gleichverteilte Wahl von $\boldsymbol{a}$ aus $[0,p)^n$. Wir erhalten damit, daß es für festes $-\sqrt{n+1} < \beta_1 < \sqrt{n+1}$ und $\boldsymbol{b} := 2^{\ell-1}(\beta_1,\ldots,\beta_1) \in \mathbb{Z}^n$ höchstens mit Wahrscheinlichkeit

$$\frac{\left|\left\{\boldsymbol{v} \in \mathbb{Z}^n : \ \|\boldsymbol{v}-\boldsymbol{b}\|_2 < 2^{\ell-1}\sqrt{n+1}\right\}\right|}{p} = \frac{\left|\left\{\boldsymbol{v} \in \mathbb{Z}^n : \ \|\boldsymbol{v}\|_2 < 2^{\ell-1}\sqrt{n+1}\right\}\right|}{p}$$

ein $\boldsymbol{x} = \sum \beta_i \boldsymbol{b}_i$ mit $\beta_2,\ldots,\beta_{n+2} \in \mathbb{Z}$ und $\|\boldsymbol{x}\|_2 < 2^{\ell-1}\sqrt{n+1}$ gibt, so daß $\boldsymbol{x} - \beta_1\boldsymbol{b}_1$ linear unabhängig von $\boldsymbol{y}'$ ist.

Auf der anderen Seite ist nach [MO90a] für alle $\alpha > 0$ die Anzahl der ganzzahligen Vektoren $\boldsymbol{x}$ mit $\|\boldsymbol{x}\|_2 \leq \sqrt{\alpha n}$ durch $e^{s_\alpha \alpha n} f(s_\alpha)^n$ beschränkt, wobei s_α durch $\frac{d}{ds}\ln f(s_\alpha) = -\alpha$ mit $f(s) := \sum_{k=-\infty}^{\infty} e^{-sk^2}$ gegeben ist.

Nach [MO90a] gilt allgemein

$$\lim_{\alpha\to\infty} s_\alpha\alpha + \ln f(s_\alpha) - \frac{1}{2}\ln(2\pi e\alpha) = 0$$

und

$$\frac{d}{d\alpha}\left(s_\alpha\alpha + \ln f(s_\alpha) - \frac{1}{2}\ln(2\pi e\alpha)\right) < 0 \ .$$

Damit erhalten wir für $\alpha \geq 1$

$$s_\alpha\alpha + \ln f(s_\alpha) \leq \frac{1}{2}\ln(2\pi e\alpha) + \epsilon$$

mit $\epsilon := s_1 + \ln f(s_1) - \frac{1}{2}\ln(2\pi e)$. Wir erhalten insgesamt, daß die Anzahl der ganzzahligen Vektoren $\boldsymbol{v} \in \mathbb{Z}^n$ mit $\|\boldsymbol{v}\|_2 < 2^{\ell-1}\sqrt{n+1}$ durch

$$2^{n(\ell-1)}\left(\frac{n+1}{n}2\pi e^{1+\epsilon}\right)^{n/2} \leq \sqrt{e}\, 2^{n(\ell-1/2+\log\pi+(1+\epsilon)\log e)}$$

beschränkt ist.

Eine numerische Berechnung ergibt $\epsilon \log e < 10^{-8}$. Damit ist die Wahrscheinlichkeit, daß es einen von $\boldsymbol{y}'$ und $\boldsymbol{b}_1$ linear unabhängigen Vektor $\boldsymbol{x}$ mit $\|\boldsymbol{x}\|_2 \leq 2^{\ell-1}\sqrt{n+1}$ gibt, höchstens $2\sqrt{e(n+1)}\,2^{n(\ell+1.0471)}/p$

□

Bemerkung. Das Resultat läßt sich leicht auf den Fall zusammengesetzter Moduln verallgemeinern. Falls p die Primfaktorzerlegung $p = \prod_{i=1}^m p_i^{e_i}$ mit $p_1, \ldots, p_k > 2^{2\ell}$ besitzt, folgt aus dem chinesischen Restsatz

$$\mathrm{Ws}_{\boldsymbol{a}}\big[\,(2.4) \text{ und } (2.3)\,\big] = \Big(\prod_{i=1}^{k} p_i^{e_i}\Big)^{-2}$$

und wir erhalten, daß $\boldsymbol{x}^0 - \langle \boldsymbol{x}^0, \boldsymbol{b}_1\rangle \boldsymbol{b}_1$ mindestens mit Wahrscheinlichkeit $1 - 4(n+1)^{1/2} 2^{n(\ell+1.0471)} / \prod_{i=1}^k p_i^{e_i}$ ein nichttriviales ganzzahliges Vielfaches von $(0, y_1, \ldots, y_n, Ks)$ ist.

Es ist klar, daß Satz 2.4 auch für eine zufällige Wahl von s aus $[0, p)$ gilt. Die dadurch definierte Verteilung von $(s, \boldsymbol{a})$ ist identisch mit der Verteilung, die man erhält, wenn man $\boldsymbol{a}$ zufällig aus $[0, p)^n$ wählt und dann $s = \sum y_i a_i \bmod p$ setzt. (Hierbei benötigen wir jedoch, daß p prim ist.) Wir erhalten somit das folgende Korollar.

Korollar 2.5 *Sei* $K > 2^{\ell-1}\sqrt{n+1}$ *und* $p > 2^{2\ell}\sqrt{n+1}$.. *Für festes* $\boldsymbol{y} \in [0, 2^\ell)^n \setminus \{\boldsymbol{0}\}$ *sei* $\boldsymbol{a}$ *zufällig aus* $[0, p)^n$ *gewählt und* $s := \sum_{i=1}^n y_i a_i \bmod p$. *Ferner sei* $\boldsymbol{x}^0$ *der (bezüglich der euklidischen Norm) kürzeste Vektor des von*

$$\begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_{n+1} \\ \boldsymbol{b}_{n+2} \end{pmatrix} = \begin{pmatrix} 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & Ks \\ 0 & 1 & & & Ka_1 \\ \vdots & & \ddots & & \vdots \\ 0 & & & 1 & Ka_n \\ 0 & 0 & \cdots & 0 & Kp \end{pmatrix}$$

erzeugten Gitters L. Dann sind die Vektoren $\boldsymbol{x}^0 - \langle \boldsymbol{x}^0, \boldsymbol{b}_1\rangle \boldsymbol{b}_1$ *und* $\boldsymbol{y}' := (0, y_1, \ldots, y_n, Ks)$ *mindestens mit Wahrscheinlichkeit*

$$1 - 4\sqrt{n+1}\,2^{n(\ell+1.0471)}/p$$

über $\mathbb{Z}$ *linear abhängig.*

Kapitel 3

Server-unterstützte RSA Protokolle

Die RSA-Signatur zu einer Nachricht m wird als $m^d \bmod n$ berechnet, wobei d der geheime Schlüssel und n der öffentliche Modul des Anwenders ist. In den typischen Anwendungen besitzen sowohl d als auch n mindestens 1024 Bit. Für Geräte mit geringer Rechenstärke (wie z.B. eine Chipkarte ohne Koprozessor) ist dieser Berechnungsaufwand zu groß. In [MKI89] entwarfen daher Matsumoto, Kato und Imai das Konzept der Server-unterstützten RSA Protokolle. In diesen hilft ein Server einem Client in einem interaktiven Protokoll bei der Erzeugung einer RSA-Unterschrift. Zugleich soll dabei jedoch der geheime Schlüssel des Client vor dem Server geheimgehalten werden.

Wir untersuchen in diesem Kapitel die Sicherheit der wichtigsten Server-unterstützten RSA Protokolle. Zum einen beweisen wir untere Schranken für Angriffe durch generische Algorithmen. Diese Schranken sind bis auf einen konstanten Faktor scharf. Zum anderen präsentieren wir einen neuen Angriff durch Gitterbasenreduktion.

Es gibt im wesentlichen zwei verschiedene Arten von Angriffen gegen Server-unterstützte RSA Protokolle: Bei *aktiven Angriffen* hält sich der Server nicht an das Protokoll, sondern liefert dem Client falsche Werte. Aus der vom Clienten dabei erzeugten (möglicherweise inkorrekten) Signatur versucht er dann Informationen über dessen geheimen Schlüssel zu erhalten. *Passive Angriffe* versuchen dagegen, den geheimen Schlüssel des Clienten nur aus den Daten von korrekt ausgeführten Protokollen zu berechnen.

Wir betrachten nur passive Angriffe. Außerdem betrachten wir keine Angriffe gegen das RSA-Verfahren. Die betrachteten Angriffe verwenden ausschließlich die in den Protokollen vorkommenden Größen.

3.1 RSA-S1

Die ersten Server-unterstützten RSA Protokolle RSA-S1 und RSA-S2 wurden in [MKI89] von Matsumoto, Kato und Imai veröffentlicht. Wir betrachten hier zunächst nur das Protokoll RSA-S1, das Protokoll RSA-S2 stellen wir in Abschnitt 3.3 vor.

3.1.1 Das Protokoll

Sei n das Produkt zweier großer verschiedener Primzahlen p und q. Die Sicherheitsparameter des Protokolles sind m, k und ℓ.

Definition 3.1 *Für ein $a \in \mathbb{N}$ ist das* Hamminggewicht $\mathrm{wt}(a)$ *definiert als die Anzahl der Einsen in der Binärdarstellung von a. Für $\boldsymbol{x} \in \mathbb{N}^n$ nennen wir* $\mathrm{wt}(\boldsymbol{x}) = \sum_{i=1}^{n} \mathrm{wt}(x_i)$ das Hamminggewicht von $\boldsymbol{x}$.

Die Zerlegung des geheimen Schlüssels. Vor der ersten Ausführung des Protokolls zerlegt der Client seinen geheimen Schlüssels d in der folgenden Weise. Zuerst wählt er einen zufälligen Vektor $\boldsymbol{f} = (f_1, \ldots, f_m) \in [0, 2^\ell)^m$ mit Hamminggewicht k, so daß $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i gilt[1]. Dann wählt er einen Vektor $\boldsymbol{d} = (d_1, \ldots, d_m) \in \mathbb{Z}_{\phi(n)}^m$ zufällig so, daß $d := \sum_{i=1}^{m} f_i d_i \in \mathbb{Z}_{\phi(n)}^*$ gilt.

Den Vektor $\boldsymbol{d}$ kann der Client nun öffentlich bekannt geben, die Vektoren $\boldsymbol{f}$ und $\boldsymbol{g}$ muß er geheim halten. Die Zerlegung $d = \sum_{i=1}^{m} f_i d_i \bmod \phi(n)$ muß nicht bei jeder Ausführungen des Protokolls neu gewählt werden.[2]

Das Protokoll. Um eine Signatur zu erzeugen, tritt der Client mit einem Server in Kontakt und führt mit diesem das folgende Protokoll aus.

1. Der Client sendet x, n und $\boldsymbol{d}$ zum Server.
2. Der Server sendet $z_i = x^{d_i} \bmod n$ für $i = 1, \ldots, m$ an den Client.
3. Der Client berechnet die Signatur y als $y = \prod_{i=1}^{m} z_i^{f_i} \bmod n$.

Bei diesem Verfahren benötigt der Client $k + \ell - 2$ Multiplikationen modulo n pro Signatur.[3] Sein Speicherbedarf ist $m + m \log n$ Bits. Da $\boldsymbol{d}$ bekannt ist, kann der Client diesen Vektor anstatt auf der Karte in einer Datei außerhalb der Karte speichern, die dem

[1] Diese Bedingung ist notwendig, um ein $\boldsymbol{d} \in \mathbb{Z}_{\phi(n)}^m$ mit $d = \sum_{i=1}^{m} f_i d_i \bmod \phi(n)$ zu finden

[2] Um sich gegen aktive Angriffe zu schützen, kann eine gelegentliche Erneuerung der Zerlegung jedoch sinnvoll sein (siehe Abschnitt 3.4)

[3] Die Berechnung in Schritt 3 wird durch das „square and multiply"-Verfahren durchgeführt

Server zugänglich ist. Der Server erhält dann in Schritt 1 vom Client nur x gesendet und liest dann die Werte d_i für $i = 1, \ldots, m$ aus dieser Datei ein. Dadurch verringert sich der Speicherbedarf des Client auf m Bits.

3.1.2 Die beste bekannte Attacke gegen RSA-S1

Der effizienteste passive Angriff gegen RSA-S1 wurde von Pfitzmann und Waidner ([PW93]) veröffentlicht. Die hier beschriebene Variante verwendet Ideen von van Oorschot und Wiener ([vOW96]) und ist noch etwas effizienter. Nach Definition gilt $x^d = \prod z_i^{f_i} \bmod n$ für ein $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k. Wir nehmen an, $m\ell$ und k seien gerade. Der Angreifer berechnet für alle $(f'_1, \ldots, f'_{\frac{m}{2}})$ mit Hamminggewicht $k/2$

$$\prod_{i \le m/2} x^{d_i f'_i} \bmod n \ ,$$

bildet damit eine Liste und sortiert diese. Außerdem berechnet er

$$y \Big(\prod_{i > m/2} x^{d_i f'_i} \Big)^{-1} \bmod n$$

für alle $(f'_{\frac{m}{2}+1}, \ldots, f'_m)$ mit Hamminggewicht $k/2$, bildet damit eine Liste und sortiert diese. Falls nun $(f_1, \ldots, f_{\frac{m}{2}})$ Hamminggewicht $k/2$ hat, gibt es mindestens eine Kollision

$$\prod_{i \le m/2} x^{d_i f'_i} = y \Big(\prod_{i > m/2} x^{d_i f'_i} \Big)^{-1} \bmod n \ .$$

Damit erhält der Angreifer $\boldsymbol{f}$.

Mit Wahrscheinlichkeit $\binom{m\ell/2}{k/2}^2 \binom{m\ell}{k}^{-1}$ besitzt $(f_1, \ldots, f_{\frac{m}{2}})$ Hamminggewicht $k/2$.[4] Die Größe der beiden Listen ist $\binom{m\ell/2}{k/2}$. Da das Sortieren von N Elementen $O(N \log N)$ Schritte benötigt, besitzt dieser Angriff eine Workload von $O\Big(\binom{m\ell}{k} \log(\binom{m\ell/2}{k/2}) \binom{m\ell/2}{k/2}^{-1}\Big)$.

3.1.3 Die Sicherheit von RSA-S1

Wir betrachten nun die Sicherheit von RSA-S1 gegen Angriffe durch generische Algorithmen.

Um die unserem Modell zugrunde liegende Verteilung der Tupel $(d, \boldsymbol{f}, \boldsymbol{d})$ zu beschreiben benötigen wir die folgende Definition.

Definition 3.2 *Wir schreiben wir im folgenden kurz* $d_{\boldsymbol{d}}(\boldsymbol{f})$ *für* $\sum f_i d_i \bmod \phi(n)$. *Für festes* $\boldsymbol{d}$ *sei* $S_{\boldsymbol{d}}$ *die Menge aller* $\boldsymbol{f} \in [0, 2^\ell)^m$ *mit Hamminggewicht* k, *für die* $\mathrm{ggT}(d_{\boldsymbol{d}}(\boldsymbol{f}), \phi(n)) = 1$ *und* $\mathrm{ggT}(f_i, \phi(n)) = 1$ *für mindestens ein* i *gilt.*

[4] Die Anzahl der $\boldsymbol{f}'$ mit $\mathrm{wt}(f'_1, \ldots, f'_{\frac{m}{2}}) = \mathrm{wt}(f'_{\frac{m}{2}+1}, \ldots, f'_m) = k/2$ ist $\binom{m\ell/2}{k/2}^2$

Bei gegebenen $\boldsymbol{d}$ ist $S_{\boldsymbol{d}}$ die Menge aller $\boldsymbol{f}$, für die das Tupel $(\boldsymbol{d}, \boldsymbol{f}, d)$ mit $d := d_{\boldsymbol{d}}(\boldsymbol{f})$ konsistent zur Spezifikation des Protokolls ist.

Wir definieren nun das Modell, in dem wir die Sicherheit von RSA-S1 untersuchen.

Definition 3.3 *Unter einem* generischen Angreifer auf RSA-S1 *verstehen wir einen generischen Algorithmus $\mathcal{A}$ auf $\mathbb{Z}_n^*$, der für den öffentlichen Parameter $\boldsymbol{d}$ und zufälliges $x \in \mathbb{Z}_n^*$ und zufälliges $\boldsymbol{f} \in S_{\boldsymbol{d}}$ die Eingaben x und $y := x^{d_{\boldsymbol{d}}(\boldsymbol{f})} \bmod n$ erhält und ein $d' \in \mathbb{Z}$ ausgibt.*

Sei ρ die Erfolgswahrscheinlichkeit von $\mathcal{A}$, d.h. die Wahrscheinlichkeit, daß $\mathcal{A}$ ein d' mit $x^{d'} = y \bmod n$ ausgibt. Die Wahrscheinlichkeit wird dabei über die zufällige Wahl von x und $\boldsymbol{f}$ genommen, nicht jedoch über eine zufällige Wahl von $\boldsymbol{d}$.[5] *Ebenso wie die Länge t von $\mathcal{A}$ kann die Erfolgswahrscheinlichkeit ρ von $\boldsymbol{d}$ abhängen. Die* Komplexität eines generischen Angreifers *definieren wir nun als den Erwartungswert von t/ρ bei zufälligem $\boldsymbol{d}$.*

Bemerkung. Es ist leider nicht möglich, den öffentlichen Schlüssel e in unser Modell zu integrieren: Die Ausgabe eines generischen Algorithmus kann in beliebiger Weise von den öffentlichen Parametern abhängen. Falls e und n öffentlicher Parameter sind, gibt es daher einen trivialen generischen Algorithmus der – unabhängig von den Eingaben – $d := e^{-1} \bmod \phi(n)$ ausgibt. Außer den bekannten Angriffen gegen RSA ist jedoch kein effizienter Angriff gegen RSA-S1 bekannt, der den öffentlichen Schlüssel e verwendet.

Die folgende Überlagung zeigt, daß die Verteilung der Tripel $(\boldsymbol{f}, \boldsymbol{d}, d)$, die wir unserem Modell zugrunde gelegt haben, identisch mit jener ist, die man erhält, wenn man zuerst d uniform aus $\mathbb{Z}_{\phi(n)}^*$ wählt und dann wie in der Beschreibung des Protokolls angegeben zerlegt:

Sei zunächst $\boldsymbol{f} \in [0, 2^\ell)^m$ mit $\mathrm{ggT}(f_{i_0}, \phi(n)) = 1$ für ein $i_0 \leq m$ fest gewählt. Außerdem seien alle d_i mit $i \neq i_0$ fest gewählt. Dann ist die Abbildung $d_{i_0} \to d_{\boldsymbol{d}}(\boldsymbol{f})$ injektiv, d.h. jedes $d_{i_0} \in \mathbb{Z}_{\phi(n)}$ definiert ein eindeutiges $d := d_{\boldsymbol{d}}(\boldsymbol{f}) \in \mathbb{Z}_{\phi(n)}$. Daher ist die Verteilung der Tripel $(\boldsymbol{f}, \boldsymbol{d}, d)$, die wir erhalten, unabhängig davon, ob wir zuerst uniform ein d_{i_0} aus $\mathbb{Z}_{\phi(n)}$ derart wählen, daß $d_{\boldsymbol{d}}(\boldsymbol{f}) \in \mathbb{Z}_{\phi(n)}^*$ gilt, und dann $d := d_{\boldsymbol{d}}(\boldsymbol{f})$ setzen, oder ob wir zuerst d uniform aus $\mathbb{Z}_{\phi(n)}^*$ wählen und dann d_{i_0} so setzen, daß $d = d_{\boldsymbol{d}}(\boldsymbol{f})$ gilt.[6]

Diese Gleichheit der Verteilungen bleibt auch dann erhalten, wenn wir in beiden Verteilungen zusätzlich unabhängig und uniform alle d_i mit $i \neq i_0$ aus $\mathbb{Z}_{\phi(n)}^*$ und $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k und $\mathrm{ggT}(f_{i_0}, \phi(n)) = 1$ wählen. Schließlich bleibt die Gleichheit der beiden Verteilungen auch dann erhalten, wenn wir i_0 zufällig aus $[1, m]$ wählen. Damit erhalten wir aber auch schon einerseits die Verteilung, die wir unserem Modell zugrunde

[5] Da $\boldsymbol{d}$ ein öffentlicher Parameter ist, kann ein generischer Angreifer für diesen optimiert sein.

[6] Die Eindeutigkeit der Abbildung $d_{i_0} \to d_{\boldsymbol{d}}(\boldsymbol{f})$ bleibt erhalten, wenn wir sie auf die $d_{i_0} \in \mathbb{Z}_{\phi(n)}$ mit $d_{\boldsymbol{d}}(\boldsymbol{f}) \in \mathbb{Z}_{\phi(n)}^*$ einschränken

gelegt haben, und andererseits die Verteilung der Tripel $(\boldsymbol{f}, \boldsymbol{d}, d)$ für uniform gewähltes d und eine zufällige Zerlegung von d gemäß der Beschreibung des Protokolls.

Die im letzten Abschnitt dargestellte Attacke läßt sich als generischer Angreifer beschreiben: Die von ihr berechneten Gruppenelemente sind die Einträge der beiden Listen. Seine Länge[7] ist daher $2\binom{m\ell/2}{k/2}$. Da seine Erfolgswahrscheinlichkeit mindestens $\binom{m\ell/2}{k/2}^2\binom{m\ell}{k}^{-1}$ ist, ergibt sich eine Komplexität[8] von höchstens $2\binom{m\ell}{k}\binom{m\ell/2}{k/2}^{-1}$.

Der folgende Satz zeigt, daß die Komplexität dieses generischen Angreifers fast optimal ist, sofern $(p-1)/2$ und $(q-1)/2$ prim sind. In diesem Fall ist die Bedingung $\mathrm{ggT}(f_i, \phi(n)) = 1$ gleichbedeutend mit $f_i = 1 \bmod 2$. Daher ist die Anzahl der Vektoren $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k und $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i durch $\binom{m\ell}{k} - \binom{m(\ell-1)}{k}$ gegeben.[9]

Satz 3.1 *Seien $r := (p-1)/2$ und $s := (q-1)/2$ prim und $r, s \gg \binom{m\ell}{k} \gg 1$. Dann hat jeder generische Angreifer auf RSA-S1 mindestens Komplexität $\frac{1}{2}\sqrt{N}$, wobei $N := \binom{m\ell}{k} - \binom{m(\ell-1)}{k}$ die Anzahl der $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k und $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i ist.*

Beweis. Wir nennen $\boldsymbol{d}$ *kollisionsfrei* falls $d_{\boldsymbol{d}}(\boldsymbol{f}) \neq d_{\boldsymbol{d}}(\boldsymbol{f}') \bmod r$ und $d_{\boldsymbol{d}}(\boldsymbol{f}) \neq d_{\boldsymbol{d}}(\boldsymbol{f}') \bmod s$ für alle $\boldsymbol{f} \neq \boldsymbol{f}' \in S_{\boldsymbol{d}}$ gilt.

Wir beweisen den Satz in drei Schritten: Zunächst zeigen wir, daß für die Länge t und die Erfolgswahrscheinlichkeit ρ eines generischen Angreifers

$$t/\rho > \sqrt{2|S_{\boldsymbol{d}}|} \tag{3.1}$$

gilt, sofern $\boldsymbol{d}$ kollisionsfrei ist. Dann schätzen wir die Wahrscheinlichkeit ab, daß ein zufälliges $\boldsymbol{d}$ kollisionsfrei ist. Schließlich leiten wir aus dieser Abschätung und (3.1) die Schranke für die Komplexität des generischen Angreifers ab.

Schritt 1. Wir zeigen zunächst, daß (3.1) gilt, falls $\boldsymbol{d}$ kollisionsfrei ist. Seien $\boldsymbol{d}$ und n fest gewählt, so daß $\boldsymbol{d}$ kollisionsfrei ist, und $\mathcal{A}$ ein generischer Angreifer mit Länge t und Erfolgswahrscheinlichkeit ρ. Für $\nu = 1, \ldots, t$ sei $F_\nu = x^{a_\nu} y^{b_\nu} \bmod n$ das von $\mathcal{A}$ in Runde ν berechnete Element.

Wir schätzen zunächst die Wahrscheinlichkeit nach oben ab, daß rs die Ordnung von x nicht teilt. Da r und s verschiedene Primzahlen sind,[10] kann das nur der Fall sein, wenn

[7] Die Länge eines generischen Algorithmus zählt nur die Anzahl der berechneten Elemente nicht jedoch den Aufwand für das Auffinden der Kollisionen

[8] Weder Länge noch Erfolgswahrscheinlichkeit dieses generischen Angreifers hängen von $\boldsymbol{d}$ ab

[9] Die Anzahl der $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k und $f_i = 0 \bmod 2$ für alle i ist $\binom{m(\ell-1)}{k}$

[10] Da p und q verschieden sind, sind auch r und s verschieden

entweder r die Ordnung von x nicht teilt oder wenn s die Ordnung von x nicht teilt. Da $\mathbb{Z}_n^*$ das direkte Produkt der beiden zyklischen Gruppen $\mathbb{Z}_p^*$ und $\mathbb{Z}_q^*$ ist und r ein Teiler der Ordnung von $\mathbb{Z}_p^*$ ist, teilt r bei zufälligem x höchstens mit Wahrscheinlichkeit $1/r$ die Ordnung von x nicht. Ebenso teilt s höchstens mit Wahrscheinlichkeit $1/s$ die Ordnung von x nicht. Insgesamt teilt also rs für ein zufälliges $x \in \mathbb{Z}_n^*$ höchstens mit Wahrscheinlichkeit $1/r + 1/s$ die Ordnung von x nicht.

Mit dem folgenden Fakt folgt nun $\rho \leq (\binom{t}{2} + 1)|S_{\boldsymbol{d}}|^{-1} + 1/r + 1/s$. Wegen $r, s \gg |S_{\boldsymbol{d}}|$ folgt daraus (3.1).

Fakt 3.2 *Falls rs die Ordnung von x teilt, ist die Ausgabe von $\mathcal{A}$ höchstens mit Wahrscheinlichkeit $(\binom{t}{2} + 1)/|S_{\boldsymbol{d}}|$ (über zufälliges $\boldsymbol{f} \in S_{\boldsymbol{d}}$) korrekt*

Beweis. Im folgenden sei x fest gewählt, so daß rs die Ordnung von x teilt. Wir nennen eine Kollision der von $\mathcal{A}$ berechneten Gruppenelemente *trivial*, wenn sie unabhängig von $\boldsymbol{f}$ (und damit auch von d) gilt. Da $\mathcal{A}$ aus trivialen Kollisionen keine Information über d erhält, können wir ohne Einschränkung davon ausgehen, daß keine trivialen Kollisionen auftreten.

Wir betrachten zunächst den Fall, daß in der gesamten Berechnung von $\mathcal{A}$ keine Kollision $F_\nu = F_\mu$ aufgetreten ist. Da die Ausgabe von $\mathcal{A}$ nur von der Menge der Kollisionen, den öffentlichen Parametern und t abhängt, gibt $\mathcal{A}$ in diesem Fall ein festes d' aus. Da rs die Ordnung von x teilt, ist diese Ausgabe nur dann korrekt, wenn $d' = d_{\boldsymbol{d}}(\boldsymbol{f}) \bmod rs$ gilt. Weil $\boldsymbol{d}$ kollisionsfrei ist, kann das aber nur für ein $\boldsymbol{f}$ gelten. Damit ist die Wahrscheinlichkeit, daß keine Kollision auftritt und $\mathcal{A}$ eine korrekte Ausgabe macht, höchstens $|S_{\boldsymbol{d}}|^{-1}$.

Wir zeigen nun, daß eine Kollision $F_\nu = F_\mu$ höchstens mit Wahrscheinlichkeit $\binom{t}{2}|S_{\boldsymbol{d}}|^{-1}$ auftritt. Daraus folgt dann unmittelbar der Fakt.

Angenommen, in den Runden $1, \ldots, \nu - 1$ ist keine Kollision aufgetreten. Falls nun $F_\nu = F_\mu$ für ein $\mu < \nu$ gilt, erhalten wir wegen $y = x^{d_{\boldsymbol{d}}(\boldsymbol{f})} \bmod n$

$$(a_\nu - a_\mu) + (b_\nu - b_\mu) d_{\boldsymbol{d}}(\boldsymbol{f}) = 0 \bmod \operatorname{ord}(x) \quad . \tag{3.2}$$

Da in den Runden $1, \ldots, \nu - 1$ keine Kollision aufgetreten ist, sind $a_\nu, a_\mu, b_\nu, b_\mu$ konstant. Die folgende Argumentation zeigt, daß (3.2) nur für ein einziges $\boldsymbol{f} \in S_{\boldsymbol{d}}$ gelten kann.

Die Kollision $F_\nu = F_\mu \bmod n$ ist nicht trivial, d.h. es gilt

$$b_\nu \neq b_\mu \bmod \operatorname{ord}(x) \quad . \tag{3.3}$$

Es können nun drei Fälle auftreten: $b_\nu \neq b_\mu \bmod r$, $b_\nu \neq b_\mu \bmod s$ oder $b_\nu = b_\mu \bmod rs$.

Falls $b_\nu \neq b_\mu \bmod r$ gilt, folgt aus (3.2) $d_{\boldsymbol{d}}(\boldsymbol{f}) = (a_\nu - a_\mu)/(b_\nu - b_\mu) \bmod r$. Da $\boldsymbol{d}$ kollisionsfrei ist, kann das nur für ein einziges $\boldsymbol{f} \in S_{\boldsymbol{d}}$ gelten. Analog dazu sieht man, daß im Fall $b_\nu \neq b_\mu \bmod s$ (3.2) nur für ein einziges $\boldsymbol{f} \in S_{\boldsymbol{d}}$ gelten kann. Im Fall $b_\nu = b_\mu \bmod rs$ muß wegen (3.3) $\mathrm{ord}(x) = 2rs$ gelten.[11] Damit vereinfacht sich (3.2) zu $a_\nu - a_\mu + (b_\nu - b_\mu) d_{\boldsymbol{d}}(\boldsymbol{f}) = 0 \bmod 2$. Da nach Definition von $S_{\boldsymbol{d}}$ jedoch $d_{\boldsymbol{d}}(\boldsymbol{f}) = 1 \bmod 2$ für alle $\boldsymbol{f} \in S_{\boldsymbol{d}}$ gilt,[12] ist diese Gleichung unabhängig von $\boldsymbol{f}$, d.h. die Kollision ist trivial. Das ist ein Widerspruch zur Voraussetzung.

Die Gleichung (3.2) gilt also höchstens für ein $\boldsymbol{f} \in S_{\boldsymbol{d}}$. Damit folgt, daß $F_\nu = F_\mu$ für festes $\mu < \nu$ höchstens mit Wahrscheinlichkeit $|S_{\boldsymbol{d}}|^{-1}$ gilt. Durch Summation über $\mu = 1, \ldots, \nu$ ergibt sich, daß höchstens mit Wahrscheinlichkeit $(\nu - 1)|S_{\boldsymbol{d}}|^{-1}$ in Runde ν eine Kollision auftritt. Indem wir über alle ν summieren, erhalten wir schließlich, daß $\mathcal{CO}_t \neq \emptyset$ höchstens mit Wahrscheinlichkeit $\binom{t}{2}|S_{\boldsymbol{d}}|^{-1}$ gilt. Damit ist der Fakt bewiesen.

$\square$

Schritt 2. Wir schätzen nun die Wahrscheinlichkeit ab, daß ein zufälliges $\boldsymbol{d}$ kollisionsfrei ist.

Da r prim ist, gilt $d_{\boldsymbol{d}}(\boldsymbol{f}) = d_{\boldsymbol{d}}(\boldsymbol{f}') \bmod r$ für feste $\boldsymbol{f} \neq \boldsymbol{f}' \in [0, 2^\ell)^m$ und zufälliges $\boldsymbol{d}$ höchstens mit Wahrscheinlichkeit $1/r$. Somit ist für zufälliges $\boldsymbol{d}$ die Wahrscheinlichkeit, daß es $\boldsymbol{f} \neq \boldsymbol{f}' \in S_{\boldsymbol{d}}$ mit $d_{\boldsymbol{d}}(\boldsymbol{f}) = d_{\boldsymbol{d}}(\boldsymbol{f}') \bmod r$ gilt, höchstens N^2/r.

Schritt 3. Wir beweisen nun den Satz, indem wir zeigen, daß

$$\mathbb{E}(t/\rho) > \sqrt{2N}\left(\frac{1}{2} - \epsilon - \frac{r+s}{2rs}\right) \tag{3.4}$$

mit $\epsilon := \binom{m\ell}{k}^2/r$ gilt, wobei $\mathbb{E}$ den Erwartungswert über zufälliges $\boldsymbol{d}$ bezeichnet. Die Behauptung folgt dann aus $\epsilon, 1/r, 1/s \ll 1$.

In Abhängigkeit von $\boldsymbol{d}$ sei die Zufallsvariable Ψ definiert als $|S_{\boldsymbol{d}}|$, falls $\boldsymbol{d}$ kollisionsfrei ist, und 0 sonst. Wegen $|S_{\boldsymbol{d}}| \leq N$ erhalten wir aus (3.1)

$$t/\rho > \sqrt{\frac{2}{N}}\,\Psi \ . \tag{3.5}$$

Wir schätzen nun $\mathbb{E}(\Psi)$ ab. Wegen $|S_{\boldsymbol{d}}| \leq N$ gilt $\mathbb{E}(\Psi) > \mathbb{E}(|S_{\boldsymbol{d}}|) - \epsilon N$. Die folgende Überlegung zeigt, daß $\mathbb{E}(|S_{\boldsymbol{d}}|) = \left(\frac{1}{2} - \frac{r+s}{2rs}\right) N$ und damit $\mathbb{E}(\Psi) > \left(\frac{1}{2} - \epsilon - \frac{r+s}{2rs}\right) N$ gilt. Wir erhalten daraus dann (3.4), indem wir in (3.5) auf beiden Seiten den Erwartungswert nehmen.

Sei $\boldsymbol{f} \in [0, 2^\ell)^m$ fest gewählt, so daß $f_i = 1 \bmod 2$ für mindestens ein i gilt. Dann ist $d_{\boldsymbol{d}}(\boldsymbol{f})$ für gleichverteiltes $\boldsymbol{d} \in \mathbb{Z}_{\phi(n)}^m$ ebenfalls gleichverteilt in $\mathbb{Z}_{\phi(n)}$. Damit

[11] $2rs = \mathrm{kgV}(p-1, q-1)$ ist die maximale Ordnung in $\mathbb{Z}_n^*$

[12] dies folgt aus $\mathrm{ggT}(d_{\boldsymbol{d}}(\boldsymbol{f}), \phi(n)) = 1$ für alle $\boldsymbol{f} \in S_{\boldsymbol{d}}$

gilt $\mathrm{ggT}(d_{\boldsymbol{d}}(\boldsymbol{f}), \phi(n)) = 1$ für zufälliges $\boldsymbol{d} \in \mathbb{Z}^m_{\phi(n)}$ mit Wahrscheinlichkeit $\frac{\phi(\phi(n))}{\phi(n)}$. Da $r = (p-1)/2$ und $s = (q-1)/2$ prim sind, ist das $\frac{(r-1)(s-1)}{2rs} = \left(\frac{1}{2} - \frac{r+s}{2rs}\right)$. Durch Summation über alle $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k, welche $f_i = 1 \bmod 2$ für mindestens ein i erfüllen, erhalten wir $\mathbb{E}(|S_{\boldsymbol{d}}|) = \left(\frac{1}{2} - \frac{r+s}{2rs}\right) N$.

□

Beispiele. Tabelle 3.1 vergleicht die untere Schranke GEN für generische Angreifer aus Theorem 3.1 mit der oberen Schranke ATT, welche durch die beschriebene Attacke gegeben ist.

m	k	l	ATT	GEN
20	30	12	$2^{65.7}$	$2^{62.4}$
42	26	8	$2^{66.5}$	$2^{63.2}$
40	40	4	$2^{65.4}$	$2^{62.0}$

Tabelle 3.1: Beispiele für die generische Komplexität der besten bekannten Attacke und der unteren Schranke für die Komplexität generischer Attacken).

3.2 RSA-S1M

Als Reaktion auf die Meet-in-the-middle Attacke von Pfitzmann und Waidner ([PW93]) veröffentlichten Matsumoto, Imai, Laih and Yen in [MILY93] zwei neue Server-unterstützte RSA Protokolle, RSA-S1M und RSA-S2M. Diese sind im wesentlichen zwei Runden-Varianten von RSA-S1 und RSA-S2. Wir betrachten zunächst nur RSA-S1M. Das Protokoll RSA-S2M behandeln wir in Abschnitt 3.3.

3.2.1 Das Protokoll RSA-S1M

Sei n das Produkt zweier großer verschiedener Primzahlen p und q. Die Sicherheitsparameter des Protokolles sind m, k und ℓ.

Die Zerlegung des geheimen Schlüssels. Vor der erstmaligen Ausführung des Protokolls zerlegt der Client seinen geheimen Schlüssels d in der folgenden Weise. Zuerst wählt er sich ein zufälliges $f \in \mathbb{Z}^*_{\phi(n)}$ und setzt $g := df^{-1} \bmod \phi(n)$. Dann wählt er zwei zufällige Vektoren $\boldsymbol{f} = (f_1, \ldots, f_m)$, $\boldsymbol{g} = (g_1, \ldots, g_m) \in [0, 2^\ell)^m$ mit Hamminggewicht k, so daß $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i und $\mathrm{ggT}(g_j, \phi(n)) = 1$ für mindestens ein j mit

$f_j = 0$ gilt[13]. Schließlich wählt er einen Vektor $\boldsymbol{d} = (d_1, \ldots, d_m) \in \mathbb{Z}^m_{\phi(n)}$ zufällig so, daß $f = \sum_{i=1}^m f_i d_i \bmod \phi(n)$ und $g = \sum_{j=1}^m g_j \bar{d}_j \bmod \phi(n)$ mit $\bar{d}_j := d_j(6m + 2j + 1) \bmod \phi(n)$ gilt.

Den Vektor $\boldsymbol{d}$ kann der Client nun öffentlich bekannt geben, die Vektoren $\boldsymbol{f}$ und $\boldsymbol{g}$ muß er geheim halten. Die Zerlegung

$$d = \Big(\sum_{i=1}^m f_i d_i\Big)\Big(\sum_{j=1}^m g_j \bar{d}_j\Big) \bmod \phi(n)$$

muß nicht bei jeder Ausführungen des Protokolls neu gewählt werden.[14]

Das Protokoll. Um eine Signatur zu erzeugen, tritt der Client mit einem Server in Kontakt und führt mit diesem das folgende Protokoll aus.

Preprocessing Der Client wählt ein zufälliges $h \in \mathbb{Z}_n^*$ und berechnet $u = h^{-g} \bmod n$.

1. Der Client sendet x, n und $\boldsymbol{d}$ an den Server.

2. Der Server antwortet mit $z_i = x^{d_i} \bmod n$ für $i = 1, \ldots, m$.

3. Der Client berechnet und sendet $z = h \cdot \prod z_i^{f_i} = h \cdot x^f \bmod n$ an den Server.

4. Der Server antwortet mit $v_j = z^{\bar{d}_j} \bmod n$ für$j = 1, \ldots, m$.

5. Der Client berechnet die Signatur y als $y = u \cdot \prod v_j^{g_j} = u \cdot z^g \bmod n$.

Für die Erzeugung einer Signatur benötigt der Client $2(k + \ell - 1)$ Multiplikationen modulo n, das Preprocessing nicht mitgerechnet.[15] Sein Speicherbedarf ist $2m + m \log n$ Bits.

Bemerkung. Das beschriebene Protokoll weicht vom Orginal in [MILY93] geringfügig ab. In Schritt 4 verwendet der Server die Exponenten $\bar{d}_j = d_j(6m + 2j + 1)$ anstelle der d_j. Diese Modifikationen hat die folgenden Gründe: Zum einen benötigen wir im Beweis von Lemma 3.8, daß für $i \neq j$ die Gleichung $ad_i\bar{d}_j = bd_j\bar{d}_i$ über $\mathbb{Z}$ betrachtet[16] keine Lösung (a, b) mit $|a/b| < 3/4$ oder $|a/b| > 4/3$ besitzt. Zum anderen benötigen wir, daß – sofern $r := (p-1)/2$ und $s := (q-1)/2$ prim sind und $m \ll p, q$ gilt – für uniformes $d_i \in \mathbb{Z}_{\phi(n)}$

[13] Diese Bedingung garantiert, daß es ein $\boldsymbol{d} \in \mathbb{Z}^m_{\phi(n)}$ mit $f = \sum_{i=1}^m f_i d_i \bmod \phi(n)$ und $g = \sum_{j=1}^m g_j \bar{d}_j \bmod \phi(n)$ gibt

[14] Um sich gegen aktive Angriffe zu schützen, kann eine gelegentliche Erneuerung der Zerlegung jedoch sinnvoll sein (siehe Abschnitt 3.4)

[15] Die Berechnung in Schritt 3 wird durch das „square and multiply“-Verfahren durchgeführt

[16] Wir nehmen dabei $d_i, d_j \in (0, \phi(n)]$ an

auch $\bar{d}_i$ gleichmäßig in $\mathbb{Z}_{\phi(n)}$ verteilt ist. Die Wahl $\bar{d}_j = d_j(6m + 2j + 1)$ erfüllt diese beiden Bedingungen, da $(6m + 2i + 1)/(6m + 2j + 1) \in [3/4, 4/3]$ für alle $1 \leq i, j \leq m$ gilt, und $6m + 2i + 1$ teilerfremd zu $\phi(n)$ ist, sofern $r := (p-1)/2$ und $s := (q-1)/2$ prim sind und $m \ll p, q$ gilt.

Um Multi-Runden-Angriffe zu vermeiden, muß der Client in jeder Ausführung des Protokolles ein neues h wählen. Anderenfalls kann ein Angreifer aus den Daten $(x^{[1]}, y^{[1]}, z^{[1]})$ und $(x^{[1]}, y^{[1]}, z^{[1]})$ von zwei Ausführungen des Protokolls $z^{[1]}/z^{[2]} = (x^{[1]}/x^{[2]})^f \bmod n$ berechnen. Mit einer Meet-in-the-middle Attacke kann er dann zunächst $\boldsymbol{f}$ aus $(x^{[1]}/x^{[2]})$, $(x^{[1]}/x^{[2]})^f$ und $\boldsymbol{d}$ berechnen und danach $\boldsymbol{g}$ aus $y^{[1]}/y^{[2]} = (x^{[1]}/x^{[2]})^d$, $(x^{[1]}/x^{[2]})^f$ und $\boldsymbol{d}$ bestimmen. Die Workload dieser Attacke ist nur doppelt so groß wie die der Meet-in-the-middle Attacke gegen RSA-S1.

Da h und u von der Nachricht unabhängig sind, kann das Paar h, u auch mit Hilfe von Precomputation berechnet werden. Diese Verfahren sind im folgenden Kapitel beschrieben. Da h uniform aus $\mathbb{Z}_n^*$ gewählt ist, gibt z keine Information über den geheimen Schlüssel d preis.

3.2.2 Der beste bekannte Angriff gegen RSA-S1M

In [LL95] zeigen Lim and Lee, daß die Ideen von [PW93] auch auf RSA-S1M anwendbar sind. Sie beschreiben eine Meet-in-the-middle Attacke gegen RSA-S1M. Wir beschreiben hier eine etwas effizientere Variante dieser Attacke, die Ideen von [vOW96] verwendet.

Seien $m\ell$ und k gerade. Nach Definition des Protokolls gibt es $\boldsymbol{f}, \boldsymbol{g} \in [0, 2^\ell)^m$ mit Hamminggewicht k, so daß

$$x^d = \prod_{j=1}^{m} x^{f g_j \bar{d}_j} \bmod n$$

gilt.

Der Angreifer berechnet für alle $(f_1', \ldots, f_m')$ mit Hamminggewicht k und für alle $(g_1', \ldots, g_{\frac{m}{2}}')$ mit Hamminggewicht $k/2$ die Werte

$$\prod_{j \leq m/2} x^{f' g_j' \bar{d}_j} \bmod n$$

mit $f' = \sum f_i' d_i$ und sortiert sie. Danach berechnet er für alle $(g_{\frac{m}{2}+1}', \ldots, g_m')$ mit Hamminggewicht $k/2$ die Werte

$$y \Big(\prod_{j > m/2} x^{f' g_j' \bar{d}_j} \Big)^{-1} \bmod n$$

und sortiert diese ebenfalls. Falls $(g_1, \ldots, g_{\frac{m}{2}})$ Hamminggewicht $k/2$ hat, gibt es eine Kollision zwischen den beiden Listen und der Angreifer kann daraus den geheimen Schlüssel d ermitteln. (Da beide Listen sortiert sind, kann er eine Kollision in linearer Zeit finden.)

Mit Wahrscheinlichkeit $\binom{m\ell/2}{k/2}^2 / \binom{m\ell}{k}$ hat $(g_1, \ldots, g_{\frac{m}{2}})$ Hamminggewicht $k/2$. Die Größe der beiden Listen ist $\binom{m\ell}{k}\binom{m\ell/2}{k/2}$. Da das Sortieren von N Elementen $O(N \log N)$ Schritte benötigt, besitzt dieser Angriff eine Workload von $2\binom{m\ell}{k}^2 \log(\binom{m\ell}{k}\binom{m\ell/2}{k/2})/\binom{m\ell/2}{k/2}$.

3.2.3 Die Komplexität von Meet-in-the-middle-Angriffen

Wir betrachten nun die Sicherheit von RSA-S1M gegen Angriffe durch generische Algorithmen. Um die unserem Modell zugrunde liegende Verteilung der Tupel $(d, \boldsymbol{f}, \boldsymbol{d})$ zu beschreiben benötigen wir die folgende Definition.

Definition 3.4 *Wir schreiben im folgenden kurz* $d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{d})$ *für* $\sum f_i g_j d_i \bar{d}_j \bmod \phi(n)$. *Für festes* $\boldsymbol{d}$ *sei außerdem* $S'_{\boldsymbol{d}}$ *die Menge der Paare* $(\boldsymbol{f}, \boldsymbol{g})$ *mit* $\boldsymbol{f}, \boldsymbol{g} \in [0, 2^\ell)^m$ *und* $\mathrm{wt}(\boldsymbol{f}) = \mathrm{wt}(\boldsymbol{g}) = k$, *so daß* $\mathrm{ggT}(f_i, \phi(n)) = 1$ *für mindestens ein* i *und* $\mathrm{ggT}(g_j, \phi(n)) = 1$ *für mindestens ein* j *mit* $f_j = 0$ *gilt.*

Bei gegebenen $\boldsymbol{d}$ ist $S'_{\boldsymbol{d}}$ die Menge aller $(\boldsymbol{f}, \boldsymbol{g})$, für die das Tupel $(\boldsymbol{d}, \boldsymbol{f}, \boldsymbol{g}, d)$ mit $d := d_{\boldsymbol{d}}(\boldsymbol{f}))$ konsistent zur Spezifikation des Protokolls ist.

Wir können die Definition eines generischen Angreifers in kanonischer Weise vom Fall RSA-S1 zu RSA-S1M übertragen.

Definition 3.5 *Unter einem* generischen Angreifer auf RSA-S1M *verstehen wir einen generischen Algorithmus* $\mathcal{A}$ *auf* $\mathbb{Z}_n^*$, *der für den öffentlichen Parameter* $\boldsymbol{d}$ *und zufälliges* $x \in \mathbb{Z}_n^*$ *und zufälliges* $(\boldsymbol{f}, \boldsymbol{g}) \in S'_{\boldsymbol{d}}$ *die Eingaben* x *und* $y := x^{d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g})} \bmod n$ *erhält und ein* $d' \in \mathbb{Z}$ *berechnet.*

Abhängig von $\boldsymbol{d}$ *sei* t *die Länge und* ρ *die Erfolgswahrscheinlichkeit von* $\mathcal{A}$. *Die Erfolgswahrscheinlichkeit wird dabei über* $\boldsymbol{f}$, $\boldsymbol{g}$ *und* x *genommen. Dann definieren wir die* Komplexität von $\mathcal{A}$ *als den Erwartungswert von* t/ρ *über zufälliges* $\boldsymbol{d}$.

Wie im Fall von RSA-S1 kann man auch hier zeigen, daß die Verteilung der Tripel $(\boldsymbol{f}, \boldsymbol{d}, d)$, die wir unserem Modell zugrunde gelegt haben, identisch mit mit jener ist, die man erhält, wenn man zuerst d uniform aus $\mathbb{Z}_{\phi(n)}^*$ wählt und dann wie in der Beschreibung des Protokolls angegeben zerlegt (siehe Abschnitt 3.1). Bei der Übertragung der Argumentation muß man lediglich beachten, daß für uniform und unabhängig aus $\mathbb{Z}_{\phi(n)}^*$ gewählte f, g auch $d = f \cdot g$ uniform in $\mathbb{Z}_{\phi(n)}^*$ verteilt ist.

Analog zum Fall RSA-S1 können wir eine untere Schranke von $\Omega(\binom{m\ell}{k})$ für die Komplexität von generischen Angreifern gegen RSA-S1M beweisen. Der Beweis ist analog zum Beweis von Theorem 3.1. Diese Schranke ist jedoch nicht scharf. Der im letzten Abschnitt beschriebene Angriff läßt als generischer Angreifer beschreiben: Die von ihm berechneten Gruppenelemente sind die Einträge der beiden Listen. Seine Länge[17] ist daher $2\binom{m\ell}{k}\binom{m\ell/2}{k/2}$. Da seine Erfolgswahrscheinlichkeit mindestens $\binom{m\ell/2}{k/2}^2\binom{m\ell}{k}^{-1}$ ist, ist seine Komplexität[18] höchstens $2\binom{m\ell}{k}^2\binom{m\ell/2}{k/2}^{-1}$.

Wenn wir diesen Angriff betrachten, fällt auf, daß alle berechneten Gruppenelemente die Form x^b oder yx^b haben. Alle bekannten generischen Angreifer gegen RSA-S1M besitzen diese Eigenschaft.

Diese Beobachtung führt uns zu der folgenden Definition.

Definition 3.6 *Unter einem* Meet-in-the-middle Angreifer auf RSA-S1M *verstehen wir einen generischen Angreifer auf RSA-S1M, der nur Gruppenelemente der Form x^b oder yx^b berechnet.*

Wir zeigen eine untere Schranke für Meet-in-the-middle Angreifer gegen RSA-S1M im Fall $\ell = 1$. Diese Schranke ist bis auf einen konstanten Faktor scharf. Im Fall $\ell > 1$ gelingt der Beweis dieser unteren Schranke aus technischen Gründen nicht. Es ist jedoch auch für $\ell > 1$ kein Meet-in-the-middle Angreifer gegen RSA-S1M bekannt, der eine kleinerere Komplexität als unsere Schranke besitzt.

Satz 3.3 *Seien $(p-1)/2$ und $(q-1)/2$ prim und $2^{40} \leq \binom{m}{k} \ll n^{1/12}$. Dann hat jeder Meet-in-the-middle Angreifer gegen RSA-S1M mindestens Komplexität $2^{-7}\binom{m}{k}^{3/2}$.*

Der Beweis gliedert sich folgendermaßen: Seien $r := (p-1)/2$ und $s := (q-1)/2$ prim. Wir nennen $\boldsymbol{d}$ *kollisionsfrei* falls $d_{\boldsymbol{d}}(\boldsymbol{f},\boldsymbol{g}) \neq d_{\boldsymbol{d}}(\boldsymbol{f}',\boldsymbol{g}') \bmod rs$ für alle $(\boldsymbol{f},\boldsymbol{g}) \neq (\boldsymbol{f}',\boldsymbol{g}') \in S'_{\boldsymbol{d}}$ gilt. Zuerst zeigen wir, wie ein Meet-in-the-middle Angreifer abhängig von $\boldsymbol{d}$ einen Graphen G definiert. Die Anzahl der Knoten von G ist dabei die Länge von $\mathcal{A}$. In Lemma 3.4 zeigen wir dann, daß für kollisionsfreies $\boldsymbol{d}$ die Anzahl der Kanten von G eine obere Schranke für die Erfolgswahrscheinlichkeit von $\mathcal{A}$ liefert. Indem wir die Struktur der Menge $S'_{\boldsymbol{d}}$ ausnutzen, erhalten wir dann in Lemma 3.6 eine untere Schranke für die Anzahl der Kanten. Schließlich kombinieren wir im Beweis von Theorem 3.3 die Resultate der Lemmata.

Sei $\boldsymbol{d}$ fest gewählt und $\mathcal{A}$ ein Meet-in-the-middle Angreifer der Länge t. Für festes x mit Erfolgswahrscheinlichkeit ρ, genommen über eine zufällige Wahl $(\boldsymbol{f},\boldsymbol{g}) \in S'_{\boldsymbol{d}}$. Dann hängt

[17] Da die Ausgabe eines generischen Algorithmus in beliebiger Weise von der Menge der Kollisionen abhängen darf, muß der Algorithmus die Listen nicht sortieren

[18] Weder Länge noch Erfolgswahrscheinlichkeit dieses generischen Angreifers hängen von $\boldsymbol{d}$ ab

das Element $F_\nu = y^{a'_\nu} x^{b'_\nu} \bmod n$, welches der Angreifer in Schritt ν berechnet, nur von der Menge der Kollisionen $F_i = F_j$ ab.

Für $\nu = 1, \ldots, t$ seien (a_ν, b_ν) die Exponenten, mit denen $\mathcal{A}$ das Element $F_\nu = y^{a_\nu} x^{b_\nu} \bmod n$ berechnet, falls in den ersten $\nu - 1$ Runden keine Kollision aufgetreten ist. Man beachte, daß $(a_1, b_1), \ldots, (a_t, b_t)$ unabhängig von der Eingabe (x, y) sind. Da $\mathcal{A}$ ein Meet-in-the-middle Angreifer ist, gilt $a_\nu \in \{0, 1\}$ für alle ν. Eine Kollision $F_i = F_j$ mit $a_i = a_j$ impliziert die Gleichung $x^{b_i} = x^{b_j} \bmod n$, d.h. die Kollision gilt unabhängig von d und damit auch von $(\boldsymbol{f}, \boldsymbol{g}) \in S'_{\boldsymbol{d}}$. Eine solche Kollision nennen wir im folgenden *triviale Kollision.*

Die Konstruktion des Graphen. In Abhängigkeit von $\boldsymbol{d}$ definieren wir einen Graphen $G = (V, E)$ mit Knoten $v_1, \ldots, v_t$ wie folgt: Für $i < j$ gilt $(v_i, v_j) \in E$ genau dann, falls es ein $x \in \mathbb{Z}_n^*$ und $(\boldsymbol{f}, \boldsymbol{g}) \in S'_{\boldsymbol{d}}$ gibt, so daß rs die Ordnung von x teilt und $F_i = F_j$ die erste nicht-triviale Kollision bei Eingabe von $x, x^{d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g})}$ ist.

Sei nun $\boldsymbol{d}$ kollisionsfrei. Dann ist das Paar $(\boldsymbol{f}, \boldsymbol{g})$, durch das eine Kante definiert ist, eindeutig bestimmt. Falls nämlich für eine Eingabe $x, x^{d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g})}$ eine Kollision $F_i = F_j$ auftritt und rs die Ordnung von x teilt, so folgt $b_i - b_j = (a_j - a_i) d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g}) \bmod rs$. Da die Kollision nicht-trivial und $\boldsymbol{d}$ kollisionsfrei ist, kann diese Gleichung nur für ein Paar $(\boldsymbol{f}, \boldsymbol{g})$ gelten. Wir können daher die Kante (v_i, v_j) mit $(\boldsymbol{f}, \boldsymbol{g})$ markieren. Falls ein Paar $(\boldsymbol{f}, \boldsymbol{g})$ mehrmals als Marke von Kanten erscheint, entfernen wir alle bis auf eine dieser Kanten.

Das folgende Lemma zeigt die Verbindung zwischen der Größe von E und der Erfolgswahrscheinlichkeit von $\mathcal{A}$ auf.

Lemma 3.4 *Falls $\boldsymbol{d}$ kollisionsfrei ist, gilt*

$$\rho \leq \frac{|E| + 2}{|S'_{\boldsymbol{d}}|} \quad . \tag{3.6}$$

Beweis. Sei $\boldsymbol{d}$ fest gewählt und kollisionsfrei. Wir zeigen zunächst, daß die Ausgabe von $\mathcal{A}$ höchstens mit Wahrscheinlichkeit $(|E| + 1)/|S'_{\boldsymbol{d}}|$ (über eine zufällige Wahl von $(\boldsymbol{f}, \boldsymbol{g}) \in S'_{\boldsymbol{d}}$) korrekt ist, sofern rs die Ordnung von x teilt. Dann zeigen wir, daß rs höchstens mit Wahrscheinlichkeit $1/r + 1/s$ nicht die Ordnung von x teilt. Wegen $1/r + 1/s < \binom{m}{k}^{-2} < 1/|S'_{\boldsymbol{d}}|$ folgt daraus (3.6).

Angenommen rs teilt die Ordnung von x und für die Eingabe $x, x^{d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g})}$ tritt in der Berechnung von $\mathcal{A}$ eine nicht-triviale Kollision der Gruppenelemente F_ν auf. Dann gibt es eine Kante, welche mit $(\boldsymbol{f}, \boldsymbol{g})$ markiert ist. Da jede Kante nur eine Marke besitzt und jedes Paar $(\boldsymbol{f}', \boldsymbol{g}')$ höchstens einmal als Marke in E erscheint, ist für zufälliges $(\boldsymbol{f}, \boldsymbol{g}) \in S'_{\boldsymbol{d}}$ die Wahrscheinlichkeit, daß es eine mit $(\boldsymbol{f}, \boldsymbol{g})$ markierte Kante gibt, $|E|/|S'_{\boldsymbol{d}}|$.

Falls andererseits keine nicht-triviale Kollision auftritt, gibt $\mathcal{A}$ ein eindeutig bestimmtes d' aus. Da rs die Ordnung von x teilt und $\boldsymbol{d}$ kollisionsfrei ist, ist diese Ausgabe höchstens für ein Paar $(\boldsymbol{f},\boldsymbol{g}) \in S'_{\boldsymbol{d}}$ korrekt (d.h. es gilt $x^{d'} = x^{d_{\boldsymbol{d}}(\boldsymbol{f},\boldsymbol{g})} \bmod n$). Damit ist insgesamt bei einer zufälligen Wahl von $(\boldsymbol{f},\boldsymbol{g}) \in S'_{\boldsymbol{d}}$ die Ausgabe von $\mathcal{A}$ höchstens mit Wahrscheinlichkeit $(|E|+1)/|S'_{\boldsymbol{d}}|$ korrekt.

Wir schätzen nun die Wahrscheinlichkeit dafür ab, daß rs die Ordnung von x nicht teilt. Dies ist nur dann der Fall, wenn $x \in \{z^r \mid z \in \mathbb{Z}_n^*\}$ oder $x \in \{z^s \mid z \in \mathbb{Z}_n^*\}$ gilt.[19] Das gilt jedoch höchstens mit Wahrscheinlichkeit $1/r + 1/s$.

□

Das folgende Lemma zeigt, daß ein zufälliges $\boldsymbol{d}$ fast immer kollisionsfrei ist.

Lemma 3.5 *Bei zufälliger Wahl ist* $\boldsymbol{d}$ *mit Wahrscheinlichkeit mindestens* $1 - 4\binom{m}{k}^4/rs$ *kollisionsfrei.*

Beweis. Wegen $\mathbb{Z}_{rs} \simeq \mathbb{Z}_r \times \mathbb{Z}_s$ ist die Gleichheit in

$$d_{\boldsymbol{d}}(\boldsymbol{f},\boldsymbol{g}) = d_{\boldsymbol{d}}(\boldsymbol{f}',\boldsymbol{g}') \bmod rs \tag{3.7}$$

äquivalent zur Gleichheit $\bmod\, r$ und $\bmod\, s$. Für feste $\boldsymbol{f},\boldsymbol{g},\boldsymbol{f}',\boldsymbol{g}' \in \{0,1\}^m$ impliziert (3.7) deshalb quadratische Gleichungen über $\mathbb{Z}_r$ und $\mathbb{Z}_s$. Falls $(\boldsymbol{f},\boldsymbol{g}) \neq (\boldsymbol{f}',\boldsymbol{g}')$ gilt, sind diese Gleichungen nichttrivial. Da r und s verschiedene Primzahlen sind, gilt (3.7) für zufälliges $\boldsymbol{d} \in \mathbb{Z}_{\phi(n)}^m$ höchstens mit Wahrscheinlichkeit $(2/r)(2/s)$. Damit ist für zufälliges $\boldsymbol{d}$ die Wahrscheinlichkeit, daß es $\boldsymbol{f},\boldsymbol{g},\boldsymbol{f}',\boldsymbol{g}' \in \{0,1\}^m$ mit Hamminggewicht k und $(\boldsymbol{f},\boldsymbol{g}) \neq (\boldsymbol{f}',\boldsymbol{g}')$ gibt, welche (3.7) erfüllen, durch $4\binom{m}{k}^4/rs$ beschränkt.

□

Indem wir nachweisen, daß bestimmte Zykel in G nicht vorkommen, erhalten wir den folgenden Satz. Der Beweis folgt in Abschnitt 3.2.4.

Satz 3.6 *Sei* $2^{40} \leq \binom{m}{k}$. *Dann gilt für zufälliges* $\boldsymbol{d}$ *mindestens mit Wahrscheinlichkeit* $1 - 8\binom{m}{k}^{12}/rs$

$$|V| > 2^{-4.75}|E|\binom{m}{k}^{-1/2} \quad . \tag{3.8}$$

Wir beweisen nun Satz 3.3.

[19] Da p und q verschieden sind, sind auch $r = (p-1)/2$ und $s = (q-1)/2$ verschieden

Beweis von Satz 3.3. Wir definieren die Zufallsvariable Ψ wie folgt. In Abhängigkeit von x und $\boldsymbol{d}$ sei $\Psi = |S'_{\boldsymbol{d}}|$ falls (3.6) und (3.8) gelten, und 0 sonst.

Falls (3.6) und (3.8) gelten, erhalten wir $t > 2^{-4.75}(\rho\Psi - 2)\binom{m}{k}^{-1/2}$. Da $\binom{m}{k}$ groß ist, können wir daraus

$$t/\rho \geq 2^{-4.8}\Psi\binom{m}{k}^{-1/2} \tag{3.9}$$

folgern.[20] Falls andererseits (3.6) oder (3.8) nicht gilt, ist (3.9) wegen $\Psi = 0$ trivial.

Wir zeigen nun $\mathbb{E}(\Psi) \geq 2^{-2.2}\binom{m}{k}^2$, wobei der Erwartungswert über zufällig gewähltes $\boldsymbol{d}$ genommen wird. Mit (3.9) folgt dann die Behauptung des Satzes.

Nach Lemma 3.4, Lemma 3.5 und Satz 3.6 gilt $\Psi = 0$ höchstens mit Wahrscheinlichkeit $\delta := (8\binom{m}{k}^{12} + 4\binom{m}{k}^4)/rs \ll 1$. Wegen $|S'_{\boldsymbol{d}}| \leq \binom{m}{k}^2$ können wir

$$\mathbb{E}(\Psi) \geq \mathbb{E}(|S'_{\boldsymbol{d}}|) - \delta\binom{m}{k}^2$$

abschätzen. Der folgende Fakt zeigt, daß $\mathbb{E}(|S'_{\boldsymbol{d}}|) \geq (\frac{1}{4} - \epsilon)\binom{m}{k}^2$ mit $0 < \epsilon \ll 1$ gilt. Wir erhalten damit

$$\mathbb{E}(\Psi) \geq \left(\frac{1}{4} - \epsilon - \delta\right)\binom{m}{k}^2 .$$

Da ϵ und δ sehr klein sind, folgt daraus $\mathbb{E}(\Psi) \geq 2^{-2.2}\binom{m}{k}^2$. Damit ist der Satz bewiesen.

Fakt 3.7 *Es gilt* $\mathbb{E}(|S'_{\boldsymbol{d}}|) \geq \left(\frac{1}{4} - \binom{m}{k}^{-1}\right)\binom{m}{k}^2$.

Beweis. Zunächst beachte man, daß sich $S'_{\boldsymbol{d}}$ für $\ell = 1$ als die Menge der Paare $(\boldsymbol{f}, \boldsymbol{g})$ mit $\boldsymbol{f} \neq \boldsymbol{g} \in \{0,1\}^m$ und $\mathrm{wt}(\boldsymbol{f}) = \mathrm{wt}(\boldsymbol{g}) = k$ beschreiben läßt.[21]

Seien $\boldsymbol{f} \neq \boldsymbol{g} \in [0,1]^m$ mit Hamminggewicht k fest gewählt. Dann sind $\sum f_i d_i \bmod \phi(n)$ und $\sum g_j \bar{d}_j \bmod \phi(n)$ für gleichverteiltes $\boldsymbol{d} \in \mathbb{Z}^m_{\phi(n)}$ gleichmäßig und unabhängig[22] in $\mathbb{Z}_{\phi(n)}$ verteilt. Damit gelten $\mathrm{ggT}(\sum f_i d_i, \phi(n)) = 1$ und $\mathrm{ggT}(\sum g_j \bar{d}_j, \phi(n)) = 1$ für zufälliges $\boldsymbol{d} \in \mathbb{Z}^m_{\phi(n)}$ unabhängig voneinander jeweils mit mit Wahrscheinlichkeit $\frac{\phi(\phi(n))}{\phi(n)}$. Es folgt, daß $\mathrm{ggT}(d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g}) = 1$ für zufälliges $\boldsymbol{d}$ mit Wahrscheinlichkeit $\left(\frac{\phi(\phi(n))}{\phi(n)}\right)^2$ gilt. Da $r = (p-1)/2$ und $s = (q-1)/2$ prim sind, ist das $\frac{(r-1)(s-1)}{2rs} = \left(\frac{1}{2} - \frac{r+s}{2rs}\right)^2$.

Durch Summation über alle $\boldsymbol{f} \neq \boldsymbol{g} \in [0,1]^m$ mit Hamminggewicht k erhalten schließlich

$$\mathbb{E}(|S'_{\boldsymbol{d}}|) \geq \binom{m}{k}\left(\binom{m}{k} - 1\right)\left(\frac{1}{2} - \frac{r+s}{2rs}\right)^2 .$$

[20] Im Fall $\rho\Psi < 2^6$ folgt (3.9) sofort. Im Fall $\rho\Psi \geq 2^6$ gilt dagegen $\rho\Psi < 2^{0.05}(\rho\Psi - 2)$

[21] Die Bedingung, daß $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i und $\mathrm{ggT}(g_j, \phi(n)) = 1$ für mindestens ein j mit $f_j = 0$ gilt, vereinfacht sich wegen $f_i, g_j \in \{0,1\}$ für alle i, j zu $\boldsymbol{f} \neq \boldsymbol{g}$.

[22] Hier benötigen wir $\boldsymbol{f} \neq \boldsymbol{g}$

Wegen $r, s \gg \binom{m}{k}$ ergibt sich die Behauptung.

□

Beispiele. Tabelle 3.2 vergleicht die untere Schranke MM für Meet-in-the-middle Angreifer aus Theorem 3.3 mit der oberen Schranke ATT, welche durch die beschriebene Attacke gegeben ist.

m	k	ATT	MM
50	20	$2^{70.2}$	$2^{61.1}$
60	15	$2^{70.4}$	$2^{61.4}$

Tabelle 3.2: Beispiele für die generische Komplexität der besten bekannten Attacke und der unteren Schranke für die Komplexität von Meet-in-the-middle Angreiferen.

3.2.4 Der Beweis von Satz 3.6

Die Haupteigenschaft von G, die wir ausnutzen werden, ist die Nichtexistenz bestimmter Zykel. Wir werden dann eine Variante des wohlbekannten Resultats beweisen, daß Graphen mit v Knoten, die keine Zykel der Länge $L \leq 2\kappa$ enthalten, höchstens $O(v^{1+1/\kappa})$ viele Kanten besitzen ([Bol78]). Die Nichtexistenz bestimmter Zykel beweisen wir in den folgenden Lemmata.

Nach Konstruktion des Graphen ist jeder Knoten $v_i \in V$ $(1 \leq i \leq t)$ mit einem von $\mathcal{A}$ berechneten Element $F_i = y^{a_i} x^{b_i} \bmod n$ assoziiert. Da $\mathcal{A}$ ein Meet-in-the-middle Angriff ist, gilt $a_i \in \{0, 1\}$ für alle $1 \leq i \leq t$. Wir definieren V_1 als die Menge aller Knoten v_i mit $a_i = 1$ und V_2 als die Menge aller Knoten v_i mit $a_i = 0$. Da die Kanten in G mit nicht-trivialen Kollisionen korrespondieren, ist $G = (V_1, V_2, E)$ bipartit.

Das folgende Lemma zeigt, daß jeder Zykel mit großer Wahrscheinlichkeit eine Gleichung in den Marken der Kanten impliziert.

Lemma 3.8 *Sei $(v_{i_1}, \ldots, v_{i_L}, v_{i_{L+1}} = v_{i_1})$ ein Zykel der Länge L in G. Für $l = 1, \ldots, L$ sei $(\boldsymbol{f}^l, \boldsymbol{g}^l)$ die Marke von $(v_{i_l}, v_{i_{l+1}})$. Dann gilt*

$$\sum_{l=1}^{L} (-1)^l d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = 0 \bmod rs.$$

Beweis. Ohne Einschränkung können wir $a_{i_1} = 1$ annehmen. Nach Konstruktion gilt $(a_{i_{l+1}} -$

$a_{i_l})d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = b_{i_l} - b_{i_{l+1}} \bmod rs$ für $l = 1, \ldots, L$. Durch Summation über l erhalten wir

$$\sum_{l=1}^{L} (a_{i_{l+1}} - a_{i_l}) d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = 0 \bmod rs.$$

Wegen $a_{i_{l+1}} - a_{i_l} = (-1)^l$ folgt daraus die Behauptung.

□

Die durch einen Zykel gegebenen Gleichungen sind noch von $\boldsymbol{d}$ abhängig. Das folgende Lemma zeigt, wie sich diese Abhängigkeit eliminieren läßt.

Definition 3.7 *Das* Tensorprodukt $\boldsymbol{v} \otimes \boldsymbol{w}$ *von zwei Vektoren* $\boldsymbol{v}, \boldsymbol{w} \in \{0,1\}^m$ *ist definiert als die Matrix* $(f_i g_j)_{ij}$.

Lemma 3.9 *Sei* $\boldsymbol{f}^1, \boldsymbol{g}^1, \ldots, \boldsymbol{f}^8, \boldsymbol{g}^8 \in \{0,1\}^m$ *mit*

$$\sum_{l=1}^{8} (-1)^l \, \boldsymbol{f}^l \otimes \boldsymbol{g}^l \neq 0$$

und seien $d_1, \ldots, d_m$ *unabhängig und zufällig aus* $\mathbb{Z}_{\phi(n)}$ *gewählt. Dann gilt*

$$\sum_{l=1}^{8} (-1)^l \, d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = 0 \bmod rs \tag{3.10}$$

höchstens mit Wahrscheinlichkeit $4/rs$.

Beweis. Wir zeigen, daß (3.10) modulo r höchstens mit Wahrscheinlichkeit $2/r$ gilt. Analog dazu folgt, daß Gleichheit modulo s höchstens mit Wahrscheinlichkeit $2/s$ gilt. Da r und s verschiedene Primzahlen sind, folgt damit die Behauptung.

Modulo r läßt sich (3.10) als

$$\sum_{i,j=1}^{m} \bar{c}_{i,j} \, d_i \bar{d}_j = 0 \bmod r$$

mit $\bar{c}_{i,j} = \sum_{l=1}^{8} (-1)^l \, f_i^l \, g_j^l$ für $1 \le i \le j \le m$ schreiben. Wegen $\bar{d}_j = (6m + 2j + 1) d_j$ für $j = 1, \ldots, m$ können wir dies als

$$\sum_{i \le j} c_{i,j} \, d_i d_j = 0 \bmod r \tag{3.11}$$

mit

$$\begin{aligned} c_{i,j} &= \bar{c}_{i,j} \, (6m + 2j + 1) + \bar{c}_{j,i} (6m + 2i + 1) \bmod r \\ c_{i,i} &= \bar{c}_{i,i} (6m + 2i + 1) \bmod r \end{aligned}$$

für $1 \leq i < j \leq m$ schreiben. Nach Voraussetzung gibt es ein $\bar{c}_{i_0,j_0} \neq 0$. Wir zeigen, daß es auch ein $c_{i'_0,j'_0} \neq 0 \bmod r$ gibt. Da $\mathbb{Z}_r$ ein Körper ist und die Residuen $d_i \bmod r$ gleichmäßig und unabhängig in $\mathbb{Z}_r$ verteilt sind, kann dann (3.11) höchstens mit Wahrscheinlichkeit $2/r$ gelten.

Für $i_0 = j_0$ ist das offensichtlich. Im Fall $i_0 \neq j_0$ setzen wir $i'_0 = \min(i_0, j_0)$ und $j'_0 = \max(i_0, j_0)$. Angenommen, es gilt $c_{i'_0,j'_0} = \bar{c}_{i'_0,j'_0}(6m+2j'_0+1)+\bar{c}_{j'_0,i'_0}(6m+2i'_0+1) = 0 \bmod r$. Wegen $r \gg 64m$ gilt diese Gleichung auch über $\mathbb{Z}$ und wir erhalten

$$\frac{6m + 2i'_0 + 1}{6m + 2j'_0 + 1} = \frac{\bar{c}_{i'_0,j'_0}}{-\bar{c}_{j'_0,i'_0}}.$$

Da $|\bar{c}_{i,j}| \leq 4$ für alle i, j gilt, ist die rechte Seite entweder gleich 1 oder nicht näher zu 1 als $3/4$ oder $4/3$. Auf der anderen Seite ist die linke Seite nicht 1 und liegt zwischen $3/4$ und $4/3$. Daher kann die Gleichung nicht gelten und es folgt $c_{i'_0,j'_0} \neq 0$.

$\square$

Definition 3.8 *Für eine Kante (v, w) mit Marke $(\boldsymbol{f}, \boldsymbol{g})$ schreiben wir $\boldsymbol{f} = \boldsymbol{f}^{v,w}$ und $\boldsymbol{g} = \boldsymbol{g}^{v,w}$. Wir nennen $\boldsymbol{f}^{v,w}$ die* f-Farbe *und $\boldsymbol{g}^{v,w}$ die* g-Farbe *von (v, w).*

Für einen Knoten $v \in V$ unterscheiden wir im Beweis von Lemma 3.6 zwischen zwei Zuständen: Entweder sind die zu v inzidenten Kanten überwiegend von einer Farbe, oder v wird von keiner Farbe dominiert. (Die genauen Definitionen *f-dominiert, g-dominiert, bunt* folgen unten.) Da G bipartit ist, ergeben sich daraus im wesentlichen 4 Möglichkeiten: Jede der beiden Knotenmengen, V_1 und V_2, kann entweder größtenteils dominierte oder größtenteils bunte Knoten enthalten.

Das folgende Lemma erlaubt es uns, im Fall, daß V_1 und V_2 größtenteils bunte Knoten enthalten, die Anzahl der 6er-Zykel abzuschätzen. Wir erhalten daraus dann eine obere Schranke für $|V|$.

Lemma 3.10 *G besitzt höchstens mit Wahrscheinlichkeit $4\binom{m}{k}^{12}/rs$ (über eine zufällige Wahl von $\boldsymbol{d}$) die folgende Eigenschaft nicht:*

Für alle $v, w \in V$ gibt es höchstens einen Pfad (v, a, b, w) mit $\boldsymbol{f}^{v,a} \neq \boldsymbol{f}^{a,b} \neq \boldsymbol{f}^{b,w}$ und $\boldsymbol{g}^{v,a} \neq \boldsymbol{g}^{a,b} \neq \boldsymbol{g}^{b,w}$.

Beweis. Wir nehmen an, es gibt zwei solche Pfade (v, a, b, w) und (v, a', b', w) mit den Marken $(\boldsymbol{f}^1, \boldsymbol{g}^1), \ldots, (\boldsymbol{f}^3, \boldsymbol{g}^3)$ bzw. $(\boldsymbol{f}^4, \boldsymbol{g}^4), \ldots, (\boldsymbol{f}^6, \boldsymbol{g}^6)$. Lemma 3.8 liefert

$$\sum_{l=1}^{6} (-1)^l d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = 0 \bmod rs \ .$$

Andererseits gilt nach Lemma 3.9 mit Wahrscheinlichkeit mindestens $1 - 4\binom{m}{k}^{12}/rs$, daß jede solche Gleichung mit $\boldsymbol{f}^1, \boldsymbol{g}^1, \ldots, \boldsymbol{f}^6, \boldsymbol{g}^6 \in \{0,1\}^m$ und $\mathrm{wt}(\boldsymbol{f}^1) = \cdots = \mathrm{wt}(\boldsymbol{g}^6) = k$ die Gleichung

$$\sum_{l=1}^{3}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l = \sum_{l=4}^{6}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l \tag{3.12}$$

impliziert.

Es reicht also aus, diese Gleichung zum Widerspruch zu führen. Angenommen, (3.12) gilt, d.h. die Summe $\sum_{l=1}^{3}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l$ hängt nur von v und w ab. Das folgende Fakt zeigt, daß $\boldsymbol{g}^2$ durch diese Summe eindeutig bestimmt ist, d.h. daß $\boldsymbol{g}^2 = \boldsymbol{g}^5$ gilt. Analog dazu sieht man $\boldsymbol{f}^2 = \boldsymbol{f}^5$, und da jedes Paar $(\boldsymbol{f}, \boldsymbol{g})$ höchstens eine Kante markiert, folgt $a = a'$ und $b = b'$. Damit ist das Lemma dann bewiesen.

Fakt 3.11 *Falls $\boldsymbol{f}^1 \neq \boldsymbol{f}^2 \neq \boldsymbol{f}^3$ gilt, ist $\boldsymbol{g}^2$ durch die Matrix $\mathbf{A} := \sum_{l=1}^{3}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l$ eindeutig bestimmt.*

Beweis. Falls es ein i_0 mit $\sum_{j=1}^{m} A_{i_0 j} = k$ gibt (dies ist äquivalent zu $f^1_{i_0} = f^3_{i_0} = 0$ und $f^2_{i_0} = 1$), so folgt $\boldsymbol{g}^2 = \boldsymbol{A}_{i_0}$ (die i_0-te Zeile von $\mathbf{A}$).

Sei $\sum_{j=1}^{m} A_{ij} \neq k$ für alle i. Wegen $\boldsymbol{f}^2 \neq \boldsymbol{f}^3$ gibt es ein i_1 mit $f^2_{i_1} = 1$ und $f^3_{i_1} = 0$. Aus $\sum_{j=1}^{m} A_{i_1 j} \neq k$ erhalten wir $f^1_{i_1} = 1$. Analog dazu sehen wir, daß es ein i_2 mit $f^2_{i_2} = f^3_{i_2} = 1$ und $f^1_{i_2} = 0$ gibt. Die beiden Zeilen $\boldsymbol{A}_{i_1}$ und $\boldsymbol{A}_{i_2}$ sind (als Vektoren) bis auf Vertauschung durch die Bedingung $\sum_{j=1}^{m} A_{i_l j} = 0$ für $l = 1, 2$ eindeutig bestimmt. Wir unterscheiden nun zwei Fälle.

Falls es ein i_3 mit $\sum_{j=1}^{m} A_{i_3 j} = -2k$ (dies ist äquivalent zu $f^1_{i_l} = f^3_{i_l} = 1$ and $f^2_{i_l} = 0$) gibt, so ergibt sich $\boldsymbol{g}^2$ durch $2\boldsymbol{g}^2 = \boldsymbol{A}_{i_2} - \boldsymbol{A}_{i_3} + \boldsymbol{A}_{i_1}$.

Falls auf der anderen Seite $\sum_{j=1}^{m} A_{ij} \neq -2k$ für alle i gilt, so gibt es wegen $\boldsymbol{f}^1 \neq \boldsymbol{f}^2$ ein i_3 mit $f^1_{i_3} = 1$ und $f^2_{i_3} = 0$. Aus $\sum_{j=1}^{m} A_{i_3 j} \neq -2k$ folgt darüber hinaus $f^3_{i_3} = 0$. Analog dazu sehen wir, daß es ein i_4 mit $f^1_{i_3} = f^2_{i_3} = 0$ und $f^3_{i_3} = 1$ gibt. Die beiden Zeilen $\boldsymbol{A}_{i_3}$ und $\boldsymbol{A}_{i_4}$ sind (als Vektoren) bis auf Vertauschung durch die Bedingung $\sum_{j=1}^{m} A_{i_l j} = -k$ für $l = 3, 4$ eindeutig bestimmt. Nun ergibt sich $\boldsymbol{g}^2$ aus $2\boldsymbol{g}^2 = \boldsymbol{A}_{i_1} + \boldsymbol{A}_{i_2} - \boldsymbol{A}_{i_3} - \boldsymbol{A}_{i_4}$. Da $(\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2})$ und $(\boldsymbol{A}_{i_3}, \boldsymbol{A}_{i_4})$ jeweils bis auf Vertauschung eindeutig bestimmt sind, ist es damit auch $\boldsymbol{g}^2$.

□

Definition 3.9 *Wir nennen $v \in V$* f-monochrom, *falls alle zu v inzidenten Kanten dieselbe f-Farbe $\boldsymbol{f}$ besitzen. $\boldsymbol{f}$ nennen wir dann die* f-Farbe von v. *Eine Knotenmenge $\tilde{V} \subseteq V$ heißt* f-sortiert, *falls alle Knoten $v \in \tilde{V}$ f-monochrom sind. Für jede f-Farbe $\boldsymbol{f}$ und $v \in V$ definieren wir dann $\tilde{V}^{\mathrm{f}}(\boldsymbol{f})$ als die Menge von Knoten in $\tilde{V}$, deren Kanten die f-Farbe $\boldsymbol{f}$ haben. Analog dazu definieren wir* g-monochrom, *die* g-Farbe eines Knoten v, g-sortiert *und $\tilde{V}^{\mathrm{g}}(\boldsymbol{g})$ für eine g-Farbe $\boldsymbol{g}$.*

Im Fall, daß eine Knotenmenge (z.B. V_1) überwiegend bunte Knoten enthält, während die Knoten der zweiten Knotenmenge größtenteils von einer f-Farbe dominiert sind, werden wir die Anzahl der 8er-Zykel abschätzen. Dazu verwenden wir das folgende Lemma.

Lemma 3.12 *G besitzt höchstens mit Wahrscheinlichkeit $4\binom{m}{k}^{12}/rs$ (über eine zufällige Wahl von $\boldsymbol{d}$) die folgende Eigenschaft nicht:*

Für alle f-monochromen $v, w \in V_1$ ist die Anzahl der 4-Pfade (v, a, b, c, w) mit $a \neq c$ und $\boldsymbol{f}^{v,a} \neq \boldsymbol{f}^{a,b} = \boldsymbol{f}^{b,c} \neq \boldsymbol{f}^{c,w}$ beschränkt durch

1. $\mathrm{dg}(v)M$ *falls v und w dieselbe f-Farbe haben,*

2. $2M$ *falls v und w verschiedene f-Farben haben,*

wobei $\mathrm{dg}(v)$ *der Grad von v und* $M := \max_{\boldsymbol{f}}(|V_1^{\mathrm{f}}(\boldsymbol{f})|)$ *ist.*

Beweis. Nach Lemma 3.8 erhalten wir für je zwei solcher 4-Pfade (v, a, b, c, w) und (v, a', b', c', w) eine Gleichung

$$\sum_{l=1}^{4}(-1)^l d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) = \sum_{l=5}^{8}(-1)^l d_{\boldsymbol{d}}(\boldsymbol{f}^l, \boldsymbol{g}^l) \bmod rs$$

in den Marken der Kanten. Dabei gilt $\boldsymbol{f}^1 \neq \boldsymbol{f}^2 = \boldsymbol{f}^3 \neq \boldsymbol{f}^4$ und $\boldsymbol{f}^5 \neq \boldsymbol{f}^6 = \boldsymbol{f}^7 \neq \boldsymbol{f}^8$. Da v, w f-monochrom sind, gilt weiterhin $\boldsymbol{f}^1 = \boldsymbol{f}^5$ und $\boldsymbol{f}^4 = \boldsymbol{f}^8$ (siehe Figur 3.1).

Wir nehmen nun an, für jedes Tupel $(\boldsymbol{f}^1, \boldsymbol{g}^1, \ldots, \boldsymbol{f}^8, \boldsymbol{g}^8)$ mit $\boldsymbol{f}^1, \ldots, \boldsymbol{g}^8 \in \{0,1\}^m$, $\mathrm{wt}(\boldsymbol{f}^1) = \cdots = \mathrm{wt}(\boldsymbol{g}^8) = k$ und mit $\boldsymbol{f}^2 = \boldsymbol{f}^3$, $\boldsymbol{f}^6 = \boldsymbol{f}^7$, $\boldsymbol{f}^1 = \boldsymbol{f}^5$ und $\boldsymbol{f}^4 = \boldsymbol{f}^8$, gilt

$$\sum_{l=1}^{4}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l = \sum_{l=5}^{8}(-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l \quad . \tag{3.13}$$

Da die Anzahl dieser Tupel $(\boldsymbol{f}^1, \boldsymbol{g}^1, \ldots, \boldsymbol{f}^8, \boldsymbol{g}^8)$ durch $\binom{m}{k}^{12}$ beschränkt ist, gilt das nach Lemma 3.9 mit Wahrscheinlichkeit mindestens $1 - 4\binom{m}{k}^{12}/rs$.

Wir zeigen zunächst, daß es für festes a höchstens M solcher Pfade (v, a, b, c, w) gibt. Die erste Schranke folgt dann sofort. Danach zeigen wir, daß es höchstens zwei Möglichkeiten für a gibt, falls v und w verschiedene f-Farben haben. Das beweist die zweite Schranke.

Angenommen, es gilt $a = a'$ und damit $\boldsymbol{g}^1 = \boldsymbol{g}^5$. Gleichung (3.13) vereinfacht sich nun zu

$$-\boldsymbol{f}^3 \otimes \boldsymbol{g}^3 + \boldsymbol{f}^4 \otimes \boldsymbol{g}^4 - \boldsymbol{f}^6 \otimes \boldsymbol{g}^6 = -\boldsymbol{f}^7 \otimes \boldsymbol{g}^7 + \boldsymbol{f}^8 \otimes \boldsymbol{g}^8 - \boldsymbol{f}^2 \otimes \boldsymbol{g}^2.$$

Falls $\boldsymbol{f}^3 \neq \boldsymbol{f}^6$ gilt, sind die Vektoren $\boldsymbol{f}^3, \boldsymbol{f}^4, \boldsymbol{f}^6$ alle verschieden. Fakt 3.11 liefert $\boldsymbol{g}^8 = \boldsymbol{g}^4$. Zusammen mit $\boldsymbol{f}^8 = \boldsymbol{f}^4$ erhalten wir $\boldsymbol{f}^2 \otimes \boldsymbol{g}^2 + \boldsymbol{f}^3 \otimes \boldsymbol{g}^3 = \boldsymbol{f}^6 \otimes \boldsymbol{g}^6 + \boldsymbol{f}^7 \otimes \boldsymbol{g}^7$. Wegen $a \neq c$ ist dies

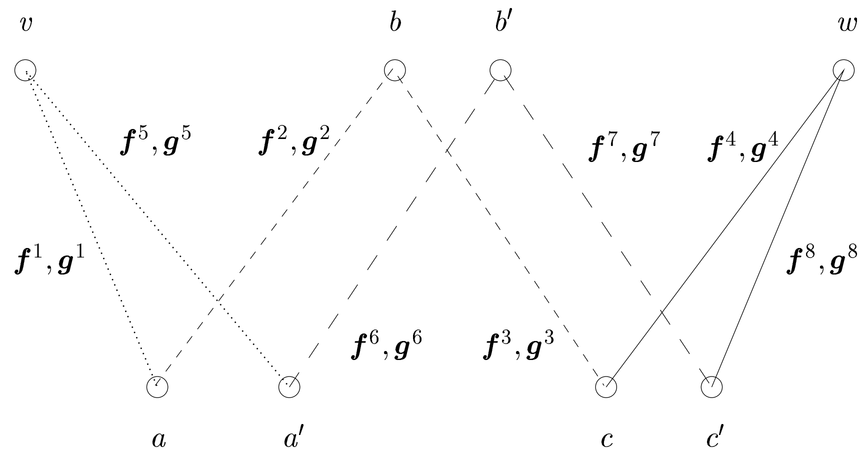

Abbildung 3.1: Zwei 4–Pfade von v nach w. (*Der Linienstil bezieht sich auf die f-Farben*). Der resultierende Zykel behält seine Farbe, wenn er einen Knoten in V_1 (*obere Knoten*) passiert und wechselt sie beim Passieren eines Knoten in V_2 (*untere Knoten*).

ungleich 0. Da $\boldsymbol{f}^2 = \boldsymbol{f}^3$ und $\boldsymbol{f}^6 = \boldsymbol{f}^7$ gilt, kann diese Gleichung für $\boldsymbol{f}^2 \neq \boldsymbol{f}^6$ offensichtlich nicht gelten. Wir erhalten also $\boldsymbol{f}^2 = \boldsymbol{f}^6$, d.h. $\boldsymbol{f}^2$ ist durch a eindeutig bestimmt. Damit gibt es höchstens $|V_1^{\mathrm{f}}(\boldsymbol{f}^3)| \leq M$ viele Möglichkeiten für b.

Falls auf der anderen Seite $b = b'$ gilt, so vereinfacht sich (3.13) zu $\boldsymbol{f}^4 \otimes \boldsymbol{g}^4 - \boldsymbol{f}^8 \otimes \boldsymbol{g}^8 = \boldsymbol{f}^3 \otimes \boldsymbol{g}^3 - \boldsymbol{f}^7 \otimes \boldsymbol{g}^7$. Wegen $\boldsymbol{f}^4 = \boldsymbol{f}^8 \neq \boldsymbol{f}^3 = \boldsymbol{f}^7$ kann das offensichtlich nur für $\boldsymbol{g}^3 = \boldsymbol{g}^7$ und $\boldsymbol{g}^4 = \boldsymbol{g}^8$ – d.h. für $c = c'$ – gelten. Somit bestimmen a und b den Pfad bereits eindeutig, und für feste a gibt es höchstens M Pfade.

Es bleibt nun nur noch zu zeigen, daß falls v und w verschiedene f-Farben haben (i.e. $\boldsymbol{f}^1 \neq \boldsymbol{f}^4$), es höchstens zwei Möglichkeiten für $\boldsymbol{g}^1$ gibt. Da a durch $\boldsymbol{g}^1$ bereits eindeutig bestimmt ist ($\boldsymbol{f}^1$ ist durch v festgelegt), gibt es damit ebenfalls höchstens zwei Möglichkeiten für a.

Sei $\boldsymbol{f}^1 \neq \boldsymbol{f}^4$. Wir setzen $\mathbf{A} := \sum_{l=1}^{4} (-1)^l \boldsymbol{f}^l \otimes \boldsymbol{g}^l$. Falls $\sum_j A_{ij} = -k$ für ein i gilt, so folgt $f_i^1 = 1$ und $f_i^4 = 0$. Wegen $\boldsymbol{f}^2 = \boldsymbol{f}^3$ gibt es somit höchstens zwei (als Vektoren) verschiedene Zeilen $\boldsymbol{A}_{i_1}$ und $\boldsymbol{A}_{i_2}$ mit $\sum_j A_{i_l j} = -k$ für $l = 1, 2$. Damit folgt $\boldsymbol{g}^1 = \boldsymbol{A}_{i_1}$ oder $\boldsymbol{g}^1 = \boldsymbol{A}_{i_2}$.

□

Das vorangegangene Lemma lieferte uns eine Schranke für die Anzahl bestimmter 4–Pfade in G. Wir haben dabei jedoch nur nicht–entartete Pfade, d.h. Pfade mit $a \neq c$ betrachtet. Das folgende Lemma hilft uns, im Beweis von Satz 3.6 auch die Anzahl der entarteten Pfade abzuschätzen.

Definition 3.10 *Für einen Knoten v sei $\mathcal{N}(v)$ die Menge der zu v adjazenten Knoten. Für eine f-Farbe $\boldsymbol{f}$ bezeichnen wir mit $\mathrm{dg}_{\boldsymbol{f}}(v)$ die Anzahl der zu v inzidenten Kanten mit f-Farbe $\boldsymbol{f}$. Analog dazu definieren wir $\mathrm{dg}_{\boldsymbol{g}}(v)$ für eine g-Farbe $\boldsymbol{g}$.*

Wir nennen einen Knoten v f-dominiert, *falls $\mathrm{dg}_{\boldsymbol{f}}(v) \geq \frac{3}{4}\mathrm{dg}(v)$ für eine f-Farbe $\boldsymbol{f}$ gilt. Analog dazu definieren wir* g-dominiert. *Falls v weder f-dominiert noch g-dominiert ist, so nennen wir ihn* bunt.

Lemma 3.13 *Sei $G' \subseteq G$ mit $|V_1'|, |V_2'| \leq K$ für ein $K \in \mathbb{N}$. Dann enthält G' einen Untergraphen $\tilde{G} = (\tilde{V}_1, \tilde{V}_2, \tilde{E})$ mit $|\tilde{E}| > |E'| - 3K \log_2 K$ so, daß für alle $v \in \tilde{V}_1$*

$$\max_{w \in \mathcal{M}(v)} (\mathrm{dg}(w) - \mathrm{dg}_{\boldsymbol{f}^{v,w}}(w)) \leq \frac{1}{2} \sum_{w \in \mathcal{M}(v)} \mathrm{dg}(w) - \mathrm{dg}_{\boldsymbol{f}^{v,w}}(w) \ , \tag{3.14}$$

wobei $\mathcal{M}(v)$ die Menge der zu v adjazenten Knoten ist, die nicht f-dominiert sind.

Beweis. Für jedes $v \in V_1$, welches (3.14) nicht erfüllt, entfernen wir die Kante (v, w_v) mit $w_v \in \mathcal{M}(v)$, für die $\mathrm{dg}(w) - \mathrm{dg}_{\boldsymbol{f}^{v,w}}(w)$ maximal ist. Wir wiederholen diese Prozedur solange, bis der verbleibende Graph die gewünschte Bedingung erfüllt.

Für jede Kante (w_v, v), die wir bei dieser Verjüngung entfernen, verringert sich die Funktion $D(v) := \sum_{w \in \mathcal{M}(v)} \mathrm{dg}(w) - \mathrm{dg}_{\boldsymbol{f}^{v,w}}(w)$ mindestens um den Faktor 2. Wegen

$$D(v) \ \leq \ \sum_{v \in V_1} D(v) \ \leq \ \sum_{v \in V_1} \sum_{w \in \mathcal{N}(v)} \mathrm{dg}(w) \ = \ \sum_{w \in V_2} \mathrm{dg}(w)^2 \ \leq \ K^3$$

werden bei der Verjüngung an jedem Knoten $v \in V_1$ höchstens $3 \log_2 K$ viele Kanten entfernt. Insgesamt werden also höchstens $3K \log_2 K$ viele Kanten entfernt.

□

Definition 3.11 *Für $U_1, U_2 \subset V$ setzen wir $E(U_1, U_2) := \{(v, w) \in E \mid v \in U_1, \, w \in U_2\}$.*

Das folgende Lemma stellt sicher, daß ein ausreichend großer Anteil der Pfade in G beim Passieren eines bunten Knotens seine Färbung wechselt.

Lemma 3.14 *Sei $v \in V$ bunt, und für $w \in \mathcal{N}(v)$ sei $A_v(w)$ die Menge von Kanten $(v, z) \neq (v, w)$ mit $\boldsymbol{f}^{v,z} = \boldsymbol{f}^{v,w}$ oder $\boldsymbol{g}^{v,z} = \boldsymbol{g}^{v,w}$. Dann gibt es höchstens ein $w \in N(v)$ mit $|A_v(w)| > \frac{7}{8}\,\mathrm{dg}(v)$.*

Beweis. Wir zeigen, daß $|A_v(w) \cap A_v(w')| \leq \frac{3}{4}\,\mathrm{dg}(v)$ für verschiedene $w, w' \in \mathcal{N}(v)$ gilt. Die Behauptung folgt daraus unmittelbar.

Sei $w \neq w' \in \mathcal{N}(v)$. Dann gilt entweder $\boldsymbol{f}^{v,w} \neq \boldsymbol{f}^{v,w'}$ oder $\boldsymbol{g}^{v,w} \neq \boldsymbol{g}^{v,w'}$. O.B.d.A. können wir annehmen, daß $\boldsymbol{f}^{v,w} \neq \boldsymbol{f}^{v,w'}$ gilt. Es ergibt sich damit $|A_v(w) \cap A_v(w')| = \mathrm{dg}_{\boldsymbol{g}^{v,w}}(v)$ für

$\boldsymbol{g}^{v,w} = \boldsymbol{g}^{v,w'}$ und $A_v(w) \cap A_v(w') = \emptyset$ sonst. Da v bunt ist, erhalten wir $|A_v(w) \cap A_v(w')| \leq \frac{3}{4}\,\mathrm{dg}(v)$.

□

Wir können nun Satz 3.6 beweisen.

Beweis von Satz 3.6. Sei $2^{40} \leq \binom{m}{k}$. Wir setzen $V_{\mathrm{c}} := \{v \in V \mid v \text{ ist bunt}\}$, $V_{\mathrm{f}} := \{v \in V \mid v \text{ ist f-dominiert}\}$ und $V_{\mathrm{g}} := \{v \in V \mid v \text{ ist g-dominiert}\}$.

Wir skizzieren kurz den Beweis: Falls E „viele“ Kanten (im Verhältnis zur Anzahl der Knoten) enthält, so enthält auch eine der Kantenmengen $E(V_{\mathrm{c}}, V_{\mathrm{c}})$, $E(V_{\mathrm{f}}, V_2)$, $E(V_1, V_{\mathrm{f}})$, $E(V_{\mathrm{g}}, V_2)$, $E(V_1, V_{\mathrm{g}})$ „viele“ Kanten. Im ersten Fall wenden wir Lemma 3.10 an und erhalten eine obere Schranke für die Anzahl der Kanten – ein Widerspruch zur Annahme. In den anderen Fällen führen wir eine weitere Fallunterscheidung durch: Enthält als Beispiel $E(V_{\mathrm{f}}, V_2)$ „viele“ Kanten, so gilt das auch für $E(V_{\mathrm{f}}, V_{\mathrm{f}})$ oder für $E(V_{\mathrm{f}}, V_2 \setminus V_{\mathrm{f}})$. Im ersten Fall besitzt G einen „großen“ f-sortierten Untergraphen $\tilde{G} = (\tilde{V}, \tilde{E})$. Für diesen erhalten wir aus $|\tilde{E}| \leq \sum_{\boldsymbol{f}} |\tilde{V}_1^{\mathrm{f}}(\boldsymbol{f})| \cdot |\tilde{V}_2^{\mathrm{f}}(\boldsymbol{f})|$ eine obere Schranke für die Anzahl der Kanten. Im zweiten Fall erhalten wir eine solche Schranke aus den Lemmata 3.12 und 3.13.

Wir nehmen an, daß G die in den Lemmata 3.10 und 3.12 beschriebenen Eigenschaften besitzt. Das gilt mit Wahrscheinlichkeit mindestens $1 - 8\binom{m}{k}^{12}/rs$. Wir zeigen nun, daß in diesem Fall $|V| > 2^{-4.75}|E|\binom{m}{k}^{-1/2}$ gilt.

In den folgenden Abschätzungen verwenden wir Konstanten ϵ_i und α_i, welche wir erst am Ende des Beweises festlegen werden.

Angenommen, es gilt $|V| < 2^{-4.75}|E|\binom{m}{k}^{-1/2}$. Wir unterscheiden nun fünf Fälle.

Falls $|E(V_{\mathrm{f}}, V_2)| > \epsilon_1 |E|$ gilt, so enthält G einen Untergraphen $G' = (V_1', V_2', E')$, so daß V_1' f-sortiert ist und $|E'| > \frac{3}{4}\epsilon_1 |E|$ gilt. Mit Lemma 3.13 finden wir einen Untergraphen $\hat{G}$ von G' so, daß (3.14) für alle $v \in \hat{V}_1$ und $|\hat{E}| > |E'| - 3|V|\log|V| = \alpha_1 |E|$ gilt. Wir setzen nun $U_2 := \{w \in \hat{V}_2 \mid w \text{ ist f-dominiert}\}$. Es gibt nun zwei Möglichkeiten:

1. Falls $\hat{E}(\hat{V}_1, U_2) > \epsilon_3 |\hat{E}|$ gilt, enthält $\hat{G}$ einen f-sortierten Untergraphen $\tilde{G} = (\tilde{V}_1, \tilde{V}_2, \tilde{E})$, so daß $|\tilde{E}| > \frac{3}{4}\epsilon_2 |\hat{E}|$ gilt. Wegen $|x| + |y| \geq 2\sqrt{xy}$ für alle $x, y \in \mathbb{R}^t$ ist die Anzahl der Knoten $|\tilde{V}_1| + |\tilde{V}_2|$ mindestens

$$\sum_{\boldsymbol{f}} |\tilde{V}_1^{\mathrm{f}}(\boldsymbol{f})| + |\tilde{V}_2^{\mathrm{f}}(\boldsymbol{f})| \geq 2 \sum_{\boldsymbol{f}} \sqrt{|\tilde{V}_1^{\mathrm{f}}(\boldsymbol{f})| \cdot |\tilde{V}_2^{\mathrm{f}}(\boldsymbol{f})|} \ .$$

Da $\tilde{G}$ f-sortiert ist, ist das mindestens $2\sum_{\boldsymbol{f}}\sqrt{|\tilde{E}^{\mathrm{f}}(\boldsymbol{f})|}$, wobei $\tilde{E}^{\mathrm{f}}(\boldsymbol{f})$ die Menge der Kanten in $\tilde{E}$ mit f-Farbe $\boldsymbol{f}$ ist. Da jedes Paar $(\boldsymbol{f},\boldsymbol{g}) \in S'_{\boldsymbol{d}}$ höchstens eine Kante markiert, gilt außerdem

$$|\tilde{E}^{\mathrm{f}}(\boldsymbol{f})| \leq |\{(\boldsymbol{f}',\boldsymbol{g}') \in S'_{\boldsymbol{d}} \mid \boldsymbol{f} = \boldsymbol{f}'\}| \leq \binom{m}{k} \ .$$

Wir erhalten damit $|\tilde{V}_1| + |\tilde{V}_2| \geq 2\binom{m}{k}^{-1/2}\sum_{\boldsymbol{f}}|\tilde{E}^{\mathrm{f}}(\boldsymbol{f})|$.

Andererseits gilt $|\tilde{E}| \leq \sum_{\boldsymbol{f}}|\tilde{V}_1^{\mathrm{f}}(\boldsymbol{f})|\cdot|\tilde{V}_2^{\mathrm{f}}(\boldsymbol{f})|$, und wir erhalten $|V| > \frac{3}{4}\epsilon_2\alpha_1|E|\binom{m}{k}^{-1/2}$.

2. Falls $|\hat{E}(\hat{V}_1,\hat{V}_2\setminus U_2)| > (1-\epsilon_2)|\hat{E}|$ gilt, betrachten wir den Graphen $\tilde{G} = (\hat{V}_1,\hat{V}_2\setminus U_2,\hat{E}(\hat{V}_1,\hat{V}_2\setminus U_2))$. Man beachte, daß (3.14) für alle Knoten $v \in \hat{V}_1$ auch in $\tilde{G}$ gilt. Da $\tilde{V}_1$ f-sortiert ist und $|\tilde{V}_1^{\mathrm{f}}(\boldsymbol{f})| \leq \binom{m}{k}^{1/2}$ für alle $\boldsymbol{f}$ gilt, folgt mit Lemma 3.12, daß die Anzahl der 4-Pfade (v,a,b,c,w) in $\tilde{G}$ mit $v,w \in \tilde{V}_1$, $\boldsymbol{f}^{v,a} \neq \boldsymbol{f}^{a,b}$ und $\boldsymbol{f}^{b,c} \neq \boldsymbol{f}^{c,w}$ durch

$$\sum_{w\in\tilde{V}_1}\left(2\binom{m}{k}^{1/2}|V| + \mathrm{dg}(w)\binom{m}{k}^{1/2}\right) \leq 4|V|^2\binom{m}{k}^{1/2}$$

beschränkt ist.

Da $\tilde{V}_1$ f-sortiert ist, ist auf der anderen Seite die Anzahl dieser 4-Pfade mindestens

$$\sum_{(v,a)\in\tilde{E}}\ \sum_{\substack{b\in\mathcal{N}(a)\\ \boldsymbol{f}^{v,a}\neq\boldsymbol{f}^{a,b}}}\ \sum_{c\in\mathcal{N}(b)\setminus\{a\}} \mathrm{dg}(c) - \mathrm{dg}_{\boldsymbol{f}^{b,c}}(c).$$

Mit (3.14) können wir dieses durch

$$\begin{aligned} & 1/2\sum_{(v,a)\in\tilde{E}}\ \sum_{\substack{b\in\mathcal{N}(a)\\ \boldsymbol{f}^{v,a}\neq\boldsymbol{f}^{a,b}}}\ \sum_{c\in\mathcal{N}(b)} \mathrm{dg}(c) - \mathrm{dg}_{\boldsymbol{f}^{b,c}}(c) \\ = \ & 1/2\sum_{b\in\tilde{V}_1}\Big(\sum_{c\in\mathcal{N}(b)}\mathrm{dg}(c) - \mathrm{dg}_{\boldsymbol{f}^{b,c}}(c)\Big)^2 \end{aligned}$$

abschätzen. Da alle $c \in \tilde{V}_2$ nicht f-dominiert sind, folgt mit der Cauchy-Schwarz'schen Ungleichung, daß dies durch

$$\begin{aligned} 2^{-5}|\tilde{V}|^{-1}\Big(\sum_{(b,c)\in\tilde{E}}\mathrm{dg}(c)\Big)^2 &= 2^{-5}|\tilde{V}|^{-1}\Big(\sum_{c\in\tilde{V}_2}\mathrm{dg}(c)^2\Big)^2 \\ &\leq 2^{-5}|\tilde{V}|^{-3}|\tilde{E}|^4 \end{aligned}$$

beschränkt ist. Wir erhalten somit $|V| > 2^{-7/5}(1-\epsilon_2)^{4/5}\alpha_1^{4/5}|E|\binom{m}{k}^{-1/2}$.

Falls $|E(V_{\mathrm{g}},V_2)| \geq \epsilon_1|E|$, $|E(V_1,V_{\mathrm{f}}) \geq \epsilon_1|E|$ *oder* $|E(V_1,V_{\mathrm{g}})| \geq \epsilon_1|E|$ *gilt,* erhalten wir die gleichen Abschätzungen auf analoge Weise.

Falls $|E(V_c, V_c)| \geq (1-4\epsilon_1)|E|$ *gilt,* definieren wir E' als die Menge der bunten Kanten (v,w), für die $|A_v(w)| \leq \frac{7}{8}\,\mathrm{dg}(v)$ und $|A_w(v)| \leq \frac{7}{8}\,\mathrm{dg}(w)$ gilt. Mit Lemma 3.14 erhalten wir $|E'| > (1-4\epsilon_1)|E| - 2|V|$.

Aus Lemma 3.10 folgt, daß die Anzahl der 3-Pfade (u,v,w,z) in G' mit $u \in V_2'$, $\boldsymbol{f}^{u,v} \neq \boldsymbol{f}^{v,w} \neq \boldsymbol{f}^{w,z}$ und $\boldsymbol{g}^{u,v} \neq \boldsymbol{g}^{v,w} \neq \boldsymbol{g}^{w,z}$ durch $|V|^2$ beschränkt ist.

Auf der anderen Seite ist die Anzahl dieser Wege mindestens

$$\sum_{(v,w)\in E'} \Big(\mathrm{dg}(v) - |A_v(w)|\Big)\Big(\mathrm{dg}(w) - |A_w(v)|\Big) \ ,$$

was wiederum größer als $2^{-6}\sum_{(v,w)\in E'} \mathrm{dg}(v)\mathrm{dg}(w)$ ist. Mit den Identitäten

$$\sum_{(v,w)\in E} \mathrm{dg}(v)^{-1} = \sum_{(v,w)\in E} \mathrm{dg}(w)^{-1} = |V|$$

können wir dies durch $2^{-6}\min\big(\sum x_i y_i \;\big|\; \sum x_i^{-1} + y_i^{-1} \leq 2|V|\big)$ abschätzen, wobei das Minimum über alle $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{|E'|} \setminus \{\mathbf{0}\}$ genommen wird. Das Minimum tritt auf, wenn alle x_i und y_i gleich sind. Mit $|E'| \geq (1-4\epsilon_1)|E| - 2|V| = \alpha_2|E|$ erhalten wir somit eine untere Schranke von $2^{-6}\alpha_2^3|E|^3|V|^{-2}$ für die Anzahl der betrachteten Wege. Zusammen mit der oberen Schranke ergibt sich nun $|V| > 2^{-3/2}\alpha_2^{3/4}|E|\binom{m}{k}^{-1/2}$.

Wir fixieren nun die Konstanten zu $\epsilon_1 = 0.2$, $\epsilon_2 = 0.5$. Wegen $\binom{m}{k} \geq 2^{40}$ und $|V| < 2^{-4.75}|E|\binom{m}{k}^{-1/2} \leq |E|^{3/4}$ können wir $|E| \geq 2^{24}$ annehmen und erhalten damit $\alpha_1 \geq 0.11$ und $\alpha_2 \geq 0.2$. Schließlich ergibt sich $|V| > 2^{-4.75}|E|\binom{m}{k}^{-1/2}$. Das ist ein Widerspruch zur Annahme.

□

3.3 Andere Protokolle

Unsere Resultate lassen sich auch auf andere Server-unterstützte RSA Protokolle übertragen. Wir betrachten zunächst das Protokoll RSA-S2 ([MKI89]). Die Parameter des Protokolls sind wieder m, k und ℓ.

Das RSA-S2 Protokoll. Vor der ersten Ausführung des Protokolls wählt der Klient zwei zufällige Vektoren $\boldsymbol{f}, \boldsymbol{g} \in [0, 2^\ell)^m$ mit Hamminggewicht k, so daß $\mathrm{ggT}(f_i, p-1) = 1$ für mindestens ein i und $\mathrm{ggT}(g_j, q-1) = 1$ für mindestens ein j mit $f_j = 0$ gilt. Dann wählt er einen Vektor $\boldsymbol{d} \in [0, \max(p,q)-1)^m$ zufällig so, daß $d = \sum_{i=1}^m f_i d_i \bmod p-1$ und $d = \sum_{i=1}^m g_i d_i \bmod q-1$ gilt. Wenn der Klient eine Unterschrift erzeugen will, tritt er mit einem Server in Kontakt und führt mit diesem das folgende Protokoll aus.

1. Der Klient sendet x, n und $\boldsymbol{d}$ zum Server.
2. Der Server sendet $z_i = x^{d_i} \bmod n$ für $i = 1, \dots, m$ an den Klienten.
3. Der Klient berechnet $y_p = \prod_{i=1}^m z_i^{f_i} \bmod p$ und $y_q = \prod_{i=1}^m z_i^{g_i} \bmod q$ und gibt dann als Unterschrift $y = y_p w_p + y_q w_q \bmod n$ mit $w_p := q(q^{-1} \bmod p)$ und $w_q := p(p^{-1} \bmod q)$ aus.

Die Sicherheit dieses Verfahrens beruht auf den zwei Zerlegungen des Schlüssels $d = \sum_{i=1}^m f_i d_i \bmod p-1$ und $d = \sum_{i=1}^m g_i d_i \bmod q-1$. Insgesamt gibt es $\binom{m\ell}{k}^2$ viele Möglichkeiten für $\boldsymbol{f}, \boldsymbol{g}$. Die folgende Überlegung zeigt jedoch, daß eine untere Schranke für die Sicherheit gegen generische Angreifer nur auf einer der beiden Zerlegungen basieren kann.

Die Schritte eines generischen Angreifers können in beliebiger Weise von dem öffentlichen Parameter n – und damit auch von seinen Primfaktoren p und q – abhängen. Dieser kann also mit Hilfe der Werte $x^{q-1} \bmod n$ und $y^{q-1} \bmod n$ zunächst die erste Zerlegung und dann – unabhängig davon – mit Hilfe der Werte $x^{p-1} \bmod n$ und $y^{p-1} \bmod n$ die zweite Zerlegung berechnen. Den geheimen Schlüssel d kann er dann aus $d \bmod p-1$ und $d \bmod q-1$ zusammensetzen. Die Komplexität dieses Angriffes ist damit nur doppelt so groß wie die Komplexität eines generische Algorithmus, der nur eine der beiden Zerlegungen berechnet. Immerhin erhalten wir analog zu Theorem 3.1 den folgenden Satz.

Satz 3.15 *Jeder generische Angreifer auf RSA-S2 hat mindestens Komplexität* $\frac{1}{2}\binom{m\ell}{k}^{1/2}$.

Auf der anderen Seite reicht in der Praxis bereits die Kenntnis einer Zerlegung aus, um n zu faktorisieren. Wegen $d' := \sum f_i^1 d_i = d \bmod p-1$ und $d' \neq d \bmod q-1$ (das gilt zumindest mit überwältigender Wahrscheinlichkeit) gilt nämlich $\mathrm{ggT}(x^{d'} - y, n) = p$. Béguin und Quisquater geben in [BQ95] einen Algorithmus an, welcher aus n, x, y und $\boldsymbol{d}$ in Zeit $O\left(\binom{m\ell}{k}^{1/2} \log^2 \binom{m\ell}{k}\right)$ die Faktorisierung von n und damit auch d berechnet. (Es handelt sich dabei eigentlich um einen Angriff auf das von den Autoren vorgestellte Server-unterstützte RSA Protokoll. Das Verfahren läßt sich jedoch auch direkt auf RSA-S2 anwenden.)

Das Protokoll von Béguin und Quisquater ([BQ95]) ist eine Variante von RSA-S2. Alles zu RSA-S2 Gesagte gilt auch für dieses Protokoll und wir erhalten die gleiche untere Schranke gegen generische Angreifer.

Das RSA-S2M Protokoll ist die 2-Runden Version von RSA-S2 – genau so, wie RSA-S1M die 2-Runden Version von RSA-S1 ist. Auch hier kann die Sicherheit gegen generische Angreifer nur *entweder* auf der Zerlegung modulo $p-1$ *oder* auf der Zerlegung modulo $q-1$ basieren. Für $\ell = 1$ erhalten wir analog zu Theorem 3.3 das folgende Resultat.

Theorem 3.16 *Falls* $\binom{m}{k}^{12} \ll r$ *gilt, hat mit Wahrscheinlichkeit* $1 - 9\binom{m}{k}^{12}/r$ *(über zufälliges* x *und* $\boldsymbol{d}$*) jeder Meet-in-the-middle Angreifer gegen RSA-S2M mindestens Komplexität*

$2^{-7}\binom{m}{k}^{3/2}$.

In der Praxis ist die Situation hier analog zu RSA-S2. Lim und Lee verwenden in [LL95] die Ideen von [BQ95] für einen Angriff auf RSA-S2M mit Laufzeit $O\left(\binom{m}{k}^{3/2}\log^2\binom{m}{k}\right)$. Seine generische Komplexität erreicht unsere untere Schranke.

3.4 Angriffe durch Gitterbasenreduktion

In den vorangegangenen Abschnitten haben wir uns ausschließlich mit passiven Angriffen gegen Server-unterstützte RSA Protokolle beschäftigt. Gegen RSA-S1 und RSA-S2 wurden jedoch auch zahlreiche aktive Angriffe veröffentlicht ([And92], [PW93], [BM94], [HCY95], [Hor98]). Selbst wenn der Klient am Ende des Protokolls die Signatur prüft, ist es dem Server möglich, partielle Information über die geheime Zerlegung des Schlüssels zu erlangen (siehe [Hor98]). Es wurde daher vorgeschlagen, diese Zerlegung nach einer festen Anzahl von Ausführungen des Protokolls zu erneuern (siehe [LL95]).[23] Wir zeigen nun, daß diese periodische Erneuerung Angriffe durch Gitterbasenreduktion ermöglicht.

3.4.1 Der Fall von RSA-S1

Wir nehmen an, der Client erneuere die Zerlegung seines geheimen Schlüssels d von Zeit zu Zeit. Er wählt sich dazu jeweils ein zufälliges $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k und $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i und dann ein zufälliges $\boldsymbol{d} \in [0, \phi(n))^m$ so, daß $\sum_{i=1}^m f_i d_i = d \bmod \phi(n)$. Dann gilt die Gleichung

$$\sum_{i=1}^m f_i d_i = \sum_{i=1}^m f_i' d_i' \tag{3.15}$$

modulo $\phi(n)$ für alle Zerlegungen $(\boldsymbol{f}, \boldsymbol{d})$ und $(\boldsymbol{f}', \boldsymbol{d}')$. Die folgende Überlegung zeigt, daß die Gleichheit mindestens mit Wahrscheinlichkeit $1/(k2^\ell)$ auch über $\mathbb{Z}$ gilt.

Ohne Einschränkung können wir annehmen, daß d der Repräsentant von $d \bmod \phi(n)$ in $[0, \phi(n))$ ist. Wegen $\sum_{i=1}^m f_i d_i \leq k2^\ell \phi(n)$ gilt

$$\sum_{i=1}^m f_i d_i = d + j\phi(n) \tag{3.16}$$

[23] Eine andere Möglichkeit, diese Angriffe abzuwehren, ist die Verifikation der Signatur und die Verwendung eines Fehlbedienungszählers, d.h. der Client erneuert die Zerlegung seines geheimen Schlüssels erst nach einer festgelegten Anzahl von fehlgeschlagenen Ausführungen des Protokolls

mit $j \in [0, k2^\ell - 1]$, d.h. $\sum_{i=1}^m f_i d_i$ kann nur $k2^\ell$ verschiedene Werte annehmen. Für $j = 0, \ldots, k2^\ell - 1$ sei P_j die Wahrscheinlichkeit, daß (3.16) gilt. Dann gilt (3.15) mit Wahrscheinlichkeit

$$\sum_{j=0}^{k2^\ell-1} P_j^2 \geq \frac{1}{k2^\ell}\Big(\sum_{j=0}^{k2^\ell-1} P_j\Big)^2 = \frac{1}{k2^\ell}$$

über $\mathbb{Z}$.

Diese Beobachtung zeigt, daß die wiederholte Zerlegung des geheimen Schlüssels mit Wahrscheinlichkeit mindestens $1/(k2^\ell)$ ein modulares ℓ-Bit Rucksackproblem definiert. Wir zeigen, wie sich dieses durch Gitterbasenreduktion lösen läßt. Wir setzen dabei voraus, daß $r := (p-1)/2$ und $s := (q-1)/2$ prim sind.

Im folgenden sei $N > 2^\ell\sqrt{2m}$ und $(\boldsymbol{f}, \boldsymbol{d})$, $(\boldsymbol{f}', \boldsymbol{d}')$ zwei Zerlegungen des geheimen Schlüssels d. Die folgende Überlegung zeigt, daß mindestens mit Wahrscheinlichkeit

$$\frac{1}{k2^\ell} - \frac{2^{2m(\ell+1.0471)}}{rs}$$

der kürzeste Vektor des von

$$\begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_m \\ \boldsymbol{b}_{m+1} \\ \vdots \\ \boldsymbol{b}_{2m} \end{pmatrix} = \begin{pmatrix} 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & 0 \\ 0 & 1 & & 0 & 0 & & 0 & Nd_1^a \\ & & \ddots & & & & & \vdots \\ 0 & 0 & & 1 & 0 & & 0 & Nd_m^a \\ 0 & 0 & & 0 & 1 & & 0 & -Nd_1^b \\ & & & & & \ddots & & \vdots \\ 0 & 0 & & 0 & 0 & & 1 & -Nd_m^b \end{pmatrix}$$

erzeugten Gitters die geheimen Vektoren ($\boldsymbol{f}$ und $\boldsymbol{f}'$) liefert.

Sei $\boldsymbol{x}$ der kürzeste Vektor des von $\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2m}$ und $(0, \ldots, 0, N\phi(n))$ erzeugten Gitters. Dann sind nach Satz 2.4 die Vektoren $\boldsymbol{x} - \langle \boldsymbol{x}, \boldsymbol{b}_0 \rangle \boldsymbol{b}_0$ und $(0, f_1, \ldots f_m, f_1', \ldots, f_m', 0)$ mindestens mit Wahrscheinlichkeit $1 - 2^{2m(\ell+2.0471)}/rs$ über $\mathbb{Z}$ linear abhängig. Andererseits gilt (3.15) mindestens mit Wahrscheinlichkeit $1/(k2^\ell)$. In diesem Fall geht der $(2m+1)$-te Basisvektor $(0, \ldots, 0, N\phi(n))$ nicht in den kürzesten Gittervektor $\boldsymbol{v}$ ein, d.h. $\boldsymbol{v}$ liegt in dem von $\boldsymbol{b}_0, \ldots, \boldsymbol{b}_{2m}$ erzeugten Gitter.

Da die Vektoren $\boldsymbol{d}$ und $\boldsymbol{d}'$ öffentlich bekannt sind, kann ein Angreifer versuchen, durch Reduktion der Gitterbasis $(\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2m})$ die geheimen Vektoren $\boldsymbol{f}$ und $\boldsymbol{f}'$ zu bestimmen. Falls ihm das gelingt, erhält er aus $\sum f_i d_i = d \bmod \phi(n)$ auch den geheimen Schlüssel d.

3.4.2 Der Fall von RSA-S2

Wir nehmen wieder an, der Client erneuere die Zerlegung seines geheimen Schlüssels d von Zeit zu Zeit. Er wählt sich dazu jeweils ein Tripel $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{d})$ mit $\boldsymbol{f}, \boldsymbol{g} \in [0, 2^\ell)^m$, $\mathrm{wt}(\boldsymbol{f}) = \mathrm{wt}(\boldsymbol{g}) = k$, so daß $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i und $\mathrm{ggT}(g_j, \phi(n)) = 1$ für mindestens ein j gilt, und mit $\boldsymbol{d} \in (0, \phi(n))^m$ so, daß $\sum_{i=1}^m f_i d_i = d \bmod p-1$ und $\sum_{i=1}^m g_i d_i = d \bmod q-1$ gilt.

Analog zum Fall von RSA-S1 sieht man, daß (3.15) für zwei Zerlegungen $(\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{d})$ und $(\boldsymbol{f}', \boldsymbol{g}', \boldsymbol{d}')$ mindestens mit Wahrscheinlichkeit $1/(k2^\ell)$ über $\mathbb{Z}$ gilt. Ebenso sieht man, daß

$$\sum g_i d_i = \sum g'_i d' i \tag{3.17}$$

mindestens mit Wahrscheinlichkeit $1/(k2^\ell)$ über $\mathbb{Z}$ gilt. Diese Ergebnisse gelten unabhängig von der Verteilung, nach der die Zerlegungen gewählt werden, sofern alle Zerlegungen gemäß der gleichen Verteilung gewählt werden.

Ebenso wie im Fall von RSA-S1 kann ein Angreifer nun hoffen, daß (3.15) oder (3.17) gilt, und versuchen, durch Reduktion der Gitterbasis $(\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2m})$ das entsprechende Rucksackproblem zu lösen. Wir können allerdings die Wahrscheinlichkeit dafür, daß der kürzeste Gittervektor die gesuchte Lösung des Rucksackproblems liefert, nicht abschätzen. Da $\boldsymbol{d}$ so gewählt wird, daß nicht nur $\sum f_i d_i = d \bmod p-1$ sondern auch $\sum g_i d_i = d \bmod q-1$ gilt, läßt sich Satz 2.4 nicht anwenden. Es ist nicht klar, ob sich für diese Voraussetzungen ein ähnliches Resultat wie Satz 2.4 beweisen läßt. In Experimenten zeigt sich jedoch, daß die Reduktion der Gitterbasis $(\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2m})$ in vielen Fällen die gesuchte Lösung des Rucksackproblems und damit $\boldsymbol{f}$ und $\boldsymbol{f}'$ liefert.

3.4.3 Resultate

In folgenden Tabellen haben wir die Resultate einiger Angriffe gegen RSA-S1 und RSA-S2 mit einem 1000-Bit RSA–Modul zusammengestellt. Hierbei sind $r = (p-1)/2$ und $s = (q-1)/2$ prim und die angegebenen Parameter liefern eine Sicherheit von 2^{72} gegen die Meet-in-the-middle Attacken von Pfitzmann und Waidner.

Die angegebenen Laufzeiten sind Mittelwerte aus jeweils 20 Experimenten. Im Fall von RSA-S1 wurden dabei zuerst zufällige $\boldsymbol{f}, \boldsymbol{f}' \in [0, 2^\ell)^m$ mit Hamminggewicht k gewählt und dann $\boldsymbol{d}, \boldsymbol{d}'$ zufällig so bestimmt, daß sie $\sum f_i d_i = \sum f'_i d'_i \bmod \phi(n)$ erfüllen. Im Fall von RSA-S2 wurden zuerst zufällige $\boldsymbol{f}, \boldsymbol{f}', \boldsymbol{g}, \boldsymbol{g}' \in [0, 2^\ell)^m$ mit Hamminggewicht k gewählt und dann $\boldsymbol{d}, \boldsymbol{d}'$ zufällig so bestimmt, daß sie $\sum f_i d_i = \sum f'_i d'_i \bmod p-1$ und $\sum g_i d_i = \sum g'_i d'_i \bmod q-1$ gilt.

Die resultierende Basis $(\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{2m})$ wurde zunächst LLL–reduziert und – falls notwendig – danach mit einer geschnittenen Blockreduktion mit den angegebenen Parametern

m	k	ℓ	β, p	Zeit(min)	gelöst (%)	Workload (h)
25	28	11	–	0.2	100	180
32	26	10	–	0.5	100	210
38	26	9	20,8	0.8	100	180
42	26	8	20,8	1.4	100	150
48	26	7	20,8	3.4	70	270
56	26	6	20,8	17	10	4700

Tabelle 3.3: Der Angriff gegen RSA-S1 mit 1000-Bit Modul.

m	k	ℓ	β, p	Zeit(min)	gelöst (%)	Workload (h)
20	24	10	–	0.08	85	40
25	24	8	–	0.18	95	20
30	24	7	–	0.38	85	23
35	24	6	–	0.6	85	18
38	22	7	48,8	4.5	70	300
45	22	6	40,8	28	5	13000

Tabelle 3.4: Der Angriff gegen RSA-S2 mit 1000-Bit Modul.

β und p reduziert. Zur Implementierung wurde die von der Arbeitsgruppe *Mathematische Informatik* entwickelte C–Programmbibliothek LARIFARI verwendet. Alle Laufzeiten beziehen sich auf eine HP–Workstation 9000/C200 mit 256 MB Hauptspeicher. Die Workload in Spalte 7 ist die Laufzeit dividiert durch die Wahrscheinlichkeit $1/(k2^\ell)$ für die Relation (3.15) bzw. (3.17) und dividiert durch die Erfolgsquote aus Spalte 6.

3.4.4 Gegenmaßnahmen.

Die beschriebenen Angriffe können leicht abgewehrt werden, indem man die d_i aus einem größeren Intervall wählt. Wir betrachten den Fall von RSA-S1.

Wir nehmen an, die Zerlegungen werden folgendermaßen erzeugt. Zunächst wählt man ein zufälliges $\boldsymbol{f} \in [0, 2^\ell)^m$ mit Hamminggewicht k so, daß $\mathrm{ggT}(f_i, \phi(n)) = 1$ für mindestens ein i gilt. Sei i_0 das kleinste i für das dieses gilt. Für eine Konstante $c \in \mathbb{N}$ werden dann die d_i mit $i \neq i_0$ gleichmäßig und unabhängig voneinander aus $[0, c\phi(n))$ gewählt, wobei $c \in \mathbb{N}$ eine Konstante ist. Schließlich wird $d_{i_0} := f_{i_0}^{-1} \sum_{i \neq i_0} f_i d_i \bmod \phi(n) \in [0, \phi(n))$ gesetzt. Der Speicherbedarf des Klienten erhöht sich durch diese Art der Zerlegung lediglich um $m \log c$ Bits. Die folgende Überlegung zeigt, daß die beschriebene Attacke bereits für $c = 2^{50}$

abgewehrt wird.

Es gilt $\sum f_i d_i \le ck2^\ell \phi(n)$ und damit (3.16) für ein $j \in [0, ck2^\ell - 1]$. Für $j = 0, \ldots, ck2^\ell - 1$ sei P_j die Wahrscheinlichkeit, daß (3.16) gilt. Dann ist die Wahrscheinlichkeit dafür, daß (3.15) gilt,

$$\sum_{j=0}^{ck2^\ell - 1} P_j^2 \;\le\; \max_j(P_j) \sum_{j=0}^{ck2^\ell - 1} P_j \;=\; \max_j(P_j) \;\; .$$

Wir zeigen nun, daß $P_j \le 1/c$ für alle j gilt. Damit gilt auch (3.15) höchstens mit Wahrscheinlichkeit $1/c$.

Man überzeugt sich leicht davon, daß (3.16) für ein $j \in [0, ck2^\ell - 1]$ genau dann gilt, wenn $\sum_{i \ne i_0} f_i d_i$ im Intervall $I_j := [d + (j-1)\phi(n),\ d + j\phi(n)]$ liegt. Sei i_1 das minimale $i \ne i_0$ mit $f_i \ne 0$. Wenn wir alle d_i mit $i \ne i_0, i_1$ fixieren, variiert $\sum_{i \ne i_0} f_i d_i$ mit $d_{i_1} \in [0, c\phi(n))$ über $c\phi(n)$ Werte. Es folgt daraus

$$P_j \;\le\; |I_j| / (c\phi(n)) \;\le\; 1/c$$

für alle j.

Kapitel 4

Pseudozufällige Präsignaturen

In Signaturverfahren, die auf dem Problem des diskreten Logarithmus beruhen, benötigt man Paare (r, g^r) mit zufälligem $r \in \{0, \ldots, q-1\}$ und festem Generator g einer endlichen zyklischen Gruppe G der primen Ordnung q; wir bezeichnen solche Paare als *Präsignaturen.* Beispiele hierfür sind das DSS von NIST ([NIS94]) sowie die Signaturverfahren von El Gamal ([ElG85]) und von Schnorr ([Sch89],[Sch91]). Der größte Rechenaufwand besteht für den Unterzeichnenden in der Berechnung des Exponentialwertes g^r aus einem zufälligen r. Diese Exponentiation benötigt im Mittel $1.5 \log q$ viele Multiplikationen in G.

Durch das Speichern einer konstanten Anzahl von Präsignaturen $(r_1, g^{r_1}), \ldots, (r_K, g^{r_K})$ kann dieser Aufwand wesentlich verringert werden. Eine Möglichkeit besteht darin, einen Exponenten r zufällig aus $(0, q)$ zu wählen, diesen als Summe der gespeicherten r_i darzustellen und den Wert g^r als das (der Summe) entsprechende Produkt der g^{r_i} zu berechnen. Diese Vorgehensweise nennt man *Exponentiation mit Precomputation.* Im folgenden Abschnitt geben wir einen kurzen Überblick über die wichtigsten Verfahren dieser Art.

In dieser Arbeit verfolgen wir jedoch einen anderen Ansatz. Anstatt den Exponenten r zuerst zufällig zu wählen und dann als Summe der r_i darzustellen, wählen wir gleich eine zufällige Summe der r_i. Danach berechnen wir den Wert g^r als das entsprechende Produkt der g^{r_i}. Damit integrieren wir die zufällige Wahl von r und seine Zerlegung in eine Summe in einen einzigen Schritt. Im Unterschied zur Exponentiation mit Precomputation ist dabei die Verteilung des Exponenten r im allgemeinen nicht gleichmäßig auf $(0, q)$, sondern hängt von der zufälligen Wahl der gespeicherten Präsignaturen und der zufälligen Wahl der Summe der r_i ab. Auf der anderen Seite hängt die Anzahl der notwendigen Multiplikationen nicht mehr von q, sondern nur von dem gewünschten Grad der Sicherheit ab.

Besonders effizient lassen sich die Präsignaturen mit diesem Ansatz erzeugen, wenn die gespeicherten Exponenten r_i oder sogar einige der gespeicherten Präsignaturen geheim gehalten werden. Für die typischen Parameter in den DL-Signaturverfahren sind unsere Verfahren weit effizienter als eine Exponentiation eines zufälligen r mit Precomputation.

Dieser Ansatz ist nicht neu. Schnorr veröffentlichte bereits zwei Verfahren für die Generierung pseudozufälliger Präsignaturen ([Sch89],[Sch91]). Beide Verfahren wurden jedoch von de Rooij gebrochen ([dR91],[dR97]). Weitere Verfahren wurden in [Mer95] entwickelt; die Sicherheitsanalysen beschränkten sich dabei jedoch auf Heuristiken. Kürzlich veröffentlichten Boyko, Peinado und Venkatesan ([BPV98]) ebenfalls Generatoren für pseudozufällige Präsignaturen. Ihre Verfahren sind jedoch für Chipkarten ungeeignet – es müssen 250–500 Präsignaturen gespeichert werden.

4.1 Exponentiation mit Precomputation

Wir stellen nun einige Verfahren zur Exponentiation mit Precomputation eines zufälligen r zu fester Basis $g \in G$ in einer endlichen Gruppe G vor. Sei $q := \mathrm{ord}(g)$.

In [BGMW92] beschreiben Brickell, Gordon, Mc Curley und Wilson ein allgemeines Verfahren für die effiziente Berechnung von g^r mit Precomputation. Wir stellen hier die beiden für $q \approx 2^{160}$ effizientesten Varianten vor.

Algorithmus BGMW-1
Für $b \geq 2$ und $m := \lceil \log_b q \rceil$ seien die Werte $g, g^b, g^{b^2} \ldots, g^{b^{m-1}}$ vorberechnet. Jedes $r \in [0, q)$ läßt sich durch $r = \sum_{i=0}^{m-1} a_i b^i$ mit $a_0, \ldots, a_{m-1} \in [0, b)$ ausdrücken. Der folgende Algorithmus berechnet g^r aus den vorgespeicherten Werten und $a_0, \ldots, a_{m-1}$.

$B := 1$, $A := 1$
for $j = b-1$ to 1 do
  for each i such that $a_i = j$ do
    $B := B \cdot g^{b^i}$
  $A := A \cdot B$
return A

Das Verfahren benötigt höchstens $b - 1 + |\{i \mid a_i \neq 0\}| - 2$ Multiplikationen. Für zufälliges r (d.h. gleichverteilte und unabhängige a_i) ergibt sich damit ein durchschnittlicher Aufwand von höchstens $b + \frac{b-1}{b} m - 3$ Multiplikationen bei m gespeicherten Werten.

Algorithmus BGMW-2
Für $b \geq 2$ und $m := \lceil \log_b (2q+1) \rceil$ seien die Werte $g^{\pm 1}, g^{\pm b}, g^{\pm b^2}, \ldots, g^{\pm b^{m-1}}$ vorberechnet. Jedes $r \in [0, q)$ läßt sich durch

$$r = \sum_{i=0}^{m-1} \epsilon_i a_i b^i$$

mit $a_0, \dots, a_{m-1} \in [0, \lfloor b/2 \rfloor)$ und $\epsilon_0, \dots, \epsilon_{m-1} \in \{-1, 1\}$ ausdrücken (siehe [BGMW92]). Der folgende Algorithmus berechnet g^r aus den vorgespeicherten Werten, $a_0, \dots, a_{m-1}$ und $\epsilon_0, \dots, \epsilon_{m-1}$.

```
B := 1, A := 1
for j = ⌊b/2⌋ to 1 do
    for each i such that a_i = j do
        B := B · g^(ε_i b^i)
    A := A · B
return A
```

Das Verfahren benötigt höchstens $\lfloor b/2 \rfloor + |\{i \mid a_i \neq 0\}| - 2$ Multiplikationen. Damit ergibt sich ein durchschnittlicher Aufwand von höchstens $\lfloor b/2 \rfloor + \frac{b-1}{b} m - 2$ Multiplikationen bei $2m$ gespeicherten Werten.

Dieses zweite Verfahren ist besonders für den Fall elliptischer Kurven interessant, da hier die Invertierung von Gruppenelementen fast „kostenlos" ist (siehe Abschnitt 4.4) und daher die Werte $g^{-1}, g^{-b}, \dots, g^{-b^{m-1}}$ nicht gespeichert werden müssen.

1994 veröffentlichte de Rooij in [dR94] einen Algorithmus, der, basierend auf dem euklidischen Algorithmus, eine kurze Additionskette für ein gegebenes r ermittelt und die Exponentiation gemäß dieser ausführt.

Algorithmus de Rooij
Für $b \geq 2$ und $m := \lceil \log_b q \rceil$ seien die Werte $g^b, g^{b^2} \dots, g^{b^{m-1}}$ vorberechnet. Außerdem gelte $r = \sum_{i=0}^{m-1} a_i b^i$ mit $a_0, \dots, a_{m-1} \in [0, b)$.

```
for i = 0 to m − 1 do x_i := a_i, g_i := g^(b^i)
find M such that x_M ≥ x_i for all i and N ≠ M such that x_N ≥ x_i for all i ≠ M
while x_N ≠ 0 do
    c := ⌊x_M/x_N⌋, g_N := (g_M)^c · g_N, x_M := x_M mod x_N
    find M such that x_M ≥ x_i for all i and N ≠ M such that x_N ≥ x_i for all i ≠ M
return (g_M)^(x_M)
```

Die Korrektheit des Algorithmus Nach der Initialisierung der x_i und g_i gilt offensichtlich $\prod_{i=0}^{m-1} (g_i)^{x_i} = g^r$. Wegen der Identität $g_N^{x_N} g_M^{x_M} = \left(g_N g_M^{\lfloor x_M/x_N \rfloor}\right)^{x_N} g_M^{x_M \bmod x_N}$ bleibt diese Gleichung bei jedem Durchlauf der while-Schleife erhalten. Auf der anderen Seite gilt am Ende $x_N = 0$ und damit $x_i = 0$ für alle $i \neq M$. Somit ist die Ausgabe gleich g^r.

Die Effizient des Algorithmus De Rooij argumentiert, daß die Werte x_i im Vergleich zu q klein sind und daher der Aufwand für die Division x_M/x_N vernachlässigbar ist. In einer heuristischen Analyse schätzt er die durchschnittliche Zahl der Multiplikationen für die Exponentiation eines zufälligen r durch $\log b / \log \rho$ ab, wobei ρ die größte reelle Lösung von $X^m = X^{m-1} + 1$ ist. Für die Details verweisen wir auf [dR94].

Bemerkung. Neben $g^b, g^{b^2} \ldots, g^{b^{m-1}}$ müssen während der Berechnung auch die Werte $g_0, \ldots, g_{m-1}$ und $x_0, \ldots, x_{m-1}$ gespeichert werden. In jedem Berechnungsschritt wird ein Paar (g_i, r_i) verändert. Bei einer Anwendung in Chipkarten ist jedoch das Speichern und wiederholte Überschreiben vieler Werte problematisch: Beim heutigen Stand der Technik müßte diese entweder im EEPROM (Electrically Erasable Programmable Read Only Memory) oder im RAM (Random Access Memory) gehalten werden. Da das Beschreiben von EEPROM zeitaufwendig ist (siehe [RE96]), kommt dieses Speichermedium hier nicht in Frage. Auf der anderen Seite ist RAM auf Chipkarten wegen seiner relativ großen Oberfläche pro Speichereinheit äußerst knapp (128 – 1024 Byte). Für Chipkarten ist der Algorithmus von de Rooij deshalb im Fall $G \subset \mathbb{Z}_p^*$ mit $p \geq 2^{750}$ nur für $m \leq 6$ praktikabel.

Ein von Lim und Lee veröffentlichtes Verfahren zur Exponentiation mit Precomputation ([LL94]) erlaubt eine flexiblere Wahl des Speicherbedarfs und der benötigten Multiplikationen.

Algorithmus Lim Lee
Für festes $1 \leq h \leq \lceil \log q \rceil$ sei $a = \lceil (\log q)/h \rceil$. Weiterhin sei $b = \lceil a/v \rceil$ für festes $1 \leq v \leq a$. h, v sind die Parameter des Algorithmus.

Jedes $r \in [0, q)$ läßt sich nun zu $r = \sum_{i=0}^{h-1} r_i 2^{ia}$ mit $r_i \in [0, 2^a)$ aufspalten. Jedes der r_i können wir wiederum als $r_i = \sum_{j=0}^{v-1} r_{ij} 2^{jb}$ mit $r_{ij} \in [0, 2^b)$ schreiben. Schließlich können wir auch alle r_{ij} zu $r_{ij} = \sum_{k=0}^{b-1} r_{ijk} 2^k$ mit $r_{ijk} \in \{0, 1\}$ aufspalten. Mit diesen Bezeichnungen folgt nun

$$\begin{aligned} g^r &= \prod_{i=0}^{h-1} \prod_{j=0}^{v-1} \prod_{k=0}^{b-1} g^{r_{ijk} 2^{ia+jb+k}} \\ &= \prod_{k=0}^{b-1} \prod_{j=0}^{v-1} \Big(\prod_{i=0}^{h-1} g^{r_{ijk} 2^{ia}} \Big)^{2^{jb+k}} \quad . \end{aligned}$$

Wenn wir nun $G[0][e] := \prod_{i=0}^{h-1} g^{e_i 2^{ia}}$ für $e = \sum_{i=0}^{h-1} e_i 2^i$ und $G[f][e] := G[0][e]^{2^{fb}}$ setzen, erhalten wir somit

$$\begin{aligned} g^r &= \prod_{k=0}^{b-1} \prod_{j=0}^{v-1} G[0][x_{jk}]^{2^{jb+k}} \\ &= \prod_{k=0}^{b-1} \prod_{j=0}^{v-1} G[j][x_{jk}]^{2^k} \end{aligned}$$

mit $x_{jk} := \sum_{i=0}^{h-1} r_{ijk} 2^i$.

Der folgende Algorithmus berechnet nun g^r zu gegebenem r aus den vorgespeicherten Werten $G[f][e]$ für alle $e \in [0, 2^h)$ und $f \in [0, v)$.

```
A:=1
for k from b − 1 to 0 do
    A := A · A
    for j from v − 1 to 0 do
        A := A · G[j][x_jk]
return A
```

Im Durchschnitt benötigt der Algorithmus $\frac{2^h-1}{2^h} bv + b - 2$ Multiplikationen bei $v(2^h - 1)$ gespeicherten Werten.

Tabelle 4.1 vergleicht die vorgestellten Verfahren für $q \approx 2^{160}$. Mem gibt dabei die Anzahl der zu speichernden Werte (inklusive g) und Mult die durchschnittlich benötigte Anzahl von Multiplikationen an. Für konkrete Rechenbeispiele und ergänzende Bemerkungen verweisen wir auf [MvOV97, 623ff.].

Verfahren	Parameter	Mem	Mult
BGMW-1	$b = 12$	45	50.3
	$b = 32$	32	60
BGMW-2	$b = 33$	64	45
de Rooij	$b = 40$	4[1]	80[2]
Lim Lee	$h = 1, v = 4$	4	118
	$h = 2, v = 3$	9	85.8
	$h = 3, v = 2$	14	72.3
	$h = 4, v = 2$	30	55.5
	$h = 4, v = 4$	60	45.5
	$h = 6, v = 2$	124	37

Tabelle 4.1: Vergleich der Verfahren zur Exponentiation mit Precomputation.

[1] Zusätzlich müssen vier variable Werte temporär gespeichert werden

[2] Dieser Wert wurde von de Rooij in [dR94] empirisch abgeschätzt

4.2 Der allgemeine Generator

Wir präsentieren zunächst die allgemeine Form unseres Generators für pseudozufällige Präsignaturen. Alle später beschriebenen Verfahren basieren auf dieser Grundform. Dabei betrachten wir nur die additiven Gleichungen in den Exponenten $r_1, \ldots, r_k$. Die Exponentialwerte $g^{r_1}, \ldots, g^{r_k}$ werden entsprechend (multiplikativ) mitberechnet. Die Parameter des Generators sind k, m, ℓ, A.

Der allgemeine Generator

1. EINGABE $\boldsymbol{r} = (r_1, \ldots, r_k) \in_R \mathbb{Z}_q^k$ und $\boldsymbol{s} = (s_1, \ldots, s_m) \in_R \mathbb{Z}_q^m$
   $\nu := 1$ (ν ist die Nummer der Runde).
2. Wähle einen zufälligen Vektor $\boldsymbol{u}_\nu = (u_{1,\nu}, \ldots, u_{k,\nu}) \in [0, 2^\ell)^k$ mit Hamminggewicht A und setze $r' := \sum_{i=1}^k u_{i,\nu} r_i \bmod q$.
3. Führe eine lineare Transformation auf $s_1, \ldots, s_m$ aus.
4. AUSGABE $r_\nu^* := r' + s_m$ (verwende $(r_\nu^*, g^{r_\nu^*})$ für die nächste Signatur)
   $\nu := \nu + 1$,
   GO TO 2 für die nächste Runde.

Die Werte $s_1, \ldots, s_m$ und $g^{s_1}, \ldots, g^{s_m}$ werden geheimgehalten. Das bedeutet, daß sie in einem auslesesicheren Teil des Speichers verwahrt werden müssen. Wir setzen $s_0 := 0$, d.h. im Fall $m = 0$ wird in Schritt 4 $r_\nu^* := r'$ gesetzt. Der Generator benötigt in diesem Fall $A + \ell - 2$ Multiplikationen für den multiplikativen Teil – die Berechnung von $g^{r'} \bmod p$ wird mit der „Square and multiply“ Methode durchgeführt. Für die Transformation in Schritt 3 stellen wir in Abschnitt 4.3 verschiedene Möglichkeiten vor.

Für die Verwahrung der Präsignaturen $(r_1, g^{r_1}), \ldots, (r_k, g^{r_k})$ sind drei Szenarien denkbar:

Der geheime Fall. $r_1, \ldots, r_k$ und ihrer Exponentialwerte $g^{r_1}, \ldots, g^{r_k}$ werden geheimgehalten. Es wird sich im Verlauf unserer Analyse zeigen, daß der Vorteil dieser Methode gegenüber dem halböffentlichen Fall gering ist.

Der halböffentliche Fall. $r_1, \ldots, r_k$ werden geheimgehalten, ihre Exponentialwerte jedoch nicht. Da in den typischen Anwendungen die Kodierung der r_i wesentlich kürzer ist als die ihre Expontialwerte, muß in diesem Fall nur ein kleiner Anteil des Speichers auslesesicher sein. Andererseits werden wir zeigen, daß der Generator im halböffentlichen Fall genauso sicher ist, wie im geheimen Fall, sofern das Berechnen des diskreten Logarithmus schwer ist.

Der öffentliche Fall. Weder $r_1, \ldots, r_k$ noch ihre Exponentialwerte werden geheimgehalten. Diese Methode stellt die geringsten Anforderungen an die Hardware. Auf der anderen Seite wird unsere Analyse zeigen, daß der öffentliche Fall Gitterattacken ermöglicht. Außerdem treten in diesem Fall technische Schwierigkeiten in unserer Analyse auf. Nicht alle unsere Sicherheitsbeweise lassen sich auf den öffentichen Fall übertragen.

Öffentliche Werte müssen nicht auf der Karte gespeichert werden. Sie können bei jeder Anwendung von einem externen Rechner (ein solcher ist in den typischen Anwendungen meist anwesend) eingelesen werden. Es ist in diesem Fall jedoch notwendig, die Korrektheit der eingelesenen Werte zu überprüfen. Das kann durch Anwendung einer Hashfunktion auf einige oder alle Werte und den Vergleich mit den auf der Karte gespeicherten Hash–Werten geschehen.

4.2.1 Das Sicherheitskonzept

Bei den in [Sch89] und [Sch91] beschriebenen Generatoren wurde die Ausgabe durch die feste Kombination der Präsignaturen $r_{\nu \bmod k} + 2r_{\nu-1 \bmod k}$ gebildet. Im Unterschied dazu bildet unser Generator die Ausgabe r_ν^* im wesentlichen aus einer zufälligen Linearkombination fester Präsignaturen. Dadurch sind wir in der Lage, die Sicherheit unseres Generators gegen Angriffe wie die von de Rooij ([dR91][dR97]) zu beweisen.

Wir nehmen an, der Generator habe n Runden durchlaufen und dabei mit Hilfe der Münzwurffolge ω die Präsignaturen $(r_1^*, g^{r_1^*}), \ldots, (r_n^*, g^{r_n^*})$ ausgegeben. Weiter nehmen wir an, mit diesen Präsignaturen seien die Unterschriften $(y_1, e_1), \ldots, (y_n, e_n)$ erzeugt worden. Dann gilt

$$y_\nu = se_\nu + r_\nu^* \bmod q \quad \text{für } \nu = 1, \ldots, n. \tag{4.1}$$

Die Präsignaturen hängen dabei von den internen Münzwürfen ω des Generators und dem Initialvektor $\boldsymbol{r}$ ab. Für $r_1^*, \ldots, r_n^*$ schreiben wir kurz $\boldsymbol{r}^*$. Das Ziel eines Angreifers ist es nun, den geheimen Schlüssel s zu ermitteln. Mit Hilfe einer Relation $\sum_\nu c_\nu r_\nu^* = c_0 \bmod q$ ist dies möglich, sofern diese Gleichung linear unabhängig von den Gleichungen (4.1) ist. Umgekehrt führt aber die Kenntnis des geheimen Schlüssels s sofort zu einer Relation der r_ν^*. Unsere Sicherheitsanalysen konzentrieren sich deshalb auf das Auffinden solcher Relationen. Im folgenden setzen wir $r_0^* := 1$ und indizieren die Vektoren $\boldsymbol{x}$ aus $\mathbb{Z}^{n+1}$ als $(x_0, \ldots, x_n)$. Mit dieser Notation schreibt sich eine Relation als $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$.

Der geheime Fall

Wir betrachten zunächst den geheimen Fall.

Definition 4.1 *Sei $\mathcal{C}$ die Menge der Vektoren $\boldsymbol{u} \in [0, 2^{\ell}-1]^k$ mit Hamminggewicht A. Eine* einfache Attacke *ist ein Vektor $\boldsymbol{c} \in \mathbb{Z}^{n+1}$. Dabei heißt n die* Länge *von $\boldsymbol{c}$. Die* Erfolgswahrscheinlichkeit $\mathrm{S}(\boldsymbol{c})$ *einer einfachen Attacke $\boldsymbol{c}$ ist definiert als* $\mathrm{Ws}\left[\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle \equiv 0 \bmod q\right]$, *wobei $\equiv$ bedeutet, daß die Gleichheit unabhängig von $\boldsymbol{r}$ und $\boldsymbol{s}$ gilt. Die Wahrscheinlichkeit wird dabei die internen Münzwürfe des Generators*[3] *genommen. Die* Güte des Generators *definieren wir dann als*

$$\mathrm{SEC} := \min\{\mathrm{S}(\boldsymbol{c})^{-1} \mid \boldsymbol{c} \in \mathbb{Z}^{n+1}, \, \boldsymbol{c} \neq \boldsymbol{0} \bmod q, \, n \in \mathbb{N}\} \ .$$

Der Konstruktion unseres Generators entnehmen wir, daß $r_\nu^* = s_m^{[\nu]} + \sum_{i=1}^k u_{i,\nu} r_i \bmod q$ gilt, wobei $s_m^{[\nu]}$ den Wert von s_m in der Runde ν bezeichnet. Die Gleichung $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ ist also äquivalent zu

$$\sum_{i=1}^{k} r_i \sum_{\nu=1}^{n} c_\nu u_{i,\nu} + \sum_{\nu=1}^{n} c_\nu s_m^{[\nu]} = c_0 \bmod q \ . \tag{4.2}$$

Falls diese Gleichung unabhängig von $\boldsymbol{r}$ und $\boldsymbol{s}$ gilt, müssen die Summen $\sum_{\nu=1}^n c_\nu u_{i,\nu}$ für $i = 1, \ldots, k$ modulo q verschwinden. Wir erhalten also

$$\mathrm{S}(\boldsymbol{c}) \leq \mathrm{Ws}\left[\sum_{\nu=1}^{n} c_\nu \boldsymbol{u}_\nu = 0 \bmod q\right] \tag{4.3}$$

für alle einfachen Attacken $\boldsymbol{c}$.

Der folgende Satz zeigt, daß unser Gütebegriff im geheimen Fall eine untere Schranke für die Komplexität von Angriffen durch generische Algorithmen liefert.

Satz 4.1 *Sei $\mathcal{A}$ ein generischer Algorithmus der Länge t, der für zufälliges $\boldsymbol{r} \in \mathbb{Z}_q^k$ und zufällige $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ die Ausgaben $x_1^*, \ldots, x_n^*$ des Generators und $x_0^* := g$ als Eingabe erhält und eine einfache Attacke $\boldsymbol{c}$ mit $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ ausgibt. Dann hat $\mathcal{A}$ höchstens die Erfolgswahrscheinlichkeit $t^2(\mathrm{SEC}^{-1} + q^{-1})/2$.*

Beweis. Wir beweisen das Resultat in zwei Schritten: Zunächst zeigen wir, daß $\mathcal{A}$ höchstens die Erfolgswahrscheinlichkeit $Pt^2/2$ besitzt, sofern $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ für alle $\boldsymbol{c} \in \mathbb{Z}^{n+1}$ höchstens mit Wahrscheinlichkeit P gilt. Dann zeigen wir, daß $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ für alle $\boldsymbol{c}$ höchstens mit Wahrscheinlichkeit $1/\mathrm{SEC} + 1/q$ gilt.

Für $\nu = n+1, \ldots, t$ sei x_ν^* das von $\mathcal{A}$ in Schritt ν berechnete Element. Dann gibt es $\boldsymbol{c}^{[0]}, \ldots, \boldsymbol{c}^{[t]} \in \mathbb{Z}^{n+1}$ mit

$$x_\nu^* \ = \ \prod_{i=0}^{n} (x_i^*)^{c_i^{[\nu]}} \ = \ g^{\langle \boldsymbol{c}^{[\nu]}, \boldsymbol{r}^* \rangle}$$

[3] d.h. über zufällige $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ und die bei der Transformation der s_i verwendeten Zufallsbits

für $\nu = 0, \dots, t$. Wir können ohne Einschränkung $\boldsymbol{c}^{[\nu]} \neq \boldsymbol{c}^{[\mu]} \bmod q$ für $\nu \neq \mu$ annehmen. Da $\mathcal{A}$ generisch ist, hängen die Vektoren $\boldsymbol{c}^{[\nu]}$ nur von der Schrittzahl ν und der Kollisionsmenge $\mathcal{CO}_{\nu-1}$ ab.

Wir nehmen nun an, für alle $\boldsymbol{c} \in \mathbb{Z}^{n+1}$ gilt $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ höchstens mit Wahrscheinlichkeit P. Die folgenden Überlegungen zeigen, daß dann $\mathcal{A}$ höchstens die Erfolgswahrscheinlichkeit $Pt^2/2$ besitzt.

Wir schätzen zunächst die Wahrscheinlichkeit dafür ab, daß eine Kollision der berechneten Gruppenelemente auftritt. Angenommen, in den Schritten $1, \dots, \nu-1$ ist keine Kollision aufgetreten. Falls nun $x_\nu^* = x_\mu^*$ für ein $\nu < \mu$ gilt, erhalten wir

$$\langle \boldsymbol{c}^{[\nu]}, \boldsymbol{r}^* \rangle = \langle \boldsymbol{c}^{[\mu]}, \boldsymbol{r}^* \rangle \bmod q \ . \tag{4.4}$$

Auf der anderen Seite gilt aber $\mathcal{CO}_0 = \cdots = \mathcal{CO}_{\nu-1} = \emptyset$ und die Vektoren $\boldsymbol{c}^{[\nu]}$ und $\boldsymbol{c}^{[\mu]}$ sind konstant. Damit gilt (4.4) höchstens mit Wahrscheinlichkeit P. Durch Summation über $\mu = 1, \dots, \nu - 1$ folgt also, daß höchstens mit Wahrscheinlichkeit $(\nu - 1)P$ in Schritt ν eine Kollision auftritt. Indem wir über alle ν summieren, erhalten wir, daß $\mathcal{CO}_t \neq \emptyset$ höchstens mit Wahrscheinlichkeit $\binom{t}{2}P$ gilt.

Es bleibt noch der Fall zu betrachten, daß keine Kollision auftritt. Da die Ausgabe $\boldsymbol{c}$ von $\mathcal{A}$ nur von t und $\mathcal{CO}_t$ abhängt, ist sie in diesem Fall konstant. Daher ist die Wahrscheinlichkeit, daß $\mathcal{CO}_t = \emptyset$ gilt und die Ausgabe korrekt ist, höchstens P.

Insgesamt besitzt $\mathcal{A}$ also höchstens die Erfolgswahrscheinlichkeit $\binom{t}{2}P$. Wir zeigen nun im zweiten Schritt, daß $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ für alle $\boldsymbol{c}$ höchstens mit Wahrscheinlichkeit $1/\mathrm{SEC} + 1/q$ gilt.

Nach Definition gilt $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle \equiv 0 \bmod q$ höchstens mit Wahrscheinlichkeit $1/\mathrm{SEC}$. Falls jedoch $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle \not\equiv 0 \bmod q$ und $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ gilt, so folgt daraus eine nichttriviale Gleichung

$$\sum_{i=1}^{k} \alpha_i r_i + \sum_{i=1}^{m} \alpha_{i+k} s_i = 0 \bmod q \ ,$$

wobei nicht alle α_i Null sind. Für zufällige $\boldsymbol{r}, \boldsymbol{s}$ und t gilt eine solche Gleichung höchstens mit Wahrscheinlichkeit $1/q$.

□

Wir beweisen nun eine untere Schranke für die Güte des allgemeinen Generators. Im Fall $m = 0$ (d.h. wenn der Generator keine s_i verwendet) ist die Schranke scharf, wie das Beispiel der einfachen Attacke $(0, 1, -1)$ zeigt.

Satz 4.2 *Der Generator besitzt mindestens die Güte* $\binom{k\ell}{A}$.

Beweis. Wir betrachten die Gleichung $\sum_\nu c_\nu \boldsymbol{u}_\nu = \boldsymbol{0} \bmod q$. Es ist keine Einschränkung, wenn wir $c_n \neq 0$ annehmen. Für feste $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n-1}$ kann diese Gleichung nur für ein $\boldsymbol{u}_n$ gelten. Nach (4.3) folgt daher $\mathrm{S}(\boldsymbol{c}) \leq |\mathcal{C}|^{-1} = \binom{k\ell}{A}^{-1}$.

□

Der halböffentliche Fall

Im halböffentlichen Fall sind die Exponentialwerte $g^{r_1}, \ldots, g^{r_k}$ öffentlich bekannt. Der Einfachheit halber gehen wir im folgenden davon aus, daß der Generator in den Runden $1, \ldots, k$ die Präsignaturen $(r_1, g^{r_1}), \ldots, (r_k, g^{r_k})$ ausgibt. Für $i = 1, \ldots, k$ ergibt sich dann $\boldsymbol{u}_i = \boldsymbol{e}_i$.

Der folgende Satz zeigt, daß die Kenntnis der Exponentialwerte $g^{r_1}, \ldots, g^{r_k}$ einem Angreifer keine Vorteile bringt, sofern das Berechnen des diskreten Logarithmus schwer ist. Wir setzen dabei $E := n\mathrm{Mult} + 2k \log q$, wobei Mult die Anzahl der Multiplikationen ist, die der Generator pro Ausgabe benötigt. Im Fall $m = 0$ gilt $\mathrm{Mult} = A + \ell - 2$.

Satz 4.3 *Sei $\mathcal{A}$ ein Algorithmus, der für zufälliges $\boldsymbol{r} \in \mathbb{Z}_q^k$ und zufällige $\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ die Eingaben $x_1^*, \ldots, x_n^*$ erhält und in Zeit $T_\mathcal{A}$ mit Wahrscheinlichkeit $P_\mathcal{A}$ eine einfache Attacke $\boldsymbol{c}$ mit $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ und $\sum_{\nu=1}^n c_\nu \boldsymbol{u}_\nu \neq \boldsymbol{0} \bmod q$ ausgibt. Dann gibt es einen Algorithmus $\mathcal{B}$, der für ein zufälliges $a \in G$ in Zeit $T_\mathcal{A} + O(E \log^2 q)$ mit Wahrscheinlichkeit $P_\mathcal{A}(1 - 1/q)$ ein b mit $g^b = a$ berechnet. Ist $\mathcal{A}$ generisch der Länge $t_\mathcal{A}$, so ist auch $\mathcal{B}$ generisch der Länge $t_\mathcal{A} + E$.*

Beweis. Der Algorithmus $\mathcal{B}$ arbeitet wie folgt. Er wählt zufällige $\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ und zufällige $\boldsymbol{e}, \boldsymbol{f} \in \mathbb{Z}_q^k$. Dann setzt er $z_i := a^{e_i} g^{f_i}$ für $1 \leq i \leq k$ und $x_\nu^* := \prod_{i=1}^k z_i^{u_{i,\nu}} \bmod p$ für $\nu = k+1, \ldots, n$. Nun startet er $\mathcal{A}$ mit der Eingabe $z_1, \ldots, z_k, x_1^*, \ldots, x_n^*$. Falls $\mathcal{A}$ eine Attacke $\boldsymbol{c}$ mit $\boldsymbol{d} := \sum_{\nu=1}^n c_\nu \boldsymbol{u}_\nu \neq \boldsymbol{0} \bmod q$ ausgibt, setzt er $b := -\langle \boldsymbol{d}, \boldsymbol{f} \rangle / \langle \boldsymbol{d}, \boldsymbol{e} \rangle \bmod q$.

Nach Konstruktion gilt $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ und $\boldsymbol{d} \neq \boldsymbol{0} \bmod q$ mindestens mit Wahrscheinlichkeit $P_\mathcal{A}$. In diesem Fall folgt

$$a^{\langle \boldsymbol{d}, \boldsymbol{e} \rangle} g^{\langle \boldsymbol{d}, \boldsymbol{f} \rangle} = \prod_{i=1}^k a^{e_i d_i} g^{f_i d_i} = \prod_{i=1}^k z_i^{d_i} = \prod_{\nu=1}^n (x_\nu^*)^{c_\nu} = 1 \ .$$

Da $\boldsymbol{e}$ zufällig und statistisch unabhängig von den Eingaben von $\mathcal{A}$ ist, gilt $\langle \boldsymbol{d}, \boldsymbol{e} \rangle \neq 0 \bmod q$ und damit $g^b = a$ mindestens mit Wahrscheinlichkeit $1 - 1/q$.

□

Analog zum geheimen Fall erhalten wir aus der Güte des Generators eine untere Schranke für die Sicherheit gegen generische Algorithmen.

Mem	k, ℓ, A	$Mult$	Prec.-Methode	$Mult$
8	8,30,43	71	de Rooij	59
16	16,19,37	54	de Rooij	52
30	30,14,32	44	Lim-Lee	55.5
60	60,10,28	36	Lim-Lee	45.5
124	124,7,25	30	Lim-Lee	37

Tabelle 4.2: Vergleich des allgemeinen Generators ($m = 0$) mit den Precomputation Verfahren aus Abschnitt 4.1 für $q \approx 2^{160}$.

Satz 4.4 *Sei $\mathcal{A}$ ein generischer Algorithmus der Länge t, der für zufälliges $\boldsymbol{r} \in \mathbb{Z}_q^k$ und zufällige $\boldsymbol{u}_{k+1}, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ die Ausgaben $x_1^*, \ldots, x_n^*$ erhält und eine einfache Attacke $\boldsymbol{c}$ mit $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ ausgibt. Dann hat $\mathcal{A}$ höchstens die Erfolgswahrscheinlichkeit $t^2(\mathrm{SEC}^{-1} + q^{-1})/2$.*

Resultate. Tabelle 4.2 gibt einige Beispiele für Parameter an, die eine Güte von ungefähr 2^{160} liefern. Mem bezeichnet die Anzahl der zu speichernden Gruppenelemente und $Mult$ die Anzahl der Multiplikationen pro Ausgabe.

Bemerkung. Die in Tabelle 4.2 gegenüber gestellten Werte lassen sich nur bedingt miteinander vergleichen. Bei unserem Generator müssen neben den Exponentialwerten g^{r_i} auch die Exponenten r_i gespeichert werden, während dies bei den in Abschnitt 4.1 angegebenen Verfahren nicht notwendig ist. (Die Exponenten sind dort in kanonischer Weise gewählt, so daß sie in sehr komprimierter Form gespeichert werden können.) In der Praxis bieten sich jedoch mehrere Möglichkeiten an, die Speicherung der Exponenten fast völlig einzusparen. In jedem Fall muß jedoch zumindest ein zufälliger Exponent r_1 gespeichert sein, welcher als Initialwert (Seed) für eine pseudozufällige Folge $r_1, \ldots, r_k$ fungiert.

Eine Möglichkeit ist die Verwendung einer Hashfunktion. Für $i = 2, \ldots, k$ wird r_i durch $(i-1)$-faches Anwenden der Hashfunktion auf r_1 gebildet. Die Sicherheit dieses Verfahrens hängt natürlich von der gewählten Hashfunktion ab.

Im geheimen Fall können dagegen für $i = 2, \ldots, k$ die untersten (oder obersten) $\lceil \log q \rceil$ Bits der Binärdarstellung von $g^{r_{i-1}}$ als r_i verwendet werden. In diesem Fall können alle r_i ohne Berechnungsaufwand direkt ausgelesen werden. Diese Prozedur der wiederholten Exponentiation in $\mathbb{Z}_p^*$ oder in elliptischen Kurven ist die Basis von sicheren Pseudozufallsgeneratoren (siehe [BM82],[Kal87]).

Der öffentliche Fall

Wir betrachten nun den öffentlichen Fall. Hier können wir nicht mehr garantieren, daß $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ nur mit Wahrscheinlichkeit $1/\mathrm{SEC} + 1/q$ gilt. Für bestimmte Initialvektoren $\boldsymbol{r}$ kann es Attacken $\boldsymbol{c}$ geben, für die diese Gleichung mit weit größerer Wahrscheinlichkeit gilt. Ein Angreifer kann nun versuchen, aus $\boldsymbol{r}$ eine solche Attacke zu ermitteln. Die in Abschnitt 4.2.3 beschriebene Gitterattacke ist ein Beispiel für eine solche Vorgehensweise. Wir benötigen also im öffentlichen Fall ein Sicherheitsmaß, das von $\boldsymbol{r}$ abhängt.

Definition 4.2 *Für* $\boldsymbol{r} \in \mathbb{Z}_q^k$ *und eine einfache Attacke* $\boldsymbol{c}$ *definieren wir die* öffentliche Erfolgswahrscheinlichkeit *von* $\boldsymbol{c}$ *bezüglich* $\boldsymbol{r}$ *als* $\mathrm{S}(\boldsymbol{c}, \boldsymbol{r}) := \mathrm{Ws}\left[\langle \boldsymbol{c}\, , \boldsymbol{r}^* \rangle = 0 \bmod q\right]$. *Für* $\boldsymbol{r} \in \mathbb{Z}_q^k$ *sei die* öffentliche Güte des Generators *definiert durch* $\mathrm{SEC_p}(\mathbf{r}) := \min\left\{\mathrm{S}(\boldsymbol{c}, \boldsymbol{r})^{-1} \mid \boldsymbol{c} \in \mathbb{Z}^{n+1},\, n \in \mathbb{N}\right\}$.

Analog zu Satz 4.1 erhalten wir aus der öffentlichen Güte des Generators eine untere Schranke für die Sicherheit generische Angriffe im öffentlichen Fall.

Satz 4.5 *Für festes* $\boldsymbol{r}$ *sei* $\mathcal{A}$ *ein generischer Algorithmus der Länge* t*, der für zufällige* $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathcal{C}$ *die Ausgaben* $x_1^*, \ldots, x_n^*$ *des Generators als Eingabe erhält und eine einfache Attacke* $\boldsymbol{c}$ *mit* $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ *ausgibt. Dann hat* $\mathcal{A}$ *höchstens die Erfolgswahrscheinlichkeit* $t^2/(2\mathrm{SEC_p}(\mathbf{r}))$.

Der Beweis verläuft analog zu dem von Satz 4.1. Man beachte, daß $\mathcal{A}$ die Werte $g^{r_1}, \ldots, g^{r_k}$ nicht als Eingabe benötigt, da er in beliebiger Weise von $\boldsymbol{r}$ abhängen darf. Satz 4.5 deckt somit sogar Algorithmen ab, die mit nicht–generischen Methoden (wie z.B. Gitterbasenreduktion) auf $\boldsymbol{r}$ operieren.

Der folgende Satz liefert uns eine untere Schranke für die Güte für fast alle $\boldsymbol{r}$, sofern $\binom{k\ell}{A} \ll q$ gilt.

Satz 4.6 *Sei* $1 \leq \eta \leq k$ *und* $0 \leq \lambda \leq \ell$*. Dann besitzt der Generator für ein zufällig gewähltes* $\boldsymbol{r}$ *mindestens mit Wahrscheinlichkeit* $\binom{k\ell}{A}^{\eta+1}/q^\eta + 2^{k(\lambda+1)}/q$ *die öffentliche Güte*

$$\mathrm{SEC_p}(\mathbf{r}) < 2^{-(\eta-1)(\ell-\lambda+1)} \binom{k\ell}{A} \;.$$

Beweis. Unser Beweis verläuft in drei Schritten: Im ersten Schritt schätzen wir die Wahrscheinlichkeit ab, daß es (1) in $\mathbb{Z}_q^k$ eine von Elementen aus $\mathcal{C}$ aufgespannte, senkrecht[4] zu $\boldsymbol{r}$

[4] bzgl. des Skalarproduktes modulo q

stehende affine Ebene der Dimension η gibt. Im zweiten Schritt geben wir eine Abschätzung für die Wahrscheinlichkeit, daß es (2) zwei Elemente $\boldsymbol{u} \neq \boldsymbol{u}' \in \mathcal{C}$ mit $\|\boldsymbol{u} - \boldsymbol{u}'\|_\infty < 2^\lambda$ gibt, die auf derselben, senkrecht zu $\boldsymbol{r}$ stehenden Ebene liegen. Schließlich zeigen wir im dritten Schritt, daß $\mathrm{SEC_p}(\mathbf{r}) < 2^{-(\eta-1)(\ell-\lambda+1)} \binom{k\ell}{A}$ gilt, sofern wenn die beiden Bedingungen (1) und (2) nicht erfüllt sind.

Falls (1) gilt, dann gibt es $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{\eta+1} \in \mathcal{C}$, so daß $\boldsymbol{u}_1 - \boldsymbol{u}_2, \dots, \boldsymbol{u}_1 - \boldsymbol{u}_{\eta+1}$ linear unabhängig sind und

$$\langle \boldsymbol{u}_1 - \boldsymbol{u}_2, \boldsymbol{r} \rangle = \cdots = \langle \boldsymbol{u}_1 - \boldsymbol{u}_{\eta+1}, \boldsymbol{r} \rangle = 0 \bmod q$$

gilt. Für jedes Tupel $(\boldsymbol{u}_1, \dots, \boldsymbol{u}_{\eta+1})$ und zufälliges $\boldsymbol{r}$ gilt das höchstens mit Wahrscheinlichkeit $q^{-\eta}$. Da auf der anderen Seite die Anzahl dieser Tupel durch $\binom{k\ell}{A}^{\eta+1}$ beschränkt ist, gilt (1) höchstens mit Wahrscheinlichkeit $\binom{k\ell}{A}^{\eta+1}/q^\eta$.

Falls (2) gilt, dann gibt es ein Paar $\boldsymbol{u} \neq \boldsymbol{u}' \in \mathcal{C}$ mit $\|\boldsymbol{u} - \boldsymbol{u}'\|_\infty < 2^\lambda$ und $\langle \boldsymbol{u} - \boldsymbol{u}', \boldsymbol{r} \rangle = 0 \bmod q$. Für jeden nichttrivialen Vektor $\boldsymbol{x} \in \mathbb{Z}^k$ gilt $\langle \boldsymbol{x}, \boldsymbol{r} \rangle = 0 \bmod q$ mit Wahrscheinlichkeit $1/q$. Da auf der anderen Seite die Anzahl der Vektoren $\boldsymbol{x} \in \mathbb{Z}^k$ mit $\|\boldsymbol{x}\|_\infty < 2^\lambda$ durch $2^{k(\lambda+1)}$ beschränkt ist, gilt (2) höchstens mit Wahrscheinlichkeit $2^{k(\lambda+1)}/q$.

Wir nehmen nun an, daß (1) und (2) nicht gelten. Sei $\boldsymbol{c} \neq \mathbf{0} \bmod q$ eine einfache Attacke. Es reicht nun aus, $\mathrm{S}(\boldsymbol{c}, \boldsymbol{r}) \leq 2^{(\eta-1)(\ell-\lambda+1)}$ zu zeigen.

Ohne Einschränkung können wir $c_n = 1$ annehmen. Wir fixieren nun $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{n-1}$ so, daß

$$r_n^* \;=\; c_0 - \sum_{\nu=1}^{n-1} c_\nu r_\nu^* \;=:\; a \quad \bmod q$$

für zufälliges $\boldsymbol{u}_n$ mindestens mit Wahrscheinlichkeit $\mathrm{S}(\boldsymbol{c}, \boldsymbol{r})$ gilt. Wegen $r_n^* = \langle \boldsymbol{u}_n, \boldsymbol{r} \rangle$ gibt es dann $\boldsymbol{u}_n^1, \dots, \boldsymbol{u}_n^d \in \mathcal{C}$ mit $d \geq \mathrm{S}(\boldsymbol{c}, \boldsymbol{r})|\mathcal{C}|$ und $\langle \boldsymbol{u}_n^i, \boldsymbol{r} \rangle = a \bmod q$ für $i = 1, \dots, d$.

Da (1) nicht gilt, liegen alle $\boldsymbol{u}_n^1 - \boldsymbol{u}_n^i$ auf einer $(\eta - 1)$-dimensionalen Ebene. Da auch (2) nicht erfüllt ist, folgt

$$\|(\boldsymbol{u}_n^1 - \boldsymbol{u}_n^i) - (\boldsymbol{u}_n^1 - \boldsymbol{u}_n^j)\|_\infty = \|\boldsymbol{u}_n^j - \boldsymbol{u}_n^i\|_\infty \geq 2^\lambda$$

für alle $i \neq j$.

Auf der anderen Seite gilt $\|\boldsymbol{u}_n^1 - \boldsymbol{u}_n^i\|_\infty < 2^\ell$ für alle $i = 2, \dots, d$, d.h. alle Vektoren $\boldsymbol{u}_n^1 - \boldsymbol{u}_n^i$ liegen im Kubus $(-2^\ell, 2^\ell)^k$. Damit erhalten wir $d \leq 2^{(\eta-1)(\ell-\lambda+1)}$.

□

Im Fall $|\mathcal{C}| = \binom{k\ell}{A} \approx q$ liefert uns dieser Satz leider keine Aussage über $\mathrm{SEC_p}(\mathbf{r})$. Die folgende heuristische Überlegung legt jedoch die Vermutung nahe, daß in diesem Fall $\mathrm{SEC_p}(\mathbf{r})$ nur um einen logarithmischen Faktor von $\binom{k\ell}{A}$ abweicht.

Heuristik. Die Werte $(\langle \boldsymbol{u}_1, \boldsymbol{r}\rangle \bmod q), \ldots, (\langle \boldsymbol{u}_{|\mathcal{C}|}, \boldsymbol{r}\rangle \bmod q)$ mit $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{|\mathcal{C}|}\} = \mathcal{C}$ sind bei zufälliger Wahl von $\boldsymbol{r} \in \mathbb{Z}_q^k$ alle gleichverteilt in $\mathbb{Z}_q$. Wenn wir vereinfachend annehmen, daß sie darüber hinaus auch statistisch unabhängig sind, so können wir folgern, daß höchstens mit Wahrscheinlichkeit $1/|\mathcal{C}|$ mehr als $e \ln q / \ln \ln q$ viele von ihnen den gleichen Wert annehmen (siehe [MR95]). Damit würde dann auch $\mathrm{SEC_p}(\mathbf{r}) < |\mathcal{C}| \ln \ln q / (e \ln q)$ höchstens mit Wahrscheinlichkeit $1/|\mathcal{C}|$ gelten.

Unsere vereinfachende Annahme ist natürlich nicht korrekt. Die Menge $\mathcal{C}$ der möglichen $\boldsymbol{u}$ ist keine zufällige Untermenge von $\mathbb{Z}_q^k$ mit Mächtigkeit $\binom{k\ell}{A}$, sondern besitzt eine feste Struktur. Zum einen ist sie im Kubus $[0, 2^\ell)^k$ enthalten, zum anderen häufen sich im Fall $A < k$ die Punkte auf den inneren Kanten der Dimension $A \le d \le k-1$. Da aber ein zufälliges $\boldsymbol{r}$ fast immer „schief“ zu den kartesischen Koordinaten steht, spricht diese an den Einheitsvektoren ausgerichtete Struktur von $\mathcal{C}$ dafür, daß sich die Punkte aus $\mathcal{C}$ mit großer Wahrscheinlichkeit gleichmäßig auf die zu $\boldsymbol{r}$ senkrechten affinen Hyperebene verteilen und damit auch die Werte $\langle \boldsymbol{u}, \boldsymbol{r}\rangle \bmod q$ gleichmäßig verteilt sind.

4.2.2 Die statistische Verteilung der Ausgaben

Wir untersuchen nun die statistischen Eigenschaften der Verteilung, welche durch die Ausgaben des Generators erzeugt wird.

Definition 4.3 *Die Zufallsvariablen $Z_1, \ldots, Z_m$ auf einer Menge M heißen* statistisch unabhängig, *wenn für alle $a_1, \ldots, a_m \in M$ gilt*

$$\mathrm{Ws}[\, Z_1 = a_1, \, \ldots, \, Z_m = a_m \,] = \prod_{i=1}^{m} \mathrm{Ws}[\, Z_i = a_i \,] \ .$$

Sei $h : \{0,1\}^{a(n)} \times \{0,1\}^{b(n)} \to \{0,1\}^{c(n)}$ eine (nach n parametrisierte) Familie von Funktionen. Wir nennen h eine universelle Hashfunktion, *wenn $h(y,x)$ und $h(y,x')$ für alle n und für alle $x, x' \in \{0,1\}^{b(n)}$ mit $x \neq x'$ gleichverteilt und statistisch unabhängig sind, d.h. für alle $a, a' \in \{0,1\}^{c(n)}$*

$$\mathrm{Ws}_y[\, h(y,x) = a, \; h(y,x') = a' \,] = 2^{-2c(n)}$$

gilt.

Satz 4.7 *Für $\ell = 1$ ist die durch den Generator definierte Funktionenfamilie $G : \mathbb{Z}_q^k \times \mathcal{C} \to \mathbb{Z}_q$, $G(\boldsymbol{r}, \boldsymbol{u}) := \langle \boldsymbol{r}, \boldsymbol{u}\rangle \bmod q$ eine universelle Hashfunktion.*

Beweis. Seien $a, a' \in \mathbb{Z}_q$ und $\boldsymbol{u}, \boldsymbol{u}' \in \mathcal{C}$ mit $\boldsymbol{u} \neq \boldsymbol{u}'$. Dann gibt es j_1, j_2 mit $u_{j_1} = 1, u'_{j_1} = 0$ und $u_{j_2} = 0, u'_{j_2} = 1$. Für fixierte $r_i, \; i \neq j_1, j_2$ und zufällige r_{j_1}, r_{j_2} variieren $G(\boldsymbol{r}, \boldsymbol{u})$ und

$G(\boldsymbol{r}, \boldsymbol{u}')$ unabhängig über ganz $\mathbb{Z}_q$. Es gilt deshalb $\mathrm{Ws}_{\boldsymbol{r}}[\, G(\boldsymbol{r}, \boldsymbol{u}) = a \wedge G(\boldsymbol{r}, \boldsymbol{u}') = a' \,] = q^{-2}$. □

Für $\ell > 1$ gilt Satz 4.7 nicht mehr, denn z. B. für $\boldsymbol{u} = 2\boldsymbol{u}'$ sind die Ausgaben des Generators statistisch abhängig.

Im folgenden wollen wir abschätzen, wie gleichmäßig die Ausgaben des Generators verteilt sind. Hierfür benötigen wir zunächst eine Definition.

Definition 4.4 *Der* statistische Abstand *zwischen zwei Verteilungen $\mathcal{D}_n$ und $\mathcal{E}_n$ über $\{0,1\}^n$ ist definiert durch*

$$\mathrm{dist}(\mathcal{D}_n, \mathcal{E}_n) = \frac{1}{2} \sum_{z \in \{0,1\}^n} \left| \mathrm{Ws}_{\mathcal{D}_n}[z] - \mathrm{Ws}_{\mathcal{E}_n}[z] \right|.$$

*Wir sagen $\mathcal{D}_n$ und $\mathcal{E}_n$ sind höchstens $\epsilon(n)$–*statistisch unterscheidbar*, falls* $\mathrm{dist}(\mathcal{D}_n, \mathcal{E}_n) \leq \epsilon(n)$ *gilt.*

Der folgende Satz liefert uns ein ϵ, für das die Verteilung der Ausgaben des Generators ϵ–statistisch unterscheidbar zur Gleichverteilung ist. Der Beweis ist eine einfache Adaption des Beweises des Smoothing Entropy Theorems in [Lub96].

Satz 4.8 *Die Ausgaben $G(\boldsymbol{r}, \boldsymbol{u})$ des Generators sind höchstens $2^{-(e+1)}$–statistisch unterscheidbar von der Gleichverteilung falls $e \leq (\log|\mathcal{C}| - \log q)/2$.*

Beweis. Für alle $\boldsymbol{r} \in \mathbb{Z}_q^k$, $z \in \mathbb{Z}_q$ und $\boldsymbol{u} \in \mathcal{C}$ definieren wir $\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z) = 1$, falls $G(\boldsymbol{r}, \boldsymbol{u}) = z$ und 0 sonst. Wir wollen zeigen, daß

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{r}}\Big[\sum_{z \in \mathbb{Z}_q} |\mathbb{E}_{\boldsymbol{u}}[\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z)] - 1/q| \Big] \leq 2^{-(e+1)}$$

gilt. Dafür reicht es aus zu zeigen, daß

$$\mathbb{E}_{\boldsymbol{r}}\left[\, |\mathbb{E}_{\boldsymbol{u}}[\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z)] - 1/q| \,\right] \leq 2^{-e} q^{-1}$$

für alle $z \in \mathbb{Z}_q$ gilt.

Allgemein gilt $\mathbb{E}_Z(|Z|)^2 \leq \mathbb{E}_Z(Z^2)$ für eine Zufallsvariable Z. Setzen wir $Z = \mathbb{E}_{\boldsymbol{u}}[\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z)] - 1/q$, so sehen wir, daß es ausreicht, wenn wir für alle $z \in \mathbb{Z}_q$

$$\mathbb{E}_{\boldsymbol{r}}\left[(\mathbb{E}_{\boldsymbol{u}}[\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z)] - 1/q)^2 \right] \leq 2^{-2e} q^{-2}$$

zeigen. Durch elementare Umformungen und Vertauschen der Summationen können wir die linke Seite als

$$\mathbb{E}_{\boldsymbol{u}, \boldsymbol{u}'}\left[\mathbb{E}_{\boldsymbol{r}}[(\chi(G(\boldsymbol{r}, \boldsymbol{u}) = z)] - 1/q)(\chi(G(\boldsymbol{r}, \boldsymbol{u}') = z)] - 1/q)] \right]$$

schreiben. Zunächst beachte man, daß $\mathbb{E}_{\boldsymbol{r}}[(\chi(G(\boldsymbol{r},\boldsymbol{u})=z)=1/q$ für alle $z,\boldsymbol{r},\boldsymbol{u}$ gilt.

Wir schätzen nun den Term $\mathbb{E}_{\boldsymbol{r}}\left[(\chi(G(\boldsymbol{r},\boldsymbol{u})=z)]-1/q)(\chi(G(\boldsymbol{r},\boldsymbol{u}')=z)-1/q)\right]$ ab. Dafür unterscheiden wir 3 Fälle: Für linear unabhängige $\boldsymbol{u},\boldsymbol{u}'\in\mathcal{C}$ sind $G(\boldsymbol{r},\boldsymbol{u})$ und $G(\boldsymbol{r},\boldsymbol{u}')$ statistisch unabhängig und der Erwartungswert über $\boldsymbol{r}$ verschwindet. Für linear abhängige, aber verschiedene $\boldsymbol{u},\boldsymbol{u}'\in\mathcal{C}$ gilt $(\chi(G(\boldsymbol{r},\boldsymbol{u})=z)(\chi(G(\boldsymbol{r},\boldsymbol{u}')=z)=0$ für alle z, und der Erwartungswert über $\boldsymbol{r}$ ist negativ. Hingegen erhalten wir im Fall $\boldsymbol{u}=\boldsymbol{u}'$ durch Ausmultiplizieren des quadratischen Terms

$$\mathbb{E}_{\boldsymbol{r}}\left[(\chi(G(\boldsymbol{r},\boldsymbol{u})=z)]-1/q)^2\right]=q^{-1}(1-q^{-1})\leq 1/q.$$

Da $\mathrm{Ws}_{\boldsymbol{u},\boldsymbol{u}'}[\boldsymbol{u}=\boldsymbol{u}']=|\mathcal{C}|^{-1}$ gilt, ist die gesamte Summe höchstens $|\mathcal{C}|^{-1}q^{-1}=2^{-2e}q^{-2}$.

□

4.2.3 Angriffe gegen den Generator

Für den einfachen Generator mit $m=0$ sind die gezeigten unteren Schranken für die Sicherheit gegen generische Angriffe scharf. Wir zeigen dies durch die folgenden Angriffe. Die Angriffe lassen sich auch für Varianten des Generators mit $m>0$ verallgemeinern.

Generische Angriffe

Auf dem Geburtstags-Paradoxon basierende Angriffe. Diese Art von Angriffen läßt sich durch einen generischen Algorithmus beschreiben und funktioniert bereits im geheimen Fall. Der Angreifer wählt eine Folge $\boldsymbol{c}^1,\ldots,\boldsymbol{c}^t$ von einfachen Attacken. Er bildet eine Liste der Werte

$$\prod_\nu (x_\nu^*)^{c_\nu^i} \bmod p$$

und sortiert diese. Sind zwei dieser Werte gleich, so ist eine Relation der r_ν^* gefunden. Dies geschieht höchstens mit Wahrscheinlichkeit t^2/SEC.

Für $j=1,\ldots,t$ sei h_j das Hamminggewicht von $\boldsymbol{c}^j$. Dann benötigt der Angriff $\sum_j(h_j+\|\boldsymbol{c}^j\|_\infty)$ Multiplikationen modulo p und $t\log t$ viele Schritte zum Sortieren. Außerdem hat er einen Speicherbedarf von $t\log p$ Bits.

Im einfachsten Beispiel ist $\boldsymbol{c}^j$ der j-te Einheitsvektor. Eine Kollision der berechneten Werte tritt hier mit Wahrscheinlichkeit $t^2\binom{k\ell}{A}^{-1}$ auf. Für $n\geq t=\binom{k\ell}{A}^{1/2}$ ergibt sich die optimale Anzahl der notwendigen Multiplikationen von $\binom{k\ell}{A}^{1/2}$.

Meet-in-the-middle Attacken. Diese Attacken sind eine spezielle Form der Geburtstagsattacken und lassen sich somit auch durch generische Algorithmen beschreiben. Sie funktionieren im halböffentlichen und öffentlichen Fall.

Sei A gerade und ν fest gewählt. Der Angreifer bildet für alle $\boldsymbol{u}$ mit Hamminggewicht $A/2$ die Werte

$$F(\boldsymbol{u}) := \prod_{i=1}^{k} g^{r_i u_i} \bmod p$$

und bildet daraus eine Liste. Dann berechnet er für alle $\boldsymbol{u}$ mit Hamminggewicht $A/2$ die Werte

$$G(\boldsymbol{u}) := x_\nu^* \Big(\prod_{i=1}^{k} g^{r_i u_i} \Big)^{-1} \bmod p$$

und bildet daraus eine Liste. Durch Sortieren der beiden Listen findet er dann mindestens eine Kollision $F(\boldsymbol{u}^1) = G(\boldsymbol{u}^2)$. Damit folgt $r_\nu^* = \langle \boldsymbol{u}^1 + \boldsymbol{u}^2 \, , \boldsymbol{r} \rangle \bmod q$.

Im öffentlichen Fall erhält der Angreifer damit schon eine Relation der r_ν^*. Im halböffentlichen Fall hat er zumindest $\boldsymbol{u}_\nu$ ermittelt. Er kann jetzt das Verfahren für verschiedene ν wiederholen und damit die entsprechenden $\boldsymbol{u}_\nu$ ermitteln. Nach höchstens k vielen Durchgängen findet er immer eine ganzzahlige Relation der $\boldsymbol{u}_\nu$. Diese liefert ihm dann auch eine Relation der r_ν^*.

Diese Attacke benötigt (für jeden Durchgang) $\binom{k\ell}{A/2}\ell A$ Multiplikationen modulo p und hat einen Speicherbedarf von $\binom{k\ell}{A/2} \log p$ Bits. Damit bewegt sich auch die Komplexität dieses Angriffes bei $\sqrt{\binom{k\ell}{A}}$.

Nicht-generische Angriffe

Die beiden folgenden Angriffe verwenden Algorithmen zur Gitterbasenreduktion und sind daher nicht generisch. Außerdem verwenden beide Angriffe die Informationen, die durch die Signaturgleichungen gegebenen Informationen, um eine lineare Gleichung in den r_ν^* zu bestimmen.

Ein Angriff mittels Gitterbasenreduktion. Der folgende Angriff funktioniert nur im öffentlichen Fall.

Aus $y_\nu = r_\nu^* + s e_\nu \bmod q$ und $r_\nu^* = \sum_{i=1}^{k} u_{\nu,i} r_i \bmod q$ für $\nu = 1, \ldots, n$ erhalten wir mit $E_\nu := \prod_{i \neq \nu} e_i \bmod q$ das Gleichungssystem

$$y_\nu E_\nu - y_1 E_1 = \sum_{i=1}^{k} u_{\nu,i} r_i E_\nu - \sum_{i=1}^{k} u_{1,i} r_i E_1 \bmod q \tag{4.5}$$

für $\nu = 2, \ldots, n$. Im Fall $n = 2$ folgt aus Korollar 2.5, daß für zufällige $r_1, \ldots, r_k$ mindestens mit Wahrscheinlichkeit $1 - 4(2k+1)^{\frac{1}{2}}\, 2^{2k(\ell+1.047\ldots)}/q$ der kürzeste Vektor des durch

$$\begin{pmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_k \\ \boldsymbol{b}_{k+1} \\ \vdots \\ \boldsymbol{b}_{2k} \\ \boldsymbol{b}_{2k+1} \end{pmatrix} = \begin{pmatrix} 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & 2^{\ell-1} & \cdots & 2^{\ell-1} & K(y_2E_2 - y_1E_1) \\ 0 & 1 & & & & & & Kr_1E_1 \\ \vdots & & \ddots & & & & & \vdots \\ 0 & & & 1 & & & & Kr_kE_1 \\ 0 & & & & 1 & & & Kr_1E_2 \\ \vdots & & & & & \ddots & & \vdots \\ 0 & & & & & & 1 & Kr_kE_2 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & Kq \end{pmatrix}$$

erzeugten Gitters die Vektoren $\boldsymbol{u}_1$ und $\boldsymbol{u}_2$ liefert. Im Fall $2^{2k\ell} \gg q$ enthält das Gitter jedoch viele linear unabhängige kurze Vektoren und der Angreifer kann $\boldsymbol{u}_1$ und $\boldsymbol{u}_2$ nicht bestimmen.

Die Nguyen-Stern-Methode. In [NS99] präsentierten Nguyen und Stern einen auf Gitterbasenreduktion basierenden Angriff auf das Verfahren von Boyko, Peinado und Venkatesan [BPV98]. Dieser Angriff läßt sich auch auf unseren Generator übertragen. Dabei übersetzt sich der Parameter n des Verfahrens [BPV98] im wesentlichen in die Größe $k\ell$. Da der Angriff nur für $n \leq 35$ praktikabel war, stellt er für unseren Generator im Fall $k\ell > 200$ keine Bedrohung dar.

4.3 Transformation der geheimen Präsignaturen

Im vorangegangenen Kapitel beruhte die Sicherheit des Generators auf den Zufallsbits einer einzigen Runde. Durch Verwendung von variablen Präsignaturen können wir die Sicherheit des Generators auf mehreren Runden basieren lassen. Dadurch erhalten wir wesentlich effizientere Generatoren.

4.3.1 Auf zwei Runden beruhende Sicherheit

Wir betrachten zunächst den einfachsten Fall, bei dem die Sicherheit des Generators auf den Zufallsbits von zwei Runden beruht. Im folgenden sei $m = 1$.

Generator 1

1. EINGABE $\boldsymbol{r} \in_R \mathbb{Z}_q^k$, $s_1 \in_R \mathbb{Z}_q$.
   $\nu := 1$.

2. Wähle ein zufälliges $\boldsymbol{u}_\nu \in \mathcal{C}$ und setze $r' := \sum_{i=1}^{k} u_{i,\nu} r_i \bmod q$.

3. $s_1 := 2^\ell s_1 \bmod q$.

4. AUSGABE $\quad r_\nu^* := r' + s_1$ (verwende $(r_\nu^*, g^{r_\nu^*})$ für die nächste Signatur)
   $\nu := \nu + 1$,
   GO TO 2 für die nächste Runde.

Das Verfahren benötigt $A + 2\ell - 1$ Multiplikationen pro Ausgabe. Sei $s_m^{[0]}$ der Initialwert von s_1. Dann gilt $r_\nu^* = r_\nu' + 2^{\nu\ell} s_1^{[0]}$ für alle ν.

Für unsere Sicherheitsanalyse unterscheiden wir zwei Typen von einfachen Attacken.

Definition 4.5 *Für $N \in \mathbb{N}$ sei $Y(N)$ die Menge der $(a_1, a_2) \in \mathbb{Z}^2$ für die $a_1 x_1 + a_2 x_2 = 0 \bmod q$ eine nichttriviale Lösung $(x_1, x_2) \in \mathbb{Z}^2$ mit $|x_1|, |x_2| < N$ besitzt. Eine einfache Attacke $\boldsymbol{c}$ nennen wir* zahm, *falls $(c_i, c_j) \in Y(2^\ell)$ für alle $1 \le i < j \le n$ gilt. Anderenfalls nennen wir die Attacke* wild.

Wir zeigen zuerst eine obere Schranke für die Erfolgswahrscheinlichkeit wilder Attacken. Diese ist quadratisch besser als die triviale Schranke $\binom{k\ell}{A}$ aus dem vorangegangenen Kapitel.

Satz 4.9 *Jede wilde Attacke hat höchstens Erfolgswahrscheinlichkeit $\binom{k\ell}{A}^{-2}$.*

Beweis. Sei $(c_v, c_w) \notin Y(2^\ell)$. Aus $\langle \boldsymbol{c}\,, \boldsymbol{r}^* \rangle \equiv 0 \bmod q$ erhalten wir $\sum_\nu c_\nu \boldsymbol{u}_\nu = \boldsymbol{0} \bmod q$. Für fixierte $\boldsymbol{u}_\nu$, $\nu \ne v, w$ erhalten wir daraus $c_v \boldsymbol{u}_v + c_w \boldsymbol{u}_w = C \bmod q$ mit einer Konstanten C. Es reicht also zu zeigen, daß die Vektoren $c_v \boldsymbol{u}_v + c_w \boldsymbol{u}_w$ modulo q paarweise verschieden sind für alle Paare von Münzwürfen $(\boldsymbol{u}_v, \boldsymbol{u}_w) \in \mathcal{C}^2$.

Angenommen, es gilt $c_v \boldsymbol{u}_v + c_w \boldsymbol{u}_w = c_v \boldsymbol{u}_v' + c_w \boldsymbol{u}_w' \bmod q$ für $(\boldsymbol{u}_v, \boldsymbol{u}_w), (\boldsymbol{u}_v', \boldsymbol{u}_w') \in \mathcal{C}^2$ und $(\boldsymbol{u}_v, \boldsymbol{u}_w) \ne (\boldsymbol{u}_v', \boldsymbol{u}_w')$. Ohne Einschränkung können wir $(u_{1,v}, u_{1,w}) \ne (u_{1,v}', u_{1,w}')$ annehmen. Dann folgt $c_v x_1 + c_w x_2 = 0 \bmod q$ mit $\boldsymbol{x} := (u_{1,v}, u_{1,w}) - (u_{1,v}', u_{1,w}') \in (-2^\ell, 2^\ell)^2 \setminus \{\boldsymbol{0}\}$. Das widerspricht jedoch $(c_v, c_w) \notin Y(2^\ell)$.

□

Wir benötigen nun noch eine Schranke für die Erfolgswahrscheinlichkeit von zahmen Attacken. Leider finden wir eine solche nur für Attacken beschränkter Länge. Der Grund dafür liegt in einer linearen Gleichung, welche eine Attacke $\boldsymbol{c}$ mit $\mathrm{S}(\boldsymbol{c}) > 0$ erfüllen muß (siehe Beweis von Lemma 4.19). Diese gilt zunächst nur modulo q. Falls nun die Länge einer zahmen Attacke beschränkt ist, können wir folgern, daß diese Gleichung auch über $\mathbb{Z}$ gilt und finden einen Widerspruch. Für große n können wir diesen Weg leider nicht gehen; die Gleichungen gelten i.A. nur modulo q und wir finden keinen Widerspruch.

Definition 4.6 *Für $a, b \in \mathbb{Z}$, $b \ne 0 \bmod q$ sei $(a)_q$ der Repräsentant von $a \bmod q$ in $(-q/2, q/2)$ und $(a/b)_q$ der Repräsentant von $a/b \bmod q$ in $(-q/2, q/2)$.*

Für $n, \lambda \in \mathbb{N}$ definieren wir $\mathcal{L}(n, \lambda)$ als das Minimum von $\mathrm{kgV}(2, 3, \ldots, 2^\lambda - 1)$ *und* $2^{n\lambda}$.

Lemma 4.10 *Sei* $\boldsymbol{c} \in \mathbb{Z}_q^{n+1}$ *eine zahme Attacke mit* $2^{(n-1)\ell+1}\mathcal{L}(n-1,\ell) < q$*. Dann gilt* $\mathrm{S}(\boldsymbol{c}) = 0$.

Beweis. Die Attacke $\boldsymbol{c}'$ mit $c'_\nu := (c_\nu/c_n)_q$ hat die gleiche Erfolgswahrscheinlichkeit wie $\boldsymbol{c}$. Wir können also $c_n = 1$ annehmen. Da $\boldsymbol{c}$ zahm ist, folgt $c_\nu = \pm(a_\nu/b_\nu)_q$ mit $1 \leq a_\nu, b_\nu \leq 2^\ell - 1$ für alle $c_\nu \neq 0$ gilt. Falls nun $\mathrm{S}(\boldsymbol{c}) > 0$ gilt, so gibt es $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$, so daß $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ unabhängig von $\boldsymbol{s}$ gilt. Wegen $r^*_\nu = r'_\nu + 2^{\nu\ell} s_1^{[0]}$ folgt daraus

$$\sum_{\nu=1}^{n} c_\nu 2^{(\nu-1)\ell} = 0 \bmod q \ . \tag{4.6}$$

Sei $D := \mathrm{kgV}(b_1, \ldots, b_{n-1})$. Einerseits gilt $D \leq 2^{(n-1)\ell}$, andererseits ist D höchstens so groß wie $\mathrm{kgV}(2, 3, \ldots, 2^\ell - 1)$. Wir erhalten damit $D \leq \mathcal{L}(n-1, \ell)$.

Wir setzen nun $c'_\nu := (Dc_\nu)_q$. Ebenso wie $\boldsymbol{c}$ erfüllt auch $\boldsymbol{c}'$ die Gleichung (4.6). Andererseits ist $\sum_{\nu=1}^n c'_\nu 2^{(\nu-1)\ell}$ wegen $c'_n = D$ und $|c'_\nu| = a_\nu D/b_\nu < D2^\ell$ durch $D2^{(n-1)\ell+1} < q$ beschränkt. Damit gilt (4.6) für $\boldsymbol{c}'$ auch über $\mathbb{Z}$. Wir zeigen nun, daß dies nicht gelten kann.

Einerseits gilt $|c'_n 2^{\ell(n-1)}| = D2^{(n-1)\ell}$. Andererseits gilt wegen $|c'_\nu| \leq D(2^\ell - 1)$

$$\begin{aligned}
\Big|\sum_{\nu=1}^{n-1} c'_\nu 2^{(\nu-1)\ell}\Big| &\leq \sum_{\nu=1}^{n-1} |c'_\nu| 2^{(\nu-1)\ell} \\
&\leq \sum_{\nu=1}^{n-1} D(2^\ell - 1) 2^{(\nu-1)\ell} \\
&= D(2^{(n-1)\ell} - 1) \ .
\end{aligned}$$

Das ist ein Widerspruch.

□

Wir erhalten nun unmittelbar eine obere Schranke für die Erfolgswahrscheinlichkeit einfacher Attacken beschränkter Länge.

Korollar 4.11 *Sei* $2^{(n-1)\ell+1}\mathcal{L}(n-1,\ell) < q$*. Dann hat jede einfache Attacke* $\boldsymbol{c} \in \mathbb{Z}^{n+1}$ *höchstens Erfolgswahrscheinlichkeit* $\binom{k\ell}{A}^{-2}$*.*

Resultate. Tabelle 4.3 vergleicht Generator 2 mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden. Die angegebenen Parameter liefern eine Sicherheit von ungefähr 2^{160} gegen Attacken beschränkter Länge. *Mem* bezeichnet die Anzahl der zu speichernden Gruppenelemente und *Mult* die Anzahl der Multiplikationen pro Ausgabe.

Wir können die Existenz effizienter zahmer Attacken nicht ausschließen. Aus dem Beweis von Satz 4.10 folgt zwar, daß eine solche von dem Modul q abhängen muß; es ist

Mem	k, ℓ, A	$Mult$	Prec.-Methode	$Mult$
8	7,15,25	54	de Rooij	59
16	15,8,22	37	de Rooij	52
30	29,5,20	29	Lim-Lee	55.5
60	59,4,16	23	Lim-Lee	45.5
124	123,3,14	19	Lim-Lee	37

Tabelle 4.3: Vergleich von Generator 2 mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

jedoch denkbar, daß ein Angreifer mittels Gitterbasenreduktion in der Lage ist, effiziente zahme Attacken zu finden. Ein solcher Ansatz wurde von J. Stern in [Ste95] verfolgt. Seine heuristische Analyse liefert jedoch keine bessere Erfolgswahrscheinlichkeit als unsere obere Schranke $\binom{k\ell}{A}^{-2}$.

In Abschnitt 4.3.4 werden wir für eine Variante dieses Generators eine obere Schranke für die Erfolgswahrscheinlichkeit aller einfacher Attacken beweisen.

4.3.2 Auf drei Runden beruhende Sicherheit

Durch Verwendung einer zusätzlichen variablen Präsignatur können wir die Sicherheit des Generators auf den Zufallsbits von drei Runden beruhen lassen.

Generator 2

1. EINGABE $\boldsymbol{r} \in_R \mathbb{Z}_q^k$, $s_1, s_2 \in_R \mathbb{Z}_q$.
   $\nu := 1$.
2. Wähle ein zufälliges $\boldsymbol{u}_\nu \in \mathcal{C}$ und setze $r' := \sum_{i=1}^k u_{i,\nu} r_i \bmod q$.
3. $s_1 := 2^{\ell+1} s_1 \bmod q\,,\;\; s_2 := 2^{\ell+1}(s_2 + s_1) \bmod q$.
4. AUSGABE $r_\nu^* := r' + s_2$ (verwende $(r_\nu^*, g^{r_\nu^*})$ für die nächste Signatur)
   $\nu := \nu + 1$,
   GO TO 2 für die nächste Runde.

Der Generator benötigt $A + 3\ell + 2$ Multiplikationen pro Ausgabe. Es gilt

$$r_\nu^* = r'_\nu + \nu 2^{(\nu+1)(\ell+1)} s_1^{[0]} + 2^{\nu(\ell+1)} s_2^{[0]} \bmod q \;\;, \tag{4.7}$$

wobei $s_1^{[0]}$ und $s_2^{[0]}$ die Initialwerte von s_1 bzw. s_2 sind.

Analog zu Abschnitt 4.3.1 unterscheiden wir zwei Typen von einfachen Attacken.

Definition 4.7 *Für $N \in \mathbb{N}$ sei $Y_3(N)$ die Menge der $(a_1, a_2, a_3) \in \mathbb{Z}^3$ für die $a_1x_1 + a_2x_2 + a_3x_3 = 0 \bmod q$ eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < N$ besitzt. Eine einfache Attacke $\boldsymbol{c}$ nennen wir* 3-fach zahm, *falls $(c_{i_1}, c_{i_2}, c_{i_3}) \in Y_3(2^\ell)$ für alle $1 \leq i_1 < i_2 < i_3 \leq n$ gilt. Anderenfalls nennen wir die Attacke* 3-fach wild.

Der folgende Satz zeigt, daß die Sicherheit von Generator 2 gegen Attacken beschränkter Länge auf den Zufallsbits von drei Runden beruht.

Satz 4.12 *Sei $n\mathcal{L}(n-2, \ell)2^{n(\ell+1)+1} < \sqrt{q}$. Dann gilt $\mathrm{S}(\boldsymbol{c}) \leq \binom{k\ell}{A}^{-3}$ für jede einfache Attacke $\boldsymbol{c} \in \mathbb{Z}^{n+1}$.*

Wir skizzieren kurz den Beweis. Zunächst folgt unmittelbar aus (4.7), daß jede einfache Attacke $\boldsymbol{c}$ mit Erfolgswahrscheinlichkeit größer 0 die Gleichungen

$$\sum_{\nu=1}^{n} c_\nu 2^{(\nu-1)(\ell+1)} = 0 \bmod q \tag{4.8}$$

und

$$\sum_{\nu=1}^{n} c_\nu \nu 2^{(\nu-1)(\ell+1)} = 0 \bmod q \tag{4.9}$$

erfüllen muß. Wir zeigen zunächst in Satz 4.13, daß jede 3-fach wilde Attacke höchstens Erfolgswahrscheinlichkeit $\binom{k\ell}{A}^{-3}$ besitzt. Dann zeigen wir in Lemma 4.14, daß es für jede Attacke, welche (4.8) und (4.9) über $\mathbb{Z}$ erfüllt, Indices $1 \leq i_1 < i_2 < i_3 \leq n$ gibt, so daß $a_1x_1 + a_2x_2 + a_3 = 0$ (über $\mathbb{Z}$) keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt. Daraus folgern wir dann in Satz 4.16, daß jede 3-fach zahme Attacke beschränkter Länge Erfolgswahrscheinlichkeit 0 hat.

Satz 4.13 *Jede 3-fach wilde Attacke $\boldsymbol{c}$ besitzt höchstens Erfolgswahrscheinlichkeit $\binom{k\ell}{A}^{-3}$.*

Beweis. Sei $(c_{i_1}, c_{i_2}, c_{i_3}) \notin Y_3(2^\ell)$. Wir zeigen, daß $c_{i_1}\boldsymbol{u}_{i_1} + c_{i_2}\boldsymbol{u}_{i_2} + c_{i_3}\boldsymbol{u}_{i_3} \bmod q$ für verschiedene Tupel $(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}) \in \mathcal{C}^3$ verschieden Werte annimmt.

Angenommen es gilt

$$c_{i_1}\boldsymbol{u}_{i_1} + c_{i_2}\boldsymbol{u}_{i_2} + c_{i_3}\boldsymbol{u}_{i_3} = c_{i_1}\boldsymbol{u'}_{i_1} + c_{i_2}\boldsymbol{u'}_{i_2} + c_{i_3}\boldsymbol{u'}_{i_3}$$

für

$$(\boldsymbol{u}_{i_1}, \boldsymbol{u}_{i_2}, \boldsymbol{u}_{i_3}) \neq (\boldsymbol{u'}_{i_1}, \boldsymbol{u'}_{i_2}, \boldsymbol{u'}_{i_3}) \in \mathcal{C}^3 \quad .$$

Wir können ohne Einschränkung $(u_{1,i_1}, u_{1,i_2}, u_{1,i_3}) \neq (u'_{1,i_1}, u'_{1,i_2}, u'_{1,i_3})$ annehmen. Dann ist $\boldsymbol{x} := (u_{1,i_1}, u_{1,i_2}, u_{1,i_3}) - (u'_{1,i_1}, u'_{1,i_2}, u'_{1,i_3})$ eine nichttriviale Lösung von $c_{i_1}x_1 + c_{i_2}x_2 + c_{i_3}x_3 =$

0 mod q. Wegen $\|\boldsymbol{x}\|_\infty < 2^\ell$ ist das ein Widerspruch zu $(c_{i_1}, c_{i_2}, c_{i_3}) \notin Y_3(2^\ell)$.

□

Im folgenden sei L^1 das Gitter der Vektoren in $\mathbb{Z}^n$, die (4.8) über $\mathbb{Z}$ erfüllen und L^2 das Gitter der Vektoren in $\mathbb{Z}^n$, die (4.8) und (4.9) über $\mathbb{Z}$ erfüllen. Wir können explizite Basen für L^1 und L^2 angeben: Man prüft leicht nach, daß die Vektoren $\boldsymbol{b}_i^1 := \boldsymbol{e}_{i+1} - 2^{\ell+1}\boldsymbol{e}_i$ für $i = 1, \ldots, n-1$ eine Basis von L^1 und die Vektoren $\boldsymbol{b}_i^2 := \boldsymbol{b}_{i+1}^1 - 2^{\ell+1}\boldsymbol{b}_i^1$ für $i = 1, \ldots, n-2$ eine Basis von L^2 bilden, wobei $\boldsymbol{e}_i$ den i-ten Einheitsvektor bezeichnet. Es folgt daraus, daß es für jedes $\boldsymbol{z} \in L^2$ ein $\boldsymbol{y} \in L^1$ mit $y_n = 0$ gibt, so daß $z_i = y_{i-1} - 2^\ell y_i$ mit $y_0 := 0$ für $i = 1, \ldots, n$ gilt.

Lemma 4.14 *Sei $\boldsymbol{d} \in L^2$. Dann gibt es $1 \le i_1 < i_2 < i_3 \le n$, so daß die Gleichung $d_{i_1}x_1 + d_{i_2}x_2 + d_{i_3}x_3 = 0$ keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt.*

Beweis. Wir zeigen zunächst, daß eine Gleichung $d_{i_1}x_1 + d_{i_2}x_2 + d_{i_3}x_3 = 0$ keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt, sofern

$$|d_{i_1}| \ge \frac{(2^{\ell+1}-1)^2}{2^{\ell+2}-1}|d_{i_2}| \tag{4.10}$$

und

$$|d_{i_1}| > (2^{\ell+1}-1)^2|d_{i_3}| \tag{4.11}$$

gilt.

Angenommen es gilt $d_{i_1}x_1 + d_{i_2}x_2 + d_{i_3}x_3 = 0$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$. Auf der einen Seite gilt $|d_{i_2}x_2 + d_{i_3}x_2| \le (2^\ell - 1)(|d_{i_2}| + |d_{i_3}|)$. Falls (4.10) und (4.11) gelten, ist das kleiner als

$$|d_{i_1}|(2^\ell - 1)\left(\frac{2^{\ell+2}-1}{(2^{\ell+1}-1)^2} + \frac{1}{(2^{\ell+1}-1)^2}\right) \le |d_{i_1}| \ .$$

Das widerspricht jedoch $d_{i_1}x_1 + d_{i_2}x_2 + d_{i_3}x_3 = 0$.

Wir zeigen nun, daß es Indices i_1, i_2, i_3 gibt, die (4.10) und (4.11) erfüllen.

Wegen $\boldsymbol{d} \in L^2$ gibt es ein $\boldsymbol{c} \in L^1$ so, daß $c_n = 0$ und $d_\nu = c_{\nu-1} - 2^{\ell+1}c_\nu$ mit $c_0 := 0$ für $\nu = 1, \ldots, n$ gilt. Ohne Einschränkung können wir $c_{n-1} > 0$ annehmen. Wir setzen i_1 als das j, für das $|c_j|$ maximal ist, i_2 als das minimale ν, so daß $c_\nu \ge c_{\nu+1} \ge \cdots \ge c_{n-1}$ gilt, und $i_3 := n$.

Wir zeigen zunächst, daß i_1, i_2, i_3 (4.10) und (4.11) erfüllen, sofern

$$|c_{i_1}| > (2^{\ell+1}-1)c_{i_2} \tag{4.12}$$

gilt. Danach beweisen wir (4.12).

Nach Konstruktion gilt $c_{i_2-1} < c_{i_2}$ oder $c_{i_2-1} < 0$ und daher $|d_{i_2}| \leq |c_{i_2-1}| + 2^{\ell+1}c_{i_2}$. Wegen $|c_{i_2-1}| \leq |c_{i_1}|$ und (4.12) erhalten wir daraus

$$|d_{i_2}| = \frac{2^{\ell+2}-1}{2^{\ell+1}-1}|c_{i_1}| \ . \tag{4.13}$$

Außerdem können wir

$$|d_{i_1}| \ \geq \ 2^{\ell+1}|c_{i_1}| - |c_{i_1-1}| \ \geq \ (2^{\ell+1}-1)|c_{i_1}| \tag{4.14}$$

abschätzen.

Auf der anderen Seite gilt nach Konstruktion $c_{i_2} \geq c_{n-1} = d_{i_3}$ und mit (4.12) erhalten wir

$$|c_{i_1}| \ \geq \ (2^{\ell+1}-1)c_{i_2} \ \geq \ (2^{\ell+1}-1)|d_{i_3}| \ .$$

Wir erhalten nun (4.10) durch (4.14) und (4.11) durch (4.13).

Wir zeigen nun (4.12). Angenommen, es gilt $|c_\nu| \leq (2^{\ell+1}-1)c_{i_2}$ für alle ν. Dann folgt

$$\begin{aligned} \Big|\sum_{\nu=1}^{i_2-1} c_\nu 2^{(\nu-1)(\ell+1)}\Big| \ &\leq \ \sum_{\nu=1}^{i_2-1} |c_\nu| 2^{(\nu-1)(\ell+1)} \\ &\leq \ \sum_{\nu=1}^{i_2-1} c_{i_2}(2^{\ell+1}-1)2^{(\nu-1)(\ell+1)} \\ &= \ c_{i_2}(2^{(i_2-1)(\ell+1)}-1) \ . \end{aligned}$$

Auf der anderen Seite gilt $\sum_{\nu=i_2}^{n-1} c_\nu 2^{(\nu-1)(\ell+1)} \geq c_{i_2}2^{(i_2-1)(\ell+1)}$. Das ist ein Widerspruch zu $\boldsymbol{c} \in L^1$. Somit folgt $|c_{i_1}| > (2^{\ell+1}-1)c_{i_2}$.

□

Wir beweisen nun Satz 4.12, indem wir zeigen, daß jede 3-fach zahme Attacke $\boldsymbol{c}$ beschränkter Länge, welche (4.8) und (4.9) erfüllt, in L^2 liegt, d.h. diese Gleichungen auch über $\mathbb{Z}$ erfüllt. Damit folgt nach Lemma 4.19, daß es Indices i_1, i_2, i_3 gibt, so daß $c_{i_1}x_1 + c_{i_2}x_2 + c_{i_3}x_3 = 0$ keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt. Wir werden dann folgern, daß auch modulo q keine solche Lösung existiert, weil die Länge von $\boldsymbol{c}$ beschränkt ist. Dazu müssen wir jedoch zuvor die Attacke $\boldsymbol{c}$ normieren, d.h. so mit einer Konstanten modulo q multiplizieren, daß $c_{i_1}, c_{i_2}, c_{i_3}$ klein werden. Das folgende Lemma zeigt, daß wir dies tun können.

Lemma 4.15 *Seien $z_1, \ldots, z_n \in \mathbb{Z}$. Dann gibt es ein $\alpha \in \mathbb{Z}$, so daß $|(\alpha z_i)_q| < q^{1-1/n}$ für alle $i \leq n$ gilt.*

Beweis. Ohne Einschränkung können wir $z_n = 1$ annehmen. Wir betrachten das Gitter $L := \{\boldsymbol{x} \in \mathbb{Z}^n \mid \exists \alpha \in \mathbb{Z} : \boldsymbol{x} = \alpha \boldsymbol{z} \bmod q\}$.

Sei $\boldsymbol{x} = \alpha \boldsymbol{z} \bmod q$. Dann gilt für alle $1 \leq i \leq n-1$

$$(\boldsymbol{x} - x_n \boldsymbol{z})_i = x_i - x_n z_i = \alpha z_i - \alpha z_n z_i = 0 \bmod q \ .$$

Damit ist $(q\boldsymbol{e}_1, \ldots, q\boldsymbol{e}_{n-1}, \boldsymbol{z})$ eine Basis von L, wobei $\boldsymbol{e}_i$ den i-ten Einheitsvektor bezeichnet. Wegen $z_n = 1$ gilt $\det(L) = q^{n-1}$ und mit dem Satz von Minkowski (Theorem 2.3) erhalten wir $\lambda_{1,\infty}(L) \leq q^{1-1/n}$. Damit ist das Lemma bewiesen.

□

Das folgende Lemma vollendet nun den Beweis von Satz 4.12.

Satz 4.16 *Sei* $n\mathcal{L}(n-2, \ell)2^{n(\ell+1)+1} < \sqrt{q}$*. Dann gilt* $\mathrm{S}(\boldsymbol{c}) = 0$ *für jede 3-fach zahme Attacke* $\boldsymbol{c} \in \mathbb{Z}^{n+1}$.

Beweis. Sei $\boldsymbol{c}$ eine einfache Attacke mit $\mathrm{S}(\boldsymbol{c}) > 0$. Falls $\boldsymbol{c}$ zahm ist, folgt $\mathrm{S}(\boldsymbol{c}) = 0$ analog zu Lemma 4.10. Wir können also annehmen, daß $\boldsymbol{c}$ wild ist, d.h. daß es $1 \leq j_1 < j_2 \leq n$ mit $(c_{j_1}, c_{j_2}) \notin Y(2^\ell)$ gibt. Insbesondere gilt $c_{j_1}, c_{j_2} \neq 0 \bmod q$. Wir zeigen, daß $\boldsymbol{c}$ auch 3-fach wild ist.

Angenommen, für jedes $1 \leq \nu \leq n$ gibt es eine nichttriviale Lösung $\boldsymbol{x}^\nu \in \mathbb{Z}^3$ von $c_{j_1}x_1 + c_{j_2}x_2 = c_\nu x_3 \bmod q$ mit $\|\boldsymbol{x}^\nu\|_\infty < 2^\ell$. Da $\boldsymbol{c}$ wild ist, kann x_3^ν nicht Null sein. (Für $\nu = i_1, i_2$ gilt $x_\nu^\nu = x_3^\nu = 1$.) Wir setzen $d_\nu := (D/x_3^\nu)(x_1^\nu c_{i_1} + x_2^\nu c_{i_2}) \in \mathbb{Z}$ mit $D := \mathrm{kgV}\,\{x_3^\nu \mid \nu \neq i_1, i_2\}$. Es folgt

$$d_\nu = D(x_1^\nu c_{i_1} + x_2^\nu c_{i_2})/x_3^\nu = Dc_\nu \bmod q \ ,$$

d.h. $\boldsymbol{d} = D\boldsymbol{c} \bmod q$ und damit $\mathrm{S}(\boldsymbol{d}) = \mathrm{S}(\boldsymbol{c}) > 0$. Nach Lemma 4.14 erfüllt $\boldsymbol{d}$ somit (4.8) und (4.9). Es reicht aus zu zeigen, daß $\boldsymbol{d}$ 3-fach wild ist. Wegen $\boldsymbol{d} = D\boldsymbol{c} \bmod q$ ist dann auch $\boldsymbol{c}$ 3-fach wild.

Andererseits können wir nach Lemma 4.15 ohne Einschränkung $|c_{j_1}|, |c_{j_2}| < \sqrt{q}$ annehmen. Es gilt daher $|d_\nu| \leq |D|(|x_1^\nu||c_{i_1}| + |x_2^\nu||c_{i_2}|) < \mathcal{L}(n-2,\ell)2^{\ell+1}\sqrt{q}$ füralle ν und damit

$$\begin{aligned} \left| \sum_{\nu=1}^{n} d_\nu \nu 2^{(\nu-1)(\ell+1)} \right| &< \mathcal{L}(n-2,\ell)2^{\ell+1}\sqrt{q} \sum_{\nu=1}^{n} \nu 2^{(\nu-1)(\ell+1)} \\ &\leq n\mathcal{L}(n-2,\ell)2^{n(\ell+1)+1}\sqrt{q} \\ &\leq q \ . \end{aligned}$$

Die Gleichung (4.9) gilt für $\boldsymbol{d}$ also auch über $\mathbb{Z}$. Analog dazu sieht man, daß $\boldsymbol{d}$ auch (4.9) über $\mathbb{Z}$ erfüllt, d.h. es gilt $\boldsymbol{d} \in L^2$.

Mit Lemma 4.19 folgt nun, daß es $1 \leq j_1 < j_2 < j_3 \leq n$ gibt, so daß $d_{j_1}x_1 + d_{j_2}x_2 + d_{j_3}x_3 = 0$ keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt. Wegen $3|d_\nu|2^\ell < \mathcal{L}(n-2,\ell)2^{2\ell+3}\sqrt{q} < q$ besitzt diese Gleichung dann aber auch keine solche Lösung modulo q, d.h. $(d_{j_1}, d_{j_2}, d_{j_3}) \notin Y_3(2^\ell)$, d.h. $\boldsymbol{d}$ ist 3-fach wild.

□

Resultate. Tabelle 4.4 vergleicht Generator 3 mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden. Die angegebenen Parameter liefern eine Sicherheit von ungefähr 2^{160} gegen Attacken beschränkter Länge. Mem bezeichnet die Anzahl der zu speichernden Gruppenelemente und $Mult$ die Anzahl der Multiplikationen pro Ausgabe.

Mem	k, ℓ, A	$Mult$	Prec.-Methode	$Mult$
8	6,10,21	53	de Rooij	59
16	14,5,17	34	de Rooij	52
30	28,3,15	26	Lim-Lee	55.5
60	58,2,12	20	Lim-Lee	45.5
124	122,1,12	17	Lim-Lee	37

Tabelle 4.4: Vergleich von Generator 3 mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

4.3.3 Auf beliebig vielen Runden beruhende Sicherheit

Der Ansatz der letzten beiden Abschnitte läßt sich verallgemeinern. Durch Verwendung von m variablen Präsignaturen können wir die Sicherheit auf $m+1$ Runden beruhen lassen. Es zeigt sich allerdings, daß der Generator bei $m > 2$ und einer angestrebten Sicherheit von 2^{160} weniger effizent ist als Generator 2.

Generator 3

1. EINGABE $\boldsymbol{r} \in_R \mathbb{Z}_q^k$, $\boldsymbol{s} \in_R \mathbb{Z}_q^m$.
   $\nu := 1$.
2. Wähle ein zufälliges $\boldsymbol{u}_\nu \in \mathcal{C}$ und setze $r' := \sum_{i=1}^k u_{i,\nu} r_i \bmod q$.
3. $s_1 := 2^{\ell'} s_1$, $s_2 := 2^{\ell'}(s_2 + s_1)$, $\ldots$, $s_m := 2^{\ell'}(s_m + s_{m-1})$.
4. AUSGABE $r_\nu^* := r' + s_m$ (verwende $(r_\nu^*, g^{r_\nu^*})$ für die nächste Signatur)
   $\nu := \nu + 1$,
   GO TO 2 für die nächste Runde.

Der Generator benötigt $A + m(\ell' + 1) + \ell - 2$ Multiplikationen pro Ausgabe. Für $m = 1$ und $\ell' = \ell$ erhalten wir Generator 1 und für $m = 2$ und $\ell' = \ell + 1$ Generator 2.

Der folgende Satz zeigt, daß bei geigneter Wahl von ℓ' die Sicherheit von Generator 3 gegen Attacken beschränkter Länge auf $m+1$ Runden beruht. Im Fall $m = 2$ benötigen wir die Voraussetzung $\ell' \geq \ell + 2$, d.h. das Resultat ist schwächer als Satz 4.12.

Satz 4.17 *Sei $2^{\ell'}/m - 1 \geq 2^{\ell}$ und $\mathcal{L}(n-m,\ell)2^{n\ell'+1}\binom{n+m-2}{m-1} < q^{1/m}$. Dann besitzt jede einfache Attacke $\boldsymbol{c} \in \mathbb{Z}^{n+1}$ höchstens Erfolgswahrscheinlichkeit $\binom{k\ell}{A}^{-(m+1)}$.*

Der Beweis ist weitgehend analog zum Beweis von Satz 4.12. Wir skizzieren deshalb nur die wesentlichen Züge.

Zunächst verallgemeinern wir die Definitionen des letzen Abschnittes. Eine einfache Attacke $\boldsymbol{c}$ nennen wir t-fach zahm, falls für alle $1 \leq i_1 < \cdots < i_t \leq n$ die Gleichung $c_{i_1}x_1 + \cdots + c_{i_t}x_t = 0 \bmod q$ eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^t$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt. Anderenfalls nennen wir die Attacke t-fach wild.

Analog zu Satz 4.13 zeigt zuerst die obere Schranke für die Erfolgswahrscheinlichkeit von m-fach wilden Attacke.

Satz 4.18 *Jede m-fach wilde Attacke hat höchstens Erfolgswahrscheinlichkeit $\binom{k\ell}{A}^{-(m+1)}$.*

Auf der anderen Seite überzeugt man sich durch vollständige Induktion davon, daß

$$s_j^{[\nu]} = \sum_{i=0}^{j-1} \binom{\nu+j-2}{j-1} 2^{(\nu+j-1)\ell'} s_{i+1}^{[0]}$$

für $j = 1, \ldots, m$ gilt. Daraus folgt, daß jede einfache Attacke $\boldsymbol{c}$ mit Erfolgswahrscheinlichkeit größer 0 die Gleichungen

$$\sum_{\nu=1}^{n} c_\nu \binom{\nu+j-2}{j-1} 2^{(\nu-1)\ell'} = 0 \bmod q \tag{4.15}$$

für $j = 1, \ldots, m$ erfüllen muß. Das folgende Lemma zeigt – analog zu Lemma 4.14 –, daß es für jede Attacke, welche (4.15) für $j = 1, \ldots, t$ über $\mathbb{Z}$ erfüllt, Indices $1 \leq i_1 < \cdots < i_{t+1} \leq n$, gibt, so daß $c_{i_1}x_1 + \cdots + c_{i_{t+1}}x_{t+1} = 0$ keine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^t$ mit $\|\boldsymbol{x}\|_\infty < 2^\ell$ besitzt.

Lemma 4.19 *Sei $2^{\ell'}/m - 1 \geq 2^\ell$ und $\boldsymbol{d}$ eine einfache Attacke, welche (4.15) für $j = 1, \ldots, m$ über $\mathbb{Z}$ erfüllt. Dann gibt es $1 \leq a_0 < \cdots < a_m \leq n$ mit $|d_{a_i}| > 2^\ell |d_{a_{i+1}}|$ für $0 \leq i \leq m-1$.*

Schließlich folgert man daraus, daß jede 3-fach zahme Attacke beschränkter Länge Erfolgswahrscheinlichkeit 0 hat.

Satz 4.20 *Sei $\mathcal{L}(n-m,\ell)2^{n\ell'+1}\binom{n+m-2}{m-1} < q^{1/m}$. Dann besitzt jede m-fach zahme Attacke $\boldsymbol{c} \in \mathbb{Z}^{n+1}$ Erfolgwahrscheinlichkeit 0.*

Damit ist dann Satz 4.17 bewiesen.

Für $m > 2$ und bei einer Sicherheit von 2^{160} ist Generator 3 weniger effizient als Generator 2. Wir geben deshalb hier keine Beispiele an.

Der Beweis von Lemma 4.19 läßt sich als einziger Teil im Beweis von Satz 4.17 nicht als einfache Verallgemeinerung aus dem entsprechenden Resultat des vorangegangenen Abschnittes ableiten. Wir geben ihn deshalb explizit an:

Beweis von Lemma 4.19 Für $\boldsymbol{d} \in \mathbb{Z}^n$ und $i \in [1,n]$ sei χ_i das größte $j < i$ mit $\operatorname{sgn}(d_j) \neq \operatorname{sgn}(d_i)$ und $M_{\boldsymbol{d}}(i)$ das $j > \chi_i$, für das $|d_j|$ maximal ist. Falls $|d_j|$ für mehrere $j \in (\chi_i, n]$ maximal ist, sei $M_{\boldsymbol{d}}(i)$ das kleinste von diesen. Ferner sei $\operatorname{sgn}(\boldsymbol{d})$ das Vorzeichen des führenden Koeffizienten von $\boldsymbol{d}$. Die Behauptung folgt nun aus Aussage 3 des folgenden Lemmas.

Lemma 4.21 *Sei $1 \leq t \leq m$ und $\boldsymbol{d}$ eine einfache Attacke, welche (4.15) für $j = 1, \ldots, t$ über $\mathbb{Z}$ erfüllt. Dann gibt es $1 \leq a_0 < \cdots < a_t \leq n$, so daß für alle $i = 0, \ldots, t$ gilt*

1. $\operatorname{sgn}(d_{a_i}) = (-1)^{t-i} \operatorname{sgn}(\boldsymbol{d})$.

2. $a_i = M_{\boldsymbol{d}}(a_i)$.

Für $t \geq 1$ und $i = 0, \ldots, t-1$ gilt weiterhin

3. $|d_{a_i}| > \left(\frac{2^{\ell'}}{t} - 1\right) |d_j|$ *für alle $j > a_i$ mit* $\operatorname{sgn}(d_j) = -\operatorname{sgn}(d_{a_i})$.

Beweis. Sei $L^0 := \mathbb{Z}^n$ und für $1 \leq t \leq m$ sei L^t das Gitter der Vektoren in $\mathbb{Z}^n$, welche (4.15) für $i = 1, \ldots, t$ über $\mathbb{Z}$ erfüllen. Man zeigt durch vollständige Induktion, daß für $t = 1, \ldots, m$ die Vektoren $\boldsymbol{b}_1^t, \ldots, \boldsymbol{b}_{n-t}^t$ eine Basis von L^t bilden, wobei $\boldsymbol{b}_i^0$ der i-te Einheitsvektor und $\boldsymbol{b}_i^j := \boldsymbol{b}_{i+1}^{j-1} - 2^{\ell'}\boldsymbol{b}_i^{j-1}$ für $j = 1, \ldots, m$ und $i = 1, \ldots, n-j$ ist. Es folgt daraus, daß es für jedes $\boldsymbol{d} \in L^t$ ein $\boldsymbol{c} \in L^{t-1}$ gibt, so daß $c_n = 0$ und $d_\nu = c_{\nu-1} - 2^{\ell+1}c_\nu$ mit $c_0 := 0$ für $\nu = 1, \ldots, n$ gilt.

Wir beweisen nun die Behauptung durch vollständige Induktion. Wir können dabei ohne Einschränkung $d_n \neq 0$ annehmen.

Im Fall $t = 0$ erfüllt $a_0 := M_{\boldsymbol{d}}(n)$ die Behauptungen. Sei nun $t \geq 1$ und die Behauptung gelte für $t-1$. Sei $\boldsymbol{d} \in L^t$. Dann gilt es ein $\boldsymbol{c} \in L^{t-1}$ mit $c_n = 0$ und

$$d_j = c_{j-1} - 2^{\ell'} c_j \tag{4.16}$$

für $j = 1, \ldots, n-1$. Nach Induktionsvoraussetzung gibt es dann $1 \leq a_0 < \cdots < a_{t-1} \leq n-1$, die 1. – 3. erfüllen. Wir setzen $b_i := M_{\boldsymbol{d}}(a_i)$ für $i = 0, \ldots, t-1$ und $b_t := M_{\boldsymbol{d}}(n)$. Wegen $a_i < a_{i+1}$ und der Monotonie von $M_{\boldsymbol{d}}$ gilt zunächst $b_i \leq b_{i+1}$ für $i = 0, \ldots, t-1$. Wir zeigen, daß die b_i die Bedingungen 1. – 3. für den Vektor $\boldsymbol{d}$ erfüllen. Mit der Eigenschaft 1. folgt dann $b_i \neq b_{i+1}$.

Wegen der Induktionsvoraussetzung $a_i = M_{\boldsymbol{c}}(a_i)$ gilt

$$|c_x| \leq |c_{a_i}| \quad \text{für alle } x \geq a_i \ . \tag{4.17}$$

Auf der anderen Seite folgt aus $a_i = M_{\boldsymbol{c}}(a_i)$ entweder $|c_{a_i}| > |c_{a_i-1}|$ oder $\operatorname{sgn}(c_{a_i-1}) = -\operatorname{sgn}(c_{a_i})$. Mit (4.16) ergibt sich

$$|d_{a_i}| > \left(2^{\ell'} - 1\right) |c_{a_i}| \tag{4.18}$$

und

$$\operatorname{sgn}(d_{a_i}) = -\operatorname{sgn}(c_{a_i}) \tag{4.19}$$

und insbesondere auch $d_{a_i} \neq 0$. Wir beweisen nun nacheinander die drei Aussagen.

1. Wegen $b_t := M_{\boldsymbol{d}}(n)$ ist $\operatorname{sgn}(d_{b_t}) = \operatorname{sgn}(\boldsymbol{d})$ unmittelbar klar.

Sei nun $i \leq t-1$. Wir können o.B.d.A. $c_{a_i} > 0$ annehmen. Wegen $\operatorname{sgn}(\boldsymbol{d}) = \operatorname{sgn}(\boldsymbol{c})$ genügt es nun, $d_{b_i} < 0$ zu zeigen. Wegen $b_i = M_{\boldsymbol{d}}(a_i)$ und (4.18) gilt $|d_{b_i}| \geq |d_{a_i}| \geq \left(2^{\ell'} - 1\right) |c_{a_i}| > 0$, d.h. $d_{b_i} \neq 0$.

Wir nehmen nun an, $d_{b_i} = c_{b_i-1} - 2^{\ell'} c_{b_i}$ wäre positiv. Mit (4.19) folgt dann $\operatorname{sgn}(d_{b_i}) \neq \operatorname{sgn}(d_{a_i})$. Wegen $b_i = M_{\boldsymbol{d}}(a_i)$ impliziert das wiederum $b_i > a_i$. Mit (4.17) folgt somit $|c_{b_i-1}| \leq c_{a_i}$. Wir zeigen nun, daß $d_{b_i} < |d_{a_i}|$ gilt und erhalten damit einen Widerspruch zur Definition von b_i. Dafür unterscheiden wir zwei Fälle:

Falls $c_{b_i} \geq 0$ gilt, folgt $d_{b_i} < |c_{b_i-1}| \leq c_{a_i}$. Nach (4.18) ist das kleiner als $|d_{a_i}|$.

Falls c_{b_i} negativ ist, gilt nach der Induktionsvoraussetzung $c_{a_i} > \left(2^{\ell'}/(t-1) - 1\right) |c_{b_i}|$. (Im Fall $i = t-1$ gilt nach Konstruktion $a_i = M_{\boldsymbol{c}}(n-1)$ und es folgt $c_{b_i} \geq 0$ aus $b_i \geq n-1$.) Wegen $|c_{b_i-1}| < c_{a_i}$ folgt mit (4.16)

$$d_{b_i} \ \leq \ 2^{\ell'} |c_{b_i}| + |c_{b_i-1}| \ < \ \frac{t 2^{\ell'} - t + 1}{2^{\ell'} - t + 1} c_{a_i} \ < \ \frac{t\left(2^{\ell'} - 1\right)}{2^{\ell'} - t} c_{a_i} \ .$$

Mit (4.18) folgt nun $d_{b_i} < t/(2^{\ell'} - t)|d_{a_i}|$, was wegen $2t \leq 2^{\ell} t \leq 2^{\ell'}$ höchstens $|d_{a_i}|$ ist.

2. Da $M_{\boldsymbol{d}}$ idempotent ist, d.h. $M_{\boldsymbol{d}}(M_{\boldsymbol{d}}(i)) = i$ für alle i gilt, folgt die Aussage sofort.

3. Sei $i \le t-1$ und $j > b_i$ mit $\operatorname{sgn}(d_j) = -\operatorname{sgn}(d_{b_i})$. O.B.d.A. sei $d_j > 0$ und damit nach (4.19) $c_{a_i} > 0$. Da nach der Definition der b_i nicht $a_i \ge j > b_i$ gelten kann, folgt $j > a_i$. Mit (4.17) ergibt sich somit $|c_{j-1}| \le c_{a_i}$. Wir unterscheiden nun zwei Fälle.

Im Fall $i = t-1$ gilt nach Konstruktion $a_i = M_{\boldsymbol{c}}(n-1)$. Aus $j > a_i$ folgt $c_j > 0$ und damit $d_j = c_{j-1} - 2^{\ell'} c_j < c_{j-1}$. Nach (4.17) ist das höchstens c_{a_i}, was wiederum nach (4.18) kleiner als $\left(2^{\ell'} - 1\right)^{-1} |d_{a_i}|$ ist. Wegen $b_i = M_{\boldsymbol{d}}(a_i)$ ergibt sich also $\left(2^{\ell'} - 1\right) d_j \le |d_{b_i}|$.

Sei nun $i \le t-2$. Falls c_j positiv ist, folgt $|d_{b_i}| > \left(2^{\ell'} - 1\right) d_j$ analog zum Fall $i = t-1$. Falls c_j negativ ist, gilt nach der Induktionsvoraussetzung $|c_j| \left(2^{\ell'} - t + 1\right) / (t-1) < c_{a_i}$. Es folgt mit (4.16)

$$d_j \;\le\; 2^{\ell'} |c_j| + |c_{j-1}| \;<\; \frac{t2^{\ell'} - t + 1}{2^{\ell'} - t + 1} c_{a_i} \;<\; \frac{t\left(2^{\ell'} - 1\right)}{2^{\ell'} - t} c_{a_i}$$

und mit (4.18) erhalten wir $|d_{a_i}| > \left(2^{\ell'}/t - 1\right) d_j$.

□

4.3.4 Probabilistische Transformationen

Für die Generatoren 1, 2 und 3 waren wir leider nicht in der Lage, eine obere Schranke für die Erfolgswahrscheinlichkeit einfacher Attacken großer Länge zu beweisen. Wir erhalten daher für diese Generatoren keine stärkere untere Schranke für die Güte als die aus Satz 4.2. Empirische Untersuchungen zeigen zwar, daß sich die Erfolgswahrscheinlichkeit von einfachen Attacken bei zunehmender Länge stetig verringert, doch diese Ergebnisse sind aus theoretischer Sicht noch nicht zufriedenstellend.

Wir stellen nun eine Variante von Generator 1 vor, bei der die Sicherheit auf den Zufallsbits von zwei Runden beruht und für die wir eine untere Schranke für die Güte beweisen können. Möglich wird das durch die Verwendung probabilistischer Transformationen.

Generator 4

1. EINGABE $\boldsymbol{r} \in_R \mathbb{Z}_q^k$, $\boldsymbol{s} \in_R \mathbb{Z}_q^m$.
   $\nu := 1$.

2. Wähle ein zufälliges $\boldsymbol{u}_\nu \in \mathcal{C}$ und setze $r' := \sum_{i=1}^k u_{i,\nu} r_i \bmod q$.

3. Wähle ein zufälliges $\boldsymbol{T}_\nu \in [0, 2^{\ell+1})^{m-1}$ mit Hamminggewicht h und setze $s_m := 2^{\ell+1} s_m + \sum_{j=1}^{m-1} T_{j,\nu} s_j$.

4. AUSGABE $r^*_\nu := r' + s_m$ (verwende $(r^*_\nu, g^{r^*_\nu})$ für die nächste Signatur)
$\nu := \nu + 1$,
GO TO 2 für die nächste Runde.

Hierbei ist $h \leq \ell$ ein zusätzlicher Parameter. Der multiplikative Teil von Schritt 3 wird mit der „Square and multiply"-Methode durchgeführt. Damit benötigt das Verfahren $A + 2\ell + h$ Multiplikationen pro Ausgabe. Es gilt

$$r^*_\nu = r'_\nu + 2^{\nu(\ell+1)} s^{[0]}_m + \sum_{j=1}^{m-1} s^{[0]}_j \sum_{i=1}^{\nu} T_{j,i}\, 2^{(\nu-i)(\ell+1)} \bmod q \ . \tag{4.20}$$

Ein einfache Attacke $\boldsymbol{c}$, für die $\sum c_\nu r^*_\nu = 0 \bmod q$ unabhängig von $\boldsymbol{r}$ und $\boldsymbol{s}$ gilt, erfüllt also

$$\sum_{\nu=1}^{n} c_\nu 2^{-\nu(\ell+1)} = 0 \bmod q \tag{4.21}$$

wegen der Unabhängigkeit von s_m,

$$\sum_{\nu=1}^{n} c_\nu \sum_{i=1}^{\nu} T_{j,i}\, 2^{(i-\nu-1)(\ell+1)} = 0 \bmod q \quad \text{für } j = 1, \ldots, m \tag{4.22}$$

wegen der Unabhängigkeit von $s_1, \ldots, s_{m-1}$ und

$$\sum_{\nu} c_\nu \boldsymbol{u}^{[\nu]} = \boldsymbol{0} \bmod q \tag{4.23}$$

wegen der Unabhängigkeit von $\boldsymbol{r}$.

Wir zeigen nun, daß sich durch die zufällige Transformation die Erfolgswahrscheinlichkeit zahmer Attacken wesentlich verkleinert.

Lemma 4.22 *Sei $\lambda \leq \ell+1$ und $\boldsymbol{c}$ eine einfache Attacke, welche (4.21) und $(c_\nu, c_\mu) \in Y(2^\lambda)$ für alle μ, ν erfüllt. Dann gilt (4.22) höchstens mit Wahrscheinlichkeit*

$$\binom{(m-1)(\ell+1)}{h}^{-\theta_{\lambda,\ell}} ,$$

wobei $\theta_{\lambda,\ell}$ das maximale $\alpha \in \mathbb{Z}$ mit $(\alpha+1)(\ell+1) + 3 + \log(\mathcal{L}(\alpha, \lambda)) < \log q$ ist.

Tabelle 4.5 gibt für $q \approx 2^{160}$ und alle $2 \leq \ell \leq 12$ und $1 \leq \lambda \leq 5$ die Werte $\theta_{\lambda,\ell}$ an. Für $\lambda \geq 6$ gilt $\theta_{\lambda,\ell} = \lfloor (156 - \ell)/(\lambda + \ell + 1) \rfloor$.
Beweis. Analog zu Lemma 4.10 folgt aus Gleichung (4.21) $2^{n(\ell+1)}\mathcal{L}(n-1, \lambda) \geq q/2$. und damit $n > \theta_{\lambda,\ell}$. Der besseren Lesbarkeit halber schreiben wir im folgenden $\theta = \theta_{\lambda,\ell}$.

	ℓ	2	3	4	5	6	7	8	9	10	11	12
λ												
1		51	38	30	25	21	18	16	14	13	12	11
2		50	37	29	24	21	18	16	14	13	11	10
3		-	36	28	23	20	17	15	13	12	11	10
4		-	-	27	22	19	16	14	13	11	10	9
5		-	-	-	18	16	13	12	10	9	8	8

Tabelle 4.5: Numerisch ermittelte Werte $\theta_{\lambda,\ell}$.

Es reicht nun aus, zu zeigen, daß jede der Gleichungen (4.22) höchstens für ein Tupel $(T_{j,n-\theta},\ldots,T_{j,n-1}) \in [0,2^{\ell+1})^\theta$ erfüllt ist. Im folgenden sei $j \in [0,m-1]$ fest gewählt.

Da $(c_\nu, c_\mu) \in Y(2^\lambda)$ für alle μ,ν gilt, können wir $c_n = 1$ und $c_\nu = a_\nu/b_\nu \bmod q$ mit $a_\nu, b_\nu \in (-2^\lambda, 2^\lambda)$ für $1 \le \nu \le n$ annehmen. Wir fixieren in (4.22) die Werte $T_{j,1},\ldots,T_{j,n-\theta-1}$ und bringen alle konstanten Terme auf die rechte Seite. Nach Multiplikation mit $d := \mathrm{kgV}(b_{n-\theta},\ldots,b_{n-1})$ erhalten wir

$$\sum_{\nu=n-\theta}^{n} da_\nu/b_\nu \sum_{i=n-\theta}^{\nu} T_{j,i}\, 2^{(\nu-i)(\ell+1)} = C \bmod q. \tag{4.24}$$

mit einer Konstanten $C \in (-q/2, q/2)$.

Zunächst beachte man, daß wegen $T_{j,i} \le 2^{\ell+1}$ und $|da_\nu/b_\nu| < d2^\ell$ alle Summanden in (4.24) mit $\nu \le n-1$ durch $d2^{(\nu+2-n+\theta)(\ell+1)}$ beschränkt sind. Wegen $da_n/b_n = d$ ist der Summand mit $\nu = n$ sogar durch $d2^{(\theta+1)(\ell+1)+1}$ beschränkt. Insgesamt ist die linke Seite von (4.24) durch

$$d\,2^{(\theta+1)(\ell+1)+2} \;\le\; 2^{\mathcal{L}(\theta,\lambda)+(\theta+1)(\ell+1)+2} \;<\; q/2$$

beschränkt. Damit gilt (4.24) auch über $\mathbb{Z}$.

Über $\mathbb{Z}$ schreiben wir (4.24) einfacher als $\sum_{i=n-\theta}^{n} T_{j,i}B_i = C$ mit $B_i := d\sum_{\nu=i}^{n} a_\nu/b_\nu 2^{(\nu-i)(\ell+1)}$ schreiben. Die folgende Überlegung zeigt, daß $\sum_{i=n-\theta}^{n-1} T_{j,i}B_i$ für verschiedene Tupel $(T_{j,n-\theta},\ldots,T_{j,n-1})$ paarweise verschiedene Werte annimmt. Damit kann Gleichung (4.24) über $\mathbb{Z}$ auch nur für ein solches Tupel erfüllt sein.

Für alle $i \le n-1$ gilt

$$B_i = 2^{\ell+1}B_{i+1} + \delta_i \tag{4.25}$$

mit $|\delta_i| < d2^\ell$. Wegen $B_n = d$ folgt daraus zunächst $B_i > 2^\ell B_{i+1}$ und damit insbesondere $B_i > d2^\ell$ für alle $i \le n-1$. Für $i \le n-2$ ergibt sich daraus mit (4.25) $B_i > (2^{\ell+1}-1)B_{i+1}$.

Wir nehmen nun an, es gilt $\sum_{i=n-\theta}^{n-1}(T_{j,i} - T'_{j,i})B_i = 0$ für $(T_{j,n-\theta},\ldots,T_{j,n-1}) \neq (T'_{j,n-\theta},\ldots,T'_{j,n-1})$. Sei i_0 das minimale i mit $T_{j,i} \neq T'_{j,i}$. Dann gilt $|T_{j,i_0} - T'_{j,i_0}|B_{i_0} \ge B_{i_0}$.

Andererseits sind wegen $h \le \ell$ die $T_{j,i}$ durch $2^{\ell+1}-2$ beschränkt und es gilt

$$\begin{aligned} |T_{j,i_0} - T'_{j,i_0}|B_{i_0} &= \left| \sum_{i=i_0+1}^{n-1} (T_{j,i} - T'_{j,i})B_i \right| \\ &\le \sum_{i=i_0+1}^{n-1} \left(2^{\ell+1}-2\right) B_i \\ &< \sum_{i=i_0+1}^{n-1} (B_{i-1} - B_i) \\ &= B_{i_0} - B_{n-1} \end{aligned}$$

Das ist ein Widerspruch.

□

Wir wären nun bereits in der Lage, eine untere Schranke für die Erfolgswahrscheinlichkeit aller Attacken zu beweisen. Für wilde Attacken haben wir dies schon in Abschnitt 4.3.1 getan, und für zahme Attacken folgt eine Schranke direkt aus dem vorangegangenen Lemma mit $\lambda = \ell$. Diese Schranke wäre jedoch sehr grob. Falls z.B. eine zahme Attacke $(c_t, c_m) \notin Y(2^{\ell-1})$ für ein Paar t, m erfüllt, so erhalten wir mit Satz 4.2 und Lemma 4.22 die Schranke $\mathrm{S}(\boldsymbol{c}, \boldsymbol{r}) \le \binom{(m-1)(\ell+1)}{h}^{-\theta_{\ell-1,\ell}} \binom{k\ell}{A}^{-1}$. Das folgende Lemma zeigt jedoch, daß $\sum c_\nu \boldsymbol{u}_\nu = 0 \bmod q$ mit wesentlich geringerer Wahrscheinlichkeit als $\binom{k\ell}{A}^{-1}$ gilt.

Lemma 4.23 *Sei* $\boldsymbol{c}$ *eine einfache Attacke mit* $(c_t, c_m) \notin Y(2^\lambda)$ *für ein Paar* t, m *und festes* $1 \le \lambda \le \ell - 1$. *Dann gilt (4.23) höchstens mit Wahrscheinlichkeit* $2^{k(\ell-\lambda)}\binom{k\ell}{A}^{-2}$.

Beweis. Falls $\boldsymbol{c}$ wild ist, folgt die Behauptung aus Lemma 4.9. Sei nun $\boldsymbol{c}$ zahm und $(c_a, c_b) \notin Y(2^\lambda)$. Dann können wir ohne Einschränkung $|c_a| < |c_b| < 2^\ell$ und $|c_b| \ge 2^\lambda$ annehmen. Außerdem können wir annehmen, daß $|c_a|$ und $|c_b|$ teilerfremd sind.

Wir fixieren alle $\boldsymbol{u}_\nu \in \mathcal{C}$ für $\nu \ne a, b$ und nehmen an, daß $\sum c_\nu \boldsymbol{u}_\nu = 0 \bmod q$ für zwei Paare $(\boldsymbol{u}_a, \boldsymbol{u}_b) \ne (\boldsymbol{u}'_a, \boldsymbol{u}'_b) \in \mathcal{C}^2$ gilt. Dann folgt

$$c_a(u_{i,a} - u'_{i,a}) + c_b(u_{i,b} - u'_{i,b}) = 0 \bmod q \tag{4.26}$$

für $i = 1, \ldots, k$. Wegen $2^{2\ell} \ll q$ gelten diese Gleichungen auch über $\mathbb{Z}$. Wegen $\mathrm{ggT}(|c_a|, |c_b|) = 1$ ist somit jede Lösung von (4.26) ein ganzzahliges Vielfaches von $(c_b, -c_a)$. Andererseits gilt $u_{i,a} - 2^\ell < (u_{i,a} - u'_{i,a}) < u_{i,a}$ für jedes $\boldsymbol{u}'_a \in \mathcal{C}$. Damit ist bei festen $\boldsymbol{u}_a, \boldsymbol{u}_b$ für jedes $1 \le i \le k$ die Anzahl der Lösungen $(\boldsymbol{u}'_a, \boldsymbol{u}'_b) \in \mathcal{C}^2$ von (4.26) durch $\left(2^\ell - 1\right)/c_b + 1 \le 2^{\ell-\lambda}$ beschränkt.

□

Wir erhalten nun direkt eine untere Schranke für die Güte des Generators 4.

Satz 4.24 *Generator 4 besitzt mindestens die Güte* $\min_\lambda(f(\lambda))$, *wobei* $f(\lambda)$ *durch*

$$f(\lambda) = \begin{cases} \binom{(m-1)(\ell+1)}{h}^{\theta_{\lambda,\ell}} \max\left(\binom{k\ell}{A}, 2^{-k(\ell+1-\lambda)}\binom{k\ell}{A}^2\right) & \text{für} \quad 1 \le \lambda \le \ell \ , \\ \binom{(m-1)(\ell+1)}{h}\binom{k\ell}{A}^2 & \text{für} \quad \lambda = \ell + 1 \end{cases}$$

und $\theta_{\lambda,\ell}$ *als das maximale* $\alpha \in \mathbb{Z}$ *mit* $(\alpha+1)(\ell+1) + 3 + \log(\mathcal{L}(\alpha,\lambda)) < \log q$ *definiert ist.*

Beweis. Sei $\boldsymbol{c}$ eine einfache Attacke. Wir betrachten zunächst den Fall, daß $(c_t, c_m) \in Y(2)$ für alle t, m gilt. Dann folgt aus Lemma 4.22 und Satz 4.2 $\mathrm{S}(\boldsymbol{c}) \le f(1)$.

Sei nun $(c_t, c_m) \in Y(2^\lambda)$ für alle μ, ν aber $(c_t, c_m) \notin Y(2^{\lambda-1})$ für ein Paar t, m und für ein $2 \le \lambda \le \ell$. Dann gilt nach Lemma 4.22, Lemma 4.23 und Satz 4.2 $\mathrm{S}(\boldsymbol{c}) \le f(\lambda)$.

Es bleibt nun nur noch der Fall, daß $\boldsymbol{c}$ wild ist. Wir können wieder annehmen, daß $c_n \neq 0$ ist und $\langle \boldsymbol{c}, \boldsymbol{r}^* \rangle = 0 \bmod q$ unabhängig von $\boldsymbol{s}$ gilt. Dann erhalten wir die Gleichungen $\sum_{i=1}^n T_{j,i} B_i = 0$ für $j = 1, \ldots, m-1$ mit $B_i := \sum_{\nu=i}^n c_\nu 2^{(\nu-i)(\ell+1)}$. Wegen $B(n) = c_n \neq 0$ ist jede dieser Gleichungen bei fixierten $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_{n-1}$ höchstens für ein $\boldsymbol{T}_n$ erfüllt. Weiterhin gilt nach Satz 4.9, daß $\sum_{\nu=1}^n c_\nu \boldsymbol{u}_\nu = \boldsymbol{0} \bmod q$ höchstens mit Wahrscheinlichkeit $\binom{k\ell}{A}^{-2}$ gilt und wir erhalten insgesamt $\mathrm{S}(\boldsymbol{c}) \le f(\ell+1)$.

□

Wir erhalten eine effizientere Variante des Generators, wenn wir die $\boldsymbol{T}_\nu$ zufällig mit Hamminggewicht kleiner oder gleich h wählen. Der Ausdruck $\binom{(m-1)(\ell+1)}{h}$ in der Definition von $f(\lambda)$ ist dann durch $\sum_{i=1}^h \binom{(m-1)(\ell+1)}{i}$ zu ersetzen.

Resultate. Tabelle 4.6 vergleicht Generator 4 bei einer Güte von 2^{160} mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden. *Mem* bezeichnet dabei die Anzahl der zu speichernden Gruppenelemente und *Mult* die Anzahl der Multiplikationen pro Ausgabe.

Mem	m, k, ℓ, A, h	*Mult*	Prec.-Methode	*Mult*
8	2,6,15,25,6	61	de Rooij	59
16	2,14,8,22,2	40	de Rooij	52
30	3,27,5,20,1	31	Lim-Lee	55.5
60	6,54,4,16,1	25	Lim-Lee	45.5
124	5,119,2,15,1	20	Lim-Lee	37

Tabelle 4.6: Vergleich von Generator 4 mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

4.4 Unterschriften mit elliptischen Kurven

DL-Signaturverfahren werden meistens auf multiplikativen Gruppen von endlichen Körpern realisiert. In den letzten Jahren finden daneben zunehmend elliptische Kurven Verwendung. Wir diskutieren hier die speziellen Aspekte, welche sich für die Berechnung von Präsignaturen bei der Verwendung elliptischer Kurven ergeben. Dabei stellen wir jedoch nur die notwendigsten Fakten über elliptische Kurven und ihre Verwendung in der Kryptographie bereit. Für eine umfassendere Einführung verweisen wir auf [Kob84] und [Men93].

Definition 4.8 *Für eine Primzahl p und eine natürliche Zahl n sei $\mathbb{F} = \mathbb{F}_{p^n}$ der Körper mit p^n Elementen und $\overline{\mathbb{F}}$ der algebraische Abschluß von $\mathbb{F}$. Außerdem sei ein Polynom*

$$F(x, y) := y^2 + a_1xy + a_3y + x^3 + a_2x^2 + a_4x + a_6$$

mit $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ gegeben, so daß für alle $(x, y) \in \overline{\mathbb{F}}^2$ mit $F(x, y) = 0$ mindestens eine der partiellen Ableitungen $\frac{\partial F}{\partial x}$, $\frac{\partial F}{\partial y}$ nicht verschwindet. Dann heißt die Menge E der Punkte $(x, y) \in \overline{\mathbb{F}}^2$ mit $F(x, y) = 0$ vereinigt mit einem symbolischen Punkt $\mathcal{O}$ eine elliptische Kurve über $\mathbb{F}$. *$\mathcal{O}$ heißt der* Unendlichkeitspunkt.

Auf einer elliptischen Kurve E über $\mathbb{F}$ läßt sich eine Verknüpfung definieren, so daß E eine kommutative Gruppe mit dem neutralen Element $\mathcal{O}$ bildet. Für die Definition der Gruppenverknüpfung verweisen wir auf [Men93]. Die Untermenge $E(\mathbb{F}) := E \cap \mathbb{F}^2$ ist eine Untergruppe von E. Außerdem gilt $|E(\mathbb{F})| = p^n + 1 - t$ mit $|t| \leq 2\sqrt{p^n}$ und mit $\mathrm{ggT}(t, p) = 1$ oder $t \in \{0, \pm\sqrt{p^n}, \pm 2\sqrt{p^n}\}$.

Nach heutigem Wissensstand ist das Problem des diskreten Logarithmus auf elliptischen Kurven wesentlich schwieriger als in endlichen Körpern – für die meisten Klassen von elliptischen Kurven sind keine subexponentiellen Algorithmen bekannt. (Für detaillierte Kriterien an kryptographisch verwendbare Kurven siehe z.B. [MP97].) Bei Verwendung solcher Kurven kann man daher die Parameter eines Kryptoverfahrens wesentlich kleiner dimensionieren als z.B. bei Verwendung der multiplikativen Gruppe von endlichen Körpern und erhält somit eine entsprechend bessere Performance. Dies erklärt die zunehmende Verbreitung von Signaturverfahren auf Basis elliptischer Kurven.

Für ein solches Verfahren wählt man sich eine Kurve E, so daß $|E(\mathbb{F})|$ mindestens einen großen Primfaktor q besitzt und ein Element $g \in E(\mathbb{F})$ der Ordnung q. (Für die verschiedenen Verfahren zum Auffinden einer solchen Kurve und des Elementes g sowie zur Bestimmung von $|E(\mathbb{F})|$ verweisen wir auf [LMMS94], [MVZ93], [BS91] und [LZ94].) Die dem Signaturverfahren zugrunde liegende Gruppe ist dann die von g erzeugte zyklische Gruppe G. Da in einer solchen Gruppe – unabhängig von p und n – die besten bekannten Verfahren zur Berechnung des diskreten Logarithmus (Pollards Rho–Methode, siehe [MvOV97, 106ff.], oder Shanks „Baby Step, Giant Step"–Algorithmus, [Sti95, 165f.]) Laufzeiten von $O(\sqrt{q})$ besitzen, reicht es aus $q \approx 2^{160}$ und $2^{160} < p^n < 2^{168}$ zu wählen.

Im Unterschied zur multiplikativen Gruppe $\mathbb{Z}_p^*$ benötigt eine Invertierung in einer elliptischen Kurve so gut wie keinen Rechenaufwand – das Inverse von (x, y) ist $(x, -y)$ falls $p > 2$ und $(x, x + y)$ falls $p = 2$. Eine Exponentiation kann daher besonders effizient mit Hilfe des BGMW-2 Algorithmus berechnet werden (siehe Abschnitt 4.1). Tabelle 4.7 vergleicht die in Abschnitt 4.1 beschriebenen Verfahren für $q := \mathrm{ord}(g) \approx 2^{160}$. In der Tabelle bezeichnet Mem die Anzahl der zu speichernden Punkte und Op die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve.

Verfahren	Parameter	Mem	Op
BGMW-1	$b = 12$	45	45
	$b = 32$	32	60
BGMW-2	$b = 19$	38	43[5]
	$b = 33$	32	45
de Rooij	$b = 40$	4[6]	80[7]
	$b = 20$	8[6]	59[7]
	$b = 10$	16[6]	52[7]
Lim-Lee	$h = 2, v = 2$	9	85.8
	$h = 2, v = 5$	15	74
	$h = 4, v = 2$	30	55.5
	$h = 4, v = 4$	60	45.5
	$h = 6, v = 2$	124	37

Tabelle 4.7: Vergleich der Verfahren zur Exponentiation mit Precomputation für eine elliptische Kurve.

4.4.1 Generierung pseudozufälliger Präsignaturen

Auch für die Generierung pseudozufälliger Präsignaturen können wir die effiziente Invertierbarkeit in der Gruppe ausnutzen und erhalten so eine effizientere Variante des allgemeinen Generators. Dafür ist es zunächst notwendig, die Münzwurfmenge neu zu definieren.

[5]Selbst bei Speicherung von 128 Punkten benötigt das BGMW-2 Verfahren nicht weniger Additionen
[6]Zusätzlich muß die gleiche Anzahl an variablen Punkten temporär gespeichert werden
[7]Die Werte sind aus den empirischen Abschätzungen in [dR94] abgeleitet

Definition 4.9 *Die Menge* $\mathcal{C}_{ec} \subset \{-1,0,1\}^{k\times\ell}$ *sei definiert durch*

$$\mathbf{U} \in \mathcal{C}_{ec} \quad :\Leftrightarrow \quad \mathrm{wt}(\mathbf{U}) = A \;\; \textit{und} \;\; (U_{i,j} \neq 0 \Rightarrow U_{i,j-1} = U_{i,j+1} = 0) \;\; \textit{für alle} \;\; i,j \;\; ,$$

wobei $\mathrm{wt}(\mathbf{U})$ *die Anzahl der von Null verschiedenen Einträge von* $\mathbf{U}$ *ist.*

Die Münzwurfmenge $\mathcal{C}_{ec}$ ist so definiert, daß die Zeilen jedes $\mathbf{U} \in \mathcal{C}_{ec}$ eine nicht-adjazente Form besitzen, d.h. daß keine 2 in einer Zeile aufeinanderfolgenden Einträge ungleich 0 sind. Es ist bekannt, daß die Darstellung von natürlichen Zahlen in nicht-adjazenter Form eindeutig ist [MO90b]. Daraus folgt, daß je zwei Elemente $\mathbf{U} \neq \mathbf{U}' \in \mathcal{C}_{ec}$ in mindestens einer Zeile verschiedene Zahlen darstellen, d.h. daß jedes $\mathbf{U} \in \mathcal{C}_{ec}$ mittels $u_i := \sum_{j=1}^{\ell} U_{i,j}2^{j-1}$ für $i = 1,\ldots,k$ ein anderes $\boldsymbol{u} \in (-2^\ell, 2^\ell)^k$ definiert. Wir erhalten damit das folgende Lemma.

Lemma 4.25 *Für alle* $\mathbf{U}, \widetilde{\mathbf{U}} \in \mathcal{C}_{ec}$ *mit* $\mathbf{U} \neq \widetilde{\mathbf{U}}$ *gibt es ein* $1 \leq i \leq k$ *mit* $\sum_{j=1}^{\ell} U_{i,j}2^{j-1} \neq \sum_{j=1}^{\ell} \widetilde{U}_{i,j}2^{j-1}$.

Wir können also jedes $\mathbf{U} \in \mathcal{C}_{ec}$ mit dem korrespondierenden $\boldsymbol{u} \in (-2^\ell, 2^\ell)^k$ identifizieren. Im folgenden schreiben wir daher $\boldsymbol{u} \in \mathcal{C}_{ec}$ falls $u_i = \sum_{j=1}^{\ell} U_{i,j}2^{j-1}$ für $1 \leq i \leq k$ gilt.

Generator EC

1. EINGABE $\quad \boldsymbol{r} \in_R \mathbb{Z}_q^k$
   $\nu := 1$ (ν ist die Nummer der Runde)
2. Wähle ein zufälliges $\boldsymbol{u}^\nu \in \mathcal{C}_{ec}$ und setze $r' := \sum_{i=1}^{k} u_i^\nu r_i \bmod q$.
3. AUSGABE $\quad r_\nu^* := r'$
   $\nu := \nu + 1$, GO TO 2 für die nächste Runde.

Es gilt $r_\nu^* = \sum_{i=1}^{k} u_i^\nu r_i \bmod q$. Der Generator benötigt $A-1$ Verknüpfungen und $\ell - 1$ Verdoppelungen pro Ausgabe.

Mit Lemma 4.25 erhalten wir analog zu Satz 4.2 eine untere Schranke für die Güte von Generator EC.

Satz 4.26 *Es gilt* $\mathrm{SEC} \geq |\mathcal{C}_{ec}|$.

Konkrete Ergebnisse erhalten wir nun aus dem folgenden Fakt.

Fakt 4.27 *Sei* $k\ell \geq 2A-1$. *Dann gilt für alle* ℓ

$$|\mathcal{C}_{ec}| \geq 2^A \binom{k\ell - A + 1}{A} \;\; .$$

Im Fall $\ell \leq 3$ können wir $|\mathcal{C}_{ec}|$ sogar exakt bestimmen:

$$|\mathcal{C}_{ec}| = 2^A \sum_{i=0}^{\lfloor A/2 \rfloor} \binom{k}{i} 3^i \binom{k-i}{A-2i} 4^{A-2i} \quad \textit{falls } \ell = 3,$$
$$|\mathcal{C}_{ec}| = 2^A \sum_{i=0}^{\lfloor A/2 \rfloor} \binom{k}{i} \binom{k-i}{A-2i} 3^{A-2i} \quad \textit{falls } \ell = 3 \textit{ und}$$
$$|\mathcal{C}_{ec}| = \binom{k}{A} (2\ell)^A \quad \textit{falls } \ell = 1, 2 \ .$$

Beweis. Wir zeigen zunächst die allgemeine Formel. Die Menge $|\mathcal{C}_{ec}|$ ist mindestens so groß, wie die Menge der nicht-adjazenten (d.h. keine zwei nichttrivialen Einträge folgen aufeinander) Tupel $(x_1, \ldots, x_{k\ell}) \in \{-1, 0, 1\}^{k\ell}$. Nun gibt es allgemein $\binom{N-K+1}{K}$ viele Möglichkeiten, K Kugeln so auf N Kisten zu verteilen, daß keine zwei Kugeln in benachbarten Kisten liegen. Auf unsere Situation bezogen, ergeben sich $\binom{k\ell-A+1}{A}$ viele Möglichkeiten, A nichttriviale Einträge nicht-adjazent auf $k\ell$ Komponenten zu verteilen. Auf der anderen Seite gibt es für jeden nichttrivialen Eintrag zwei mögliche Werte (-1 oder 1). Damit folgt die Abschätzung für den allgemeinen Fall.

Wir betrachten und den Fall $\ell = 4$. Für jedes $i \in [0, \lfloor A/2 \rfloor]$ gibt es $\binom{k}{i}$ viele Möglichkeiten für die Auswahl der Zeilen eines $\mathbf{U} \in \mathcal{C}_{ec}$, in denen jeweils zwei nichttriviale Einträge stehen und dann noch $\binom{k-i}{A-2i}$ viele Möglichkeiten für die Auswahl der Zeilen, in denen jeweils ein nichttrivialer Eintrag steht. Für jede Zeile mit zwei nichttrivialen Einträgen gibt es 3 Möglichkeiten für eine nicht-adjazente Anordnung dieser Einträge innerhalb der Zeile. Schließlich gibt es für jeden der A nichttrivialen Einträge 2 mögliche Werte. Damit ergibt sich die Abschätzung.

Der Beweis der Abschätzung für $\ell = 3$ ist analog dazu.

Im Fall $\ell = 2$ oder $\ell = 1$ gibt es $\binom{k}{A}$ viele Möglichkeiten für die Auswahl der Zeilen, in denen jeweils ein nichttrivialer Eintrag steht. (Im Fall $\ell = 2$ sind 2 nichttriviale Einträge in einer Zeile unzulässig, da sie nebeneinander stehen würden.) Schließlich gibt es für jeden der A nichttrivialen Einträge 2 mögliche Werte und ℓ Möglichkeiten für die seine Position innerhalb der Zeile. Damit ergibt sich auch die letzte Abschätzung.

□

Resultate. Tabelle 4.8 vergleicht Generator EC bei einer Güte von 2^{160} mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden für $q \approx 2^{160}$. Dabei bezeichnet Mem die Anzahl der gespeicherten Punkte und Op die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve.

Auf den Zufallsbits mehrerer Runden beruhende Sicherheit

Analog zu den Generatoren 2-4 können wir auch bei Generator EC die Sicherheit auf den Zufallsbits von mehreren Runden beruhen lassen.

Mem	k,ℓ,A	Op	Prec.-Methode	Op
8	8,27,35	60	de Rooij	59
16	16,16,32	46	de Rooij	52
32	32,9,30	37	BGMW-2	45
60	60,8,25	31	Lim-Lee	45.5
93	93,7,23	28	Lim-Lee	41
124	124,6,22	26	Lim-Lee	37

Tabelle 4.8: Vergleich von Generator EC mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

Generator EC2

1. EINGABE $\boldsymbol{r} \in_R \mathbb{Z}_q^k$, $\boldsymbol{s} \in_R \mathbb{Z}_q^m$
   $\nu := 1$
2. Wähle ein zufälliges $\mathbf{U}^\nu \in \mathcal{C}_{ec}$ und setze $r' := \sum_{j=1}^{\ell} 2^{j-1} \sum_{i=1}^{k} U_{i,j}^\nu r_i \bmod q$.
3. $s_1 := 2^{\ell'} s_1, \quad s_2 := 2^{\ell'}(s_2 + s_1), \quad \ldots, \quad s_m := 2^{\ell'}(s_m + s_{m-1})$.
4. AUSGABE $r_\nu^* := r' + s_m$
   $\nu := \nu + 1$, GO TO 2 für die nächste Runde.

Der Generator benötigt $A + m - 1$ Verknüpfungen und $\ell + m\ell' - 1$ Verdoppelungen pro Ausgabe.

Analog zu den Abschnitten 4.3.1, 4.3.2 und 4.3.3 können wir eine untere Schranke für die Güte des Generators beweisen. Beim Übertragen der Beweise ist jedoch zu beachten, daß für $\boldsymbol{u}, \boldsymbol{u}' \in \mathcal{C}_{ec}$ und $1 \leq i \leq k$ die Differenz $u_i - u_i'$ im allgemeinen nur im Intervall $(-2^{\ell+1}, 2^{\ell+1})$ und nicht im Intervall $(-2^\ell, 2^\ell)$ liegt. Es ist daher notwendig, $\ell' = \ell + 1 + \lceil \log m \rceil$ für $m = 1, 2$ und $\ell' = \ell + 2 + \lceil \log m \rceil$ für $m \geq 3$ zu wählen. Analog zu den Sätzen 4.11, 4.12 und 4.17 erhalten wir dann das folgende Resultat.

Satz 4.28 *Sei* $m 2^{\mathcal{L}(n-m,\ell)+\ell+2} \binom{n+m-2}{m-1} 2^{(n+m-1)\ell'} < q^{1/m}$. *Dann gilt* $\mathrm{S}(\boldsymbol{c}) \leq |\mathcal{C}_{ec}|^{-(m+1)}$ *für jede einfache Attacke* $\boldsymbol{c} \in \mathbb{Z}^{n+1}$.

Resultate. Tabelle 4.9 vergleicht Generator EC2 mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden für $q \approx 2^{160}$. Die angegebenen Parameter liefern eine Sicherheit von ungefähr 2^{160} gegen Attacken beschränkter Länge. Mem bezeichnet die Anzahl der gespeicherten Punkte und Op die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve. Für $m \geq 3$ ist Generator EC2 weniger effizient.

Mem	$m=1$		$m=2$		Precomputation	
	k,ℓ,A	*Op*	k,ℓ,A	*Op*	Methode	*Op*
8	7,13,21	47	6,10,15	49	de Rooij	59
16	15,7,18	32	14,4,15	31	de Rooij	52
32	31,4,16	24	30,2,13	23	BGMW-2	45
60	59,3,16	20	58,1,13	20	Lim-Lee	45.5
93	92,2,14	18	91,1,11	18	Lim-Lee	41
124	123,2,13	17	122,1,9	16	Lim-Lee	37

Tabelle 4.9: Vergleich von Generator EC2 mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

Probabilistische Transformationen

Analog zu Abschnitt 4.3.4 können wir durch Verwendung probabilistischer Transformationen einen Generator erhalten, dessen Sicherheit auf den Zufallsbits von zwei Runden beruht und für dessen Güte wir eine untere Schranke beweisen können. Dafür müssen wir jedoch wiederum beachten, daß für $\boldsymbol{u}, \boldsymbol{u}' \in \mathcal{C}_{ec}$ und $1 \leq i \leq k$ die Differenz $u_i - u_i'$ im allgemeinen nur im Intervall $(-2^{\ell+1}, 2^{\ell+1})$ und nicht im Intervall $(-2^{\ell}, 2^{\ell})$ liegt. Wir müssen daher Lemma 4.22 auch für $\lambda = \ell+1$ beweisen und deshalb s_2 in Schritt 3 mit $2^{\ell+2}$ multiplizieren.

Generator EC3

1. EINGABE $\quad \boldsymbol{r} \in_R \mathbb{Z}_q^k,\ \boldsymbol{s} \in_R \mathbb{Z}_q^m$
   $\nu := 1$
2. Wähle ein zufälliges $\boldsymbol{u}^\nu \in \mathcal{C}_{ec}$ und setze $r' := \sum_{i=1}^k u_i^\nu r_i \bmod q$.
3. Wähle ein zufälliges $\boldsymbol{T}_\nu \in [0, 2^{\ell+2})^{m-1}$ mit Hamminggewicht h und setze $s_m := 2^{\ell+2} s_m + \sum_{j=1}^{m-1} T_{j,\nu} s_j \bmod q$.
4. AUSGABE $\quad r_\nu^* := r' + s_m \bmod q$
   $\nu := \nu + 1$, GO TO 2 für die nächste Runde.

Der Generator benötigt $A + h$ Additionen und $2\ell + 1$ Verdoppelungen pro Ausgabe. Analog zu Satz 4.24 können wir eine untere Schranke für die Güte von Generator EC3 beweisen. Beim Übertragen der Beweise sind die folgenden beiden Punkte zu beachten:

- Da in s_m in Schritt 3 mit $2^{\ell+2}$ multipliziert wird, ist $\theta_{\lambda,\ell}$ durch $\theta_{\lambda,\ell+1}$ zu ersetzen.

- Da $\boldsymbol{T}_\nu$ aus $[0, 2^{\ell+2})^{m-1}$ ist, muß der Term $\binom{(m-1)(\ell+1)}{h}$ durch $\binom{(m-1)(\ell+2)}{h}$ ersetzt werden.

Wir erhalten den folgenden Satz.

Satz 4.29 *Generator EC3 besitzt mindestens die Güte* $\min_\lambda(f(\lambda)^{-1})$*, wobei* $f(\lambda)$ *durch*

$$\begin{array}{ll} \binom{(m-1)(\ell+2)}{h}^{-\theta_{\lambda,\ell+1}} \min\left(|\mathcal{C}_{ec}|^{-1}, 2^{k(\ell-\lambda+1)}\,|\mathcal{C}_{ec}|^{-2}\right) & \textit{für} \quad 1 \le \lambda \le \ell+1 \\ \binom{(m-1)(\ell+2)}{h}^{-1}|\mathcal{C}_{ec}|^{-2} & \textit{für} \quad \lambda = \ell+2 \end{array}$$

und $\theta_{\lambda,\ell}$ *als das maximale* $\alpha \in \mathbb{Z}$ *mit* $(\alpha+1)(\ell+2)+3+\log(\mathcal{L}(\alpha,\lambda)) < \log q$ *definiert ist.*

Resultate Tabelle 4.10 vergleicht Generator EC3 bei einer Güte von 2^{160} mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden für $q \approx 2^{160}$. Dabei bezeichnet *Mem* die Anzahl der gespeicherten Punkte und *Op* die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve.

Mem	m, k, ℓ, A, h	*Op*	Prec.-Methode	*Op*
8	1,6,13,22,4	53	de Rooij	59
16	1,16,7,19,2	36	de Rooij	52
32	2,29,4,16,1	26	BGMW-2	45
60	2,57,3,14,1	22	Lim-Lee	45.5
93	3,89,2,14,1	20	Lim-Lee	41
124	5,118,2,12,1	18	Lim-Lee	37

Tabelle 4.10: Vergleich von Generator EC3 mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

Verwendung des Frobenius-Homomorphismus

Bei der Verwendung elliptischer Kurven über $\mathbb{F}_{2^n}$, welche über einem kleinen Unterkörper definiert sind, bietet sich beim Generator EC2 die Möglichkeit die Verdoppelungen in Schritt 3 fast völlig einzusparen.

Definition 4.10 *Sei* E *eine elliptische Kurve über* $\mathbb{F}_{p^n}$*. Dann heißt der auf* E *durch* $\phi_{p^n}N(x,y) := (x^{p^n}, y^{p^n})$ *definierte Homomorphismus* Frobenius-Homomorphismus.

In der (bei elliptische Kurven üblichen) additiven Schreibweise der Gruppenverknüpfung erfüllt der Frobenius-Homomorphismus die Funktionalgleichung $\phi_{p^n}^2 - t\phi_{p^n} + p^n = 0$, wobei *t die Spur von* ϕ_{p^n} heißt und durch $|E(\mathbb{F}_{p^n})| = p^n + 1 - t$ gegeben ist (siehe [Men93, 103f.]).

Generator EC2 Im folgenden sei $n = z\ell'$ für ein ganzzahliges z, E eine elliptische Kurve über $\mathbb{F}_{2^{\ell'}}$ und G eine zyklische Untergruppe primer Ordnung von $E(\mathbb{F}_{2^n})$. Dann folgt (additiv geschrieben) $2^{\ell'}X = t\phi_{2^{\ell'}}(X) - \phi^2_{2^{\ell'}}(X)$ für alle $X \in G$. Mit Hilfe dieser Gleichung läßt sich in Schritt 3 von Generator EC2 die Exponentiation der Gruppenelemente mit $2^{\ell'}$ durch $|t|$ Additionen und 3 Anwendungen des Frobenius-Homomorphismus berechnen.

Das Quadrieren eines Körperelementes ist wesentlich weniger aufwendig, als z.B. eine Punktaddition (siehe [SOOS95] oder [MOVW88]). Daher fällt der Aufwand für die Berechnung des Frobenius-Homomorphismus gegenüber den Punkteverknüpfungen kaum ins Gewicht und kann vernachlässigt werden. Wir erhalten also insgesamt für den Generator EC2 einen Aufwand von $A + m + m|t| - 1$ Additionen und $\ell - 1$ Verdoppelungen.

Resultate Eine optimale Performance würden wir für eine Spur $t = 0$ erhalten. Allerdings kann der diskrete Logarithmus auf solchen Kurven in subexponentieller Zeit gelöst werden (siehe [MOV93]) – diese Kurven sind somit für kryptographische Zwecke ungeeignet. Im Fall $t = 1$ gilt $|E(\mathbb{F}_{2^n})| = 2^n$ und somit existiert keine große Untergruppe G von $E(\mathbb{F}_{2^n})$ mit primer Ordnung. Für unseren Generator ist der nächst-effizienteste Fall $t = -1$.

Tabelle 4.11 vergleicht für diesen Fall ($t = -1$) Generator EC2 mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden für $q \approx 2^{160}$. Die angegebenen Parameter liefern eine Sicherheit von ungefähr 2^{160} gegen Attacken beschränkter Länge. Mem bezeichnet die Anzahl der gespeicherten Punkte und Op die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve. Für $m = 1$ oder $m \geq 4$ ist der Generator weniger effizient.

Mem	$m = 2$		$m = 3$		Precomputation	
	k, ℓ, A	Op	k, ℓ, A	Op	Methode	Op
8	6,11,13	26	5,10,10	24	de Rooij	59
16	14,6,11	19	13,5,9	18	de Rooij	52
32	30,4,10	16	29,3,8	15	BGMW-2	45
60	58,3,9	14	57,2,7	13	Lim-Lee	45.5
93	92,2,9	13	90,2,6	12	Lim-Lee	41
124	122,2,8	12	121,1,7	12	Lim-Lee	37

Tabelle 4.11: Vergleich von Generator EC2 bei Verwendung des Frobenius-Homomorphismus mit Spur $t = -1$ mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

Generator EC3 Da bei Generator EC3 die Transformationen in Schritt 3 durch das „Square and multiply“ Verfahren durchgeführt werden, können wir dort die Quadrierschritte nicht durch den Frobenius-Homomorphismus ersetzen. Dies wird erst möglich, wenn wir $\boldsymbol{T}_\nu$ aus $\{0,1\}^{m-1}$ und $h < m-1$ wählen. In dieser Variante benötigt Generator EC3 bei Verwendung des Frobenius-Homomorphismus $\phi_{2^{\ell+2}}$ auf einer Kurve über $\mathbb{F}_{2^{z(\ell+2)}}$ pro Ausgabe $A+h+|t|$ Additionen und $\ell-1$ Verdoppelungen. In Satz 4.29 ist dafür der Ausdruck $\binom{(m-1)(\ell+2)}{h}$ durch $\binom{m-1}{h}$ zu ersetzen. Tabelle 4.12 vergleicht Generator EC3 bei $t=-1$ und einer Güte von 2^{160} mit den in Abschnitt 4.1 vorgestellten Precomputation-Methoden für $q \approx 2^{160}$. Dabei bezeichnet Mem die Anzahl der gespeicherten Punkte und Op die Anzahl der benötigten Operationen (d.h. Additionen und Verdoppelungen) auf der Kurve.

Mem	m, k, ℓ, A, h	Op	Prec.-Methode	Op
16	1,13,17,32,1	50	de Rooij	52
32	7,25,5,16,2	23	BGMW-2	45
60	8,52,3,14,1	18	Lim-Lee	45.5
93	6,87,2,14,1	17	Lim-Lee	41
124	6,118,2,13,1	16	Lim-Lee	37

Tabelle 4.12: Vergleich von Generator EC3 bei Verwendung des Frobenius-Homomorphismus mit Spur $t=-1$ mit den Precomputation-Methoden aus Abschnitt 4.1 für $q \approx 2^{160}$.

Bemerkung. Die Verwendung des Frobenius-Homomorphismus für die Exponentiation von Punkten auf Kurven, welche über einem kleinen Teilkörper definiert sind, wurde erstmals von Koblitz in [Kob92] vorgeschlagen. Diese Kurven sind daher auch als *Koblitz-Kurven* bekannt.

Nach dem heutigen Kenntnisstand ist das Problem des diskreten Logarithmus auf elliptischen Kurven über $\mathbb{F}_{2^n}$, die auf einem Unterkörper $\mathbb{F}_{2^{\ell'}}$ definiert sind, fast genauso schwer wie auf allgemeinen elliptischen Kurven. Die besten bekannten DL-Algorithmen auf solchen Kurven haben eine Laufzeit von $O(\sqrt{q/(2z)})$, wobei q der größte Primteiler von $E(\mathbb{F}_{2^n})$ und $z = n/\ell'$ der Grad der Körpererweiterung ist (siehe [GLV98] oder [WZ98]). In kryptographische Anwendungen sollte $z \geq 160$ gewählt werden, da $E(\mathbb{F}_{2^{\ell'}})$ eine Untergruppe von $E(\mathbb{F}_{2^n})$ ist und daher $q \leq |E(\mathbb{F}_{2^n})|/|E(\mathbb{F}_{2^{\ell'}})| \approx 2^z$ gilt.

Symbolverzeichnis

(a,b), $[a,b]$, $(a,b]$, $[a,b)$	Intervalle in $\mathbb{Z}$
$(a/b)_q$	Der betraglich kleinste Repräsentant von $a/b \bmod q$, **71**
$\equiv$	Gleichheit, unabhängig von $\boldsymbol{r}$ und $\boldsymbol{s}$, **60**
$\langle \boldsymbol{a}, \boldsymbol{b} \rangle$	Das Standardskalarprodukt von $\boldsymbol{a}$ und $\boldsymbol{b}$
$\lfloor x \rfloor$	Die größte ganze Zahl kleiner x
$\lceil x \rceil$	Die kleinste ganze Zahl größer x
$\lVert \boldsymbol{a} \rVert_\infty$	Die Maximums-Norm von $\boldsymbol{a}$
$\otimes$	Tensorprodukt, **37**
$\mathcal{C}$	Die Menge der möglichen Münzwürfe des Generators, **60**
$\mathcal{C}_{ec}$	Die Menge der möglichen Münzwürfe des Generators für elliptische Kurven, **89**
$d_{\boldsymbol{d}}(\boldsymbol{f})$	Der bei RSA-S1 durch $\boldsymbol{f}$ und $\boldsymbol{d}$ gegebene Schlüssel d, **23**
$d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g})$	Der bei RSA-S1M durch $\boldsymbol{f}$, $\boldsymbol{g}$ und $\boldsymbol{d}$ gegebene Schlüssel d, **31**
$\mathrm{dg}(v)$	Der Grad des Knoten v, **40**
$\mathrm{dg}_{\boldsymbol{f}}(v)$	Die Anzahl der zu v inzidenten Kanten mit f-Farbe $\boldsymbol{f}$, **42**
$\mathbb{E}$	Erwartungswert
$E(\mathbb{F})$	Die Gruppe der $\mathbb{F}$-rationalen Punkte der Kurve E, **87**
$E(U_1, U_2)$	Die Menge der Kanten von U_1 nach U_2, **42**
$\overline{\mathbb{F}}$	Der algebraische Abschluß von $\mathbb{F}$
$\mathbb{F}_{p^n}$	Der Körper mit p^n Elementen
$\boldsymbol{f}^{v,w}$, $\boldsymbol{g}^{v,w}$	Die f-Farbe bzw. g-Farbe der Kante (v,w), **38**
ggT	Der größte gemeinsame Teiler
kgV	Das kleinste gemeinsame Vielfache
$\mathcal{L}(n, \lambda)$	Das Minimum von $2^{n\lambda}$ und $\mathrm{kgV}(2, 3, \ldots, 2^\lambda - 1)$, **71**
$\lambda_{i, \lVert\cdot\rVert}$	Das i-te sukksessive Minimum bezüglich $\lVert \cdot \rVert$, **14**
$\mathcal{M}(v)$	Die Menge der zu v adjazenten, nicht f-dominierten Knoten, **42**
$\mathcal{N}(v)$	Die Menge der zu v adjazenten Knoten, **42**
$\mathcal{O}$	Der Unendlichkeitspunkt einer elliptischen Kurve, **87**
$O(f)$, $\Omega(f)$	Groß–O Notation, Groß–Omega Notation
$\phi(n)$	Die Euler'sche ϕ-Funktion, **7**

ϕ_{p^n}	Der Frobenius-Homomorphismus, **93**
$\mathrm{S}(\boldsymbol{c})$	Die Erfolgswahrscheinlichkeit einer einfachen Attacke $\boldsymbol{c}$, **60**
$\mathrm{S}(\boldsymbol{c}, \boldsymbol{r})$	Die öffentliche Erfolgswahrscheinlichkeit von $\boldsymbol{c}$, **64**
$S_{\boldsymbol{d}}$	Die Menge der $\boldsymbol{f}$ mit $d_{\boldsymbol{d}}(\boldsymbol{f}) \in \mathbb{Z}^*_{\phi(n)}$, **23**
$S'_{\boldsymbol{d}}$	Die Menge der $(\boldsymbol{f}, \boldsymbol{g})$ mit $d_{\boldsymbol{d}}(\boldsymbol{f}, \boldsymbol{g}) \in \mathbb{Z}^*_{\phi(n)}$, **31**
SEC	Die Güte des Generators, **60**
$\mathrm{SEC}_{\mathrm{p}}(\mathbf{r})$	Die öffentliche Güte des Generators, **64**
$\mathrm{sgn}(x)$	Das Vorzeichen von x
$V^{\mathrm{f}}(\boldsymbol{f})$, $V^{\mathrm{g}}(\boldsymbol{g})$	Die Menge der Knoten $v \in V$ mit f-Farbe $\boldsymbol{f}$ bzw. g-Farbe $\boldsymbol{g}$, **39**
$\mathrm{wt}(\boldsymbol{x})$, $w(a)$	Das Hamminggewicht von $\boldsymbol{x}$ bzw. a, **22**
$Y(N)$	Die Menge der $(a_1, a_2) \in \mathbb{Z}^2$, für die $a_1x_1 + a_2x_2 = 0 \bmod q$ eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^2$ mit $\\|\boldsymbol{x}\\|_\infty < N$ besitzt, **71**
$Y_3(N)$	Die Menge der $(a_1, a_2, a_3) \in \mathbb{Z}^3$, für die $a_1x_1 + a_2x_2 + a_3x_3 = 0 \bmod q$ eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^3$ mit $\\|\boldsymbol{x}\\|_\infty < N$ besitzt, **74**
$Y_t(N)$	Die Menge der $(a_1, \ldots, a_t) \in \mathbb{Z}^t$, für die $a_1x_1 + \cdots + a_tx_t = 0 \bmod q$ eine nichttriviale Lösung $\boldsymbol{x} \in \mathbb{Z}^t$ mit $\\|\boldsymbol{x}\\|_\infty < N$ besitzt, **79**
$\mathbb{Z}^*_n$	Die Gruppe der Einheiten in $\mathbb{Z}_n$

Index

Literaturverzeichnis

[Ajt97] Miklós Ajtai. The shortest vector problem in L_2 is NP–hard for randomized reductions. Technical Report 97-047, Electronic Colloquium on Computational Complexity, 1997. To appear in the Proceedings of STOC'98.

[And92] R. J. Anderson. Attack on server assisted authentication protocols. *Electronics Letters*, 28(15):1473, 1992.

[BGMW92] E. F. Brickell, D. M. Gordon, K. S. Mc Curley, and D. Wilson. Fast exponentiation with precomputation. In *Advances in Cryptology - Proceedings of Eurocrypt'92*, volume 658 of *Lecture Notes in Computer Science*, pages 200–207. Springer Verlag, 1992.

[BL96] D. Boneh and R. Lipton. Algorithms for black-box fields and their application to cryptography. In *Advances in Cryptology - Proceedings of Crypto'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 283–297. Springer Verlag, 1996.

[BM82] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23th Annual Symposium on Foundations of Computer Science - FOCS'82*, pages 112–117, 1982.

[BM94] J. Burns and C.J. Mitchell. Parameter selection for server-aided RSA computation schemes. *IEEE Transactions on Computers*, 43(2):163–174, 1994.

[BMB] Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie. *Informations- und Kommunikationsdienste-Gesetz.*

[Bol78] B. Bollobas. *Extremal graph theory*, volume 11 of *L.M.S. Monographs*, page 158. Academic Press. XX, London, 1978.

[BPV98] V. Boyko, M. Peinado, and R. Venkatesan. Speeding up discrete log and factoring based schemes via precomputations. In *Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 221–235. Springer Verlag, 1998.

[BQ95] P. Béguin and J. J. Quisquater. Fast server-aided rsa signatures secure against active attacks. In *Advances in Cryptology - Proceedings of Crypto'95*, volume 963 of *Lecture Notes in Computer Science*, pages 57–69. Springer Verlag, 1995.

[BS84] L. Babai and E. Szemerédi. On the complexity of matrix group problems I. In *25th Annual Symposium on Foundations of Computer Science - FOCS'84*, pages 229–240, 1984.

[BS91] T. Beth and F. Schäfer. Non supersingular elliptic curves for public key cryptosystems. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 316–327. Springer Verlag, 1991.

[Cas71] J.W.S. Cassels. *An Introduction to the geometry of numbers.* Classics in Mathematics. Springer Verlag, 1971. Wiederveröffentlicht 1997.

[CJL+92] M. Coster, A. Joux, B. LaMacchia, A.M. Odlyzko, C.P. Schnorr, and J. Stern. Improved low–density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[CR88] B. Chor and R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Information Theory*, 34:901–909, 1988.

[CvA90] D. Chaum and H. van Antwerpen. Undeniable signatures. In *Advances in Cryptology - Proceedings of Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer Verlag, 1990.

[DH76] W. Diffie and M.E. Hellmann. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[dR91] P. de Rooij. On the security of the Schnorr Scheme using preprocessing. In *Advances in Cryptology - Proceedings of Eurocrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 71–80. Springer Verlag, 1991.

[dR93] P. de Rooij. On Schnorr preprocesing for digital signature schemes. In *Advances in Cryptology - Proceedings of Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 435–449. Springer Verlag, 1993.

[dR94] P. de Rooij. Efficient exponentiation using precomputation and vector addition chains. In *Advances in Cryptology - Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 389–399. Springer Verlag, 1994.

[dR97] P. de Rooij. On Schnorr's preprocessing of digital signature schemes. *Journal of Cryptology*, 10(1):1–16, 1997.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[FS87] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology - Proceedings of Crypto'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1987.

[GL69] P.M. Grubner and C.G. Lekkerkerker. *Geometry of numbers.* North Holand Mathematical Library. North Holland, 1969.

[GLV98] R. Gallant, R. Lambert, and S. Vanstone. Improving the parallelized pollard lambda search on binary anomalous curves. Preprint, 1998.

[GUQ92] L.C. Guillou, M. Ugon, and J.J. Quisquater. The samrt card: a standardized security device dedicated to public cryptology. In G.J. Simmons, editor, *Contemporary Cryptology: The Science of Information Integrity*, pages 561–613. IEEE Press, 1992.

[HCY95] S.J. Hwang, C.C. Chang, and W.P. Yang. Some active attacks on fast server-aided secret computation protocols for modular expontiation. In *Proceedings of Crptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 215–227. Springer Verlag, 1995.

[Hör94] H. H. Hörner. Verbesserte Gitterbasenreduktion; getestet am Chor–Rivest Kryptosystem und an allgemeinen Rucksack–Problemen. Master's thesis, Universität Frankfurt, Fachbereich Mathematik, 1994.

[Hor98] G. Horng. An active attack on protocols for server-aided RSA signature computation. *Information Processing Letters*, 65:71–73, 1998.

[HP98] H. Handschuh and P. Paillier. Smart card crypto-coprocessors for public-key cryptography. *CryptoBytes*, 4(1):6–10, 1998.

[Kal87] B. Kaliski. A pseudo-random bit generator based on elliptic logarithms. In *Advances in Cryptology - Proceedings of Crypto'86*, volume 293 of *Lecture Notes in Computer Science*, pages 84–103. Springer Verlag, 1987.

[Kob84] N. Koblitz. *Intoduction to elliptic curve and modular forms.* Springer Verlag, Berlin, 1984.

[Kob92] N. Koblitz. CM-curves with good cryptographic properties. In *Advances in Cryptology - Proceedings of Crypto'91*, volume 576 of *Lecture Notes in Computer Science*, pages 279–287. Springer Verlag, 1992.

[KS93] S. Kawamura and A. Shimbo. Fast server-aided secret computation protocols for modular exponentiation. *IEEE Journal on selected areas in communications*, 11(5):778–784, 1993.

[LL94] C. H. Lim and P. J. Lee. More flexible exponentiation with precomputation. In *Advances in Cryptology - Proceedings of Crypto'94*, volume 839 of *Lecture Notes in Computer Science*, pages 95–105. Springer Verlag, 1994.

[LL95] C. H. Lim and P. J. Lee. Security and performance of server-aided RSA computation protocolls. In *Advances in Cryptology - Proceedings of Crypto'95*, volume 963 of *Lecture Notes in Computer Science*, pages 70–83. Springer Verlag, 1995.

[LLL82] A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

[LLMP90] A. K. Lenstra, H. W. Lenstra, M. Manasse, and J. M. Pollard. The number field sieve. In *Proceedings 22nd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 564–572, 1990.

[LMMS94] F. Lehmann, M. Maurer, V. Müller, and V. Shoup. Counting the number of points on elliptic curves over finite fields of characteristic greater than three. In *Proceedings of the 1st International Algorithmic Number Theory Symposium (ANTS-I)*, volume 877 of *Lecture Notes in Computer Science*, pages 60–70. Springer Verlag, 1994.

[Lub96] M. Luby. *Pseudorandomness and cryptographic applications*, page 86. Princeton Computer Science notes. Princeton University Press, 1996.

[LZ94] G.J. Lay and H.G. Zimmer. Constructing elliptic curves with given group order over large finite fields. In *Proceedings of the 1st International Algorithmic Number Theory Symposium (ANTS-I)*, volume 877 of *Lecture Notes in Computer Science*, pages 250–263. Springer Verlag, 1994.

[Men93] A.J. Menezes. *Elliptic curve public key cryptosystems.* Kluwer Academic Publishers, 1993.

[Mer95] J. Merkle. Über Schnorrs Preprocessing für diskrete LOG-Unterschriften. Master's thesis, Universität Frankfurt, 1995.

[MH78] R. Merkle and M. Hellmann. Hidden information and signatures in trapdoor knapsack. *IEEE Trans. Information Theory*, 24:525–530, 1978.

[MILY93] T. Matsumoto, H. Imai, C. S. Laih, and S. M. Yen. On verifiable implicit asking protocols for RSA computation. In *Advances in Cryptology - Proceedings of Auscrypt'92*, volume 718 of *Lecture Notes in Computer Science*, pages 296–307. Springer Verlag, 1993.

[MKI89] T. Matsumoto, K. Kato, and H. Imai. Speeding up computation with insecure auxiliary devices. In *Advances in Cryptology - Proceedings of Crypto'88*, volume 403 of *Lecture Notes in Computer Science*, pages 497–506. Springer Verlag, 1989.

[MO90a] J.E. Mazo and A.M. Odlyzko. Lattice points in high-dimensional spheres. *Monatsh. Math.*, 110(1):47–61, 1990.

[MO90b] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *Informatique Théoretique et Applications*, 24:531–543, 1990.

[MOV93] A.J. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, IT-39:1639–1646, 1993.

[MOVW88] R. Mullin, I. Onyszchuk, S. Vanstone, and R. Wilson. Optimal normal bases in $GF(p^n)$. *Discrete Applied Mathematics*, 22:149–161, 1988.

[MP97] V. Müller and S. Paulus. On the generation of cryptographically strong elliptic curves. Preprint, 1997.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*, pages 43–45. Cambridge University Press, 1995.

[MS98] J. Merkle and C. P. Schnorr. Perfect, generic pseudo-randomness for cyclic groups. Unpublished, 1998.

[MvOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, Boca Raton, 1997.

[MVZ93] A. Menezes, S. Vanstone, and R. Zuccherato. Counting points on elliptic curves over $\mathbb{F}_{2^m}$. *Mathematics of Computation*, 60(201):407–420, 1993.

[MW98] J. Merkle and R. Werchner. On the security of server-aided RSA protocols. In *Proceedings of the first International Workshop on Practice and Theory in Public Key Cryptography - PKC'98*, volume 1431 of *Lecture Notes in Computer Science*, pages 99–116. Springer Verlag, 1998.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[NIS94] National Institute of Standards and Technology, NIST FIPS PUB 186. *Digital signature standard (DSS)*, 1994.

[NM96] D. Naccache and D. M'Raihi. Cryptographic smart cards. *IEEE Micro*, 16(3):14–24, 1996.

[NS99] P. Nguyen and J. Stern. On the hardness of the hidden subset sum problem and its cryptographic implications. In *Advances in Cryptology - Proceedings of Crypto'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 31–46. Springer Verlag, 1999.

[OS90] H. Ong and C.P. Schnorr. Fast signature generation with a Fait Shamir-like scheme. In *Advances in Cryptology - Proceedings of Eurocrypt'90*, volume 473 of *Lecture Notes in Computer Science*, pages 432–440. Springer Verlag, 1990.

[PKC93] RSA Data Security Inc. *The Public-Key Cryptography Standards (PKCS)*, 1993.

[PW93] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. In *Advances in Cryptology - Proceedings of Eurocrypt'92*, volume 658 of *Lecture Notes in Computer Science*, pages 153–162. Springer Verlag, 1993.

[Rab79] M.O. Rabin. Digitized signatures and public-key functions as intractible as factorization. Technical Report LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[RE96] W. Rankl and W. Effing. *Handbuch der Chipkarten. Aufbau - Funktionsweise - Einsatz von Smart Cards.* Hanser Elektronik, 1996.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communication of the ACM*, 21:120–126, 1978.

[Sch75] A. Schönhage. A lower bound for the length of addition chains. *Theoretical Computer Science*, 1:1–12, 1975.

[Sch87] C.P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.

[Sch89] C. P. Schnorr. Efficient signature generation for smart cards. In *Advances in Cryptology - Proceedings of Crypto'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer Verlag, 1989.

[Sch91] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch94] C.P. Schnorr. Block reduced lattice bases and successive minima. *Combinatorics, Probability and Computing*, 3:507–522, 1994.

[Sch95] C.P. Schnorr. Gittertheorie und algorithmische Geometrie, Reduktion von gitterbasen und polynomidealen. Lecture Note, 1995.

[Sch98] C. P. Schnorr. Security of arbitrary RSA and of all discrete log bits. Unpublished, 1998.

[SE94] C.P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994.

[SH95] C.P. Schnorr and H.H. Hörner. Attacking the chor–rivest cryptosystem by improved lattice reduction. In *Advances in Cryptology – Proceedings of EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, 1995.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - Proceedings of Eurocrypt'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer Verlag, 1997.

[SOOS95] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. In *Advances in Cryptology - Proceedings of Crypto'95*, volume 963 of *Lecture Notes in Computer Science*, pages 43–56. Springer Verlag, 1995.

[Ste95] J. Stern. Analysis of a preprocessing algorithm for DSS signatures. Unpublished, 1995.

[Sti95] D. R. Stinson. *Cryptography: Theory and practice*. CRC Press, 1995.

[vE81] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Dept. of Mathematics, University of Amsterdam, 1981.

[vOW96] P. van Oorschot and M. Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In *Advances in Cryptology - Proceedings of Crypto'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236. Springer Verlag, 1996.

[Wie90] M. J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36:553–558, 1990.

[WZ98] M. Wiener and R. Zuccherato. Faster attacks on elliptic curve cryptosystems. To appear in the proceedings of Selected Areas in Cryptography 1998 (LNCS), 1998.

Johannes Merkle
Kurhausstr.62
65719 Hofheim

Lebenslauf

Persönliche Daten	
Geburtsdatum/-ort	27.4.1968 in Kelkheim/Ts.
Familienstand	Ledig
Staatsangehörigkeit	Deutsch
Schulausbildung	
1980 – 1984	Gesamtschule am Rosenberg, Hofheim
1984 – 1987	Gymnasiale Oberstufe, Main–Taunus–Schule, Hofheim
Juni 1987	Abitur
Hochschulausbildung	
1988 – 1995	Studium der Mathematik an der Johann Wolfgang Goethe-Universität, Frankfurt/M., Nebenfach: theoretische Physik
August 1990	Diplom-Vorprüfung
Februar 1995	Diplom-Hauptprüfung Diplomarbeit: „Über Schnorrs Preprocessing für diskrete Log–Unterschriften“, Betreuer: Prof. Dr. C. P. Schnorr
1995 - 1999	Erstellung der Dissertation „Effiziente Signaturerzeugung durch Server-Unterstützung und Vorberechnung“, Betreuer: Prof. Dr. C. P. Schnorr
Berufliche Stationen	
1995 – 1998	Anstellung als wissenschaftlicher Mitarbeiter am Fachbereich Mathematik der Universität Frankfurt/M.
seit Juni 1998	Berater bei der Secunet AG in Eschborn

Akademische Lehrer: R. Bieri, H. F. de Groote, R. Jellito, W. Metzler, K. H. Müller, C. P. Schnorr