Efficient Signature Generation by Smart Cards*

C.P. Schnorr

Fachbereich Mathematik / Informatik

Universität Frankfurt

March 1991

1 Introduction.

Public key signature schemes are necessary for the access control to communication networks and for proving the authenticity of sensitive messages such as electronic fund transfers. Since the invention of the RSA–scheme by Rivest, Shamir and Adleman (1978) research has focused on improving the efficiency of these schemes. In this paper we present an efficient algorithm for generating public key signatures which is particularly suited for interactions between smart cards and terminals.

The new signature scheme minimizes the message dependent amount of computation the smart card has to perform to generate a signature. This is important since the computational power of current processors for smart cards is rather limited. Previous signature schemes require many modular multiplications for signature generation. In the new scheme the main work for signature generation does not depend on the message and can be done during the idle time of the processor. The message dependent part of signature generation consists of multiplying a 140 bit integer with a 72 bit integer.

*european patent application 89103290.6 from 24.2.1989

Our signature scheme relies on the interactive protocol of Chaum, Evertse, Graaf (1988) that proves possession of a discrete logarithm. It combines various ideas from the schemes by ElGamal (1985) and Fiat–Shamir (1986). It is derived from an underlying interactive authentication scheme by replacing the verifier's challenge by a hash value. The novel features of our scheme can be incorporated into the Beth authentication scheme and into the key distribution scheme by Günther (1989). The new scheme comprises the following novel features:

(1) Most of the computational effort for signature generation is done in a preprocessing stage that is independent of the message and can be done during the idle time of the processor. The preprocessing consists of the exponentiation of a random number modulo a large prime. Given this exponentiated residue a signature can be generated very fast, it requires only the multiplication of a 72 bit integer with a 140 bit integer. The idea of preprocessing signatures is similar in spirit to the concept of on–line/off-line signatures that has been independently proposed by Even, Goldreich and Micali (1989).

(2) We use a prime modulus p with $p-1$ having a prime factor q of appropriated size (e.g. 140 bits long) and we use a base α for the discrete logarithm such that $\alpha^q = 1 \pmod{p}$. All logarithms are calculated modulo q. The length of signatures is about 212 bits, it is less than half of the length of RSA signatures. The number of communication bits of the authentication scheme is less than half of that of other schemes.

(3) We propose an efficient algorithm for simulating the exponentiation of random numbers. This algorithm is independent of the rest of the paper. If proven to be secure our algorithm reduces the amount of computation for generating random exponentiated residues by using additional memory for storing some statistically independent exponentiated residues.

The security of the scheme relies on the one–way property of the exponentiation $y \mapsto \alpha^y \pmod{p}$. We therefore have to assume that discrete logarithms with base α are difficult to compute.

The paper is organized as follows. We present in section 2 a version of the signature scheme and of the underlying authentication scheme that uses exponentiation of a random integer. The performance of the scheme is exemplified in section 3. In section 4 we propose an efficient algorithm that simulates the exponentiation of a random number.

2 The Authentication And Signature Scheme.

Notation. For $n \in \mathbb{N}$ let $\mathbb{Z}_n$ be the ring of integers modulo n. We identify $\mathbb{Z}_n$ with the set of integers $\{1, \dots, n\}$.

Initiation of the key authentication center (KAC). The KAC chooses

- primes p and q such that $q \mid p-1$, $q \geq 2^{140}$, $p \geq 2^{512}$,
- $\alpha \in \mathbb{Z}_p$ with order q, i.e. $\alpha^q = 1 \pmod p$, $\alpha \neq 1$,
- a one–way hash function $h : \mathbb{Z}_q \times \mathbb{Z} \to \{0, \dots 2^t - 1\}$,
- its own private and public key.

The KAC publishes p, q, α, h and its public key.

The security complexity 2^t. We wish to choose the parameters p, q so that forging a signature or an authentication requires by known methods about 2^t steps. For this we choose $q \geq 2^{2t}$ and p such that 2^t is about $e^{\sqrt{\ln p \ln \ln p}}$. The *security number* t may depend on the application intended. For *signatures* we consider in particular $t = 72$ rather that $t = 64$, since 2^{64} steps may be insufficient in view of the rapid technological progress in computing power and speed. For $p \geq 2^{512}$ and $q \geq 2^{140}$ the discrete logarithm problem requires at least 2^{72} steps by known algorithms. (It may soon be necessary to increase the lower bound $p \geq 2^{512}$ due to the current progress in computing discrete logarithms.) The restriction that the order of α is a prime much smaller than p does provide no advantage in any of the known discrete logarithm algorithms provided that $q \geq 2^{140}$. The prime q is necessary in order to avoid an index calculus attack and a square root attack (see section 2). A lower security level may be sufficient for *authentication* in particular if the prover is requested to respond fast, say within a few seconds. A security complexity 2^{40} for authentication requires to choose $t \geq 40$, $p \geq 2^{256}$ and $q \geq 2^{80}$.

Registration of users, signatures by the KAC. When a user comes to the KAC for registration the KAC verifies its identity, generates an identification number I (containing name, address, ID–number etc.) and generates a signature S for the pair (I, v) consisting of I and the user's public key v. In our scheme as well as in the RSA–scheme each user may produce by himself his own private key s and the corresponding public key v. It is necessary to complete the corresponding public information (I, v) by a signature S of a trusted authority, the KAC. The verification of a signature or an authentification with the public key v must also contain a verification of the public key v. This verification can either be done on–line by reading (I, v) from a public file or off–line by verifiying KAC's signature S for (I, v) using the public key of the KAC. In an interaction between two smart cards the verification of v is always off–line.

The KAC can use for its own signatures any secure public key signature scheme whatsoever. For instance the KAC can use our scheme which yields short signatures that can be verified using about 228 modular multiplications. Alternatively the KAC can base its signature S for (I, v) on the identity $S^2 = h(j, I, v) \pmod N$ where N is a public RSA–modulus, h a one–way hash function and j a small integer. Only the KAC can generate such a signature S using the secret factorization of N. The verification of this signature S requires only one modular squaring. — This is the same amount of computation that is necessary for the verification of the public key in the identity based Fiat–Shamir scheme.

The user's private und public key. A user generates by himself a private key s which is a random number in $\{1, 2, \ldots, q\}$. The corresponding public key v is the number $v = \alpha^{-s} \pmod p$.

Once the private key s has been chosen one can easily compute the corresponding public key v. The inverse process, to compute s from v, requires to compute the discrete logarithm with base α of v^{-1}, i.e. $s = -log_\alpha v$.

The following authentication protocol is essentially equal to protocol 1 in Chaum, Evertse, Graaf (1988). Their protocol 1 is the particular case $t = 0$ and $q = p - 1$ of the protocol below which for its part is a parallel variant for t sequential rounds of their protocol 1. Chaum et alii prove that their protocol 1 is zero–knowledge, i.e. it does not reveal any information on the secret s. The parallel variant of the Chaum et alii protocol is not known to be zero–knowledge.

The authentication protocol.
(Prover A proves its identity to verifier B)

1. *Preprocessing* (see section 4). A picks a random number $r \in \{1, \dots, q-1\}$, and computes $x := \alpha^r (\bmod p)$.
2. *Initiation.* A sends to B its identication string I, its public key v, the KAC's signature S for (I, v), and x.
3. B verifies the signature S and sends a random number $e \in \{0, \dots, 2^t - 1\}$ to A.
4. A sends to B $y := r + se \pmod q$.
5. *Verification.* B verifies (I, v) either by checking the signature S or by verifying (I, v) on–line. B checks that $x = \alpha^y v^e \pmod p$.

Obviously if A und B follow the protocol then B always accepts A's proof of identity. We next consider the possibilities of cheating for A and B. We call (x, y) the *proof* and e the *exam* of the authentication. Let $\widetilde{A}$ ($\widetilde{B}$, resp.) denote a fraudulent A (B, resp.). $\widetilde{A}$ ($\widetilde{B}$, resp.) may deviate from the protocol in computing x, y (e, resp.). $\widetilde{A}$ does not know the secret s. $\widetilde{B}$ can spy upon A's method of authentication.

A fraudulent $\widetilde{A}$ can cheat by guessing the correct e and sending with an arbitrary $r \in \mathbb{Z}_q$ the crooked proof

$$x := \alpha^r v^e \pmod p, \quad y := r \ .$$

The probability of success for this attack is 2^{-t}.

By the following theorem this success rate cannot be increased unless computing $\log_\alpha v$ is easy. For this let $\widetilde{A}$ be any probabilistic, interactive algorithm (Turing machine) that is given the fixed values p, q, α. Let RA

denote the internal random bit string of $\widetilde{A}$. Let the success bit $S_{\tilde{A},v}(RA,e)$ be 1 if $\widetilde{A}$ succeeds with v, RA, e and 0 otherwise. The *success rate* $S_{\tilde{A},v}$ of $\widetilde{A}$ for v is the average of $S_{\tilde{A},v}(RA,e)$, where RA, e are chosen at random with uniform distribution. We assume that the time $T_{\tilde{A},v}(RA,e)$ of $\widetilde{A}$ with v, RA, e is independent of RA and e, i.e. $T_{\tilde{A},v}(RA,e) = T_{\tilde{A},v}$. This is no restriction since limiting the time to twice the average running time for successful pairs (RA,e) decreases the success rate at most by a factor 2.

Theorem 2.1 *There is a probabilistic algorithm AL which on input* $(\widetilde{A}, v)$ *computes* $\log_\alpha v$. *If the success rate* $S_{\tilde{A},v}$ *of* $\widetilde{A}$ *with* v *is greater than* 2^{-t+1} *then AL runs in expected time* $O(T_{\tilde{A},v} \,/\, S_{\tilde{A},v})$ *where* $T_{\tilde{A},v}$ *is the time of* $\widetilde{A}$ *on input* v.

Proof. The argument extends Theorem 5 in Feige, Fiat, Shamir (1987). We assume that the time $T_{\tilde{A},v}$ also covers the time required for B.

Algorithm AL with input v.

1. Pick RA at random. Compute $x = x(\widetilde{A}, RA, v)$, i.e. compute x the same way as does algorithm $\widetilde{A}$ using the coin tossing sequence RA. Pick a random $e \in \{0, \dots, 2^t - 1\}$. Compute $y := y(\widetilde{A}, RA, v, e)$ the same way as algorithm $\widetilde{A}$. If $S_{\tilde{A},v}(RA,e) = 1$ then fix RA, retain x, y, e and go to 2. Otherwise repeat step 1 using an independent RA.
2. Let u be the number of probes (i.e. passes of step 1) in the computation of RA, x, y, e. Probe up to $4u$ random $\overline{e} \in \{0, \dots, 2^t - 1\}$ whether $S_{\tilde{A},v}(RA, \overline{e}) = 1$. If some 1 occurs with $\overline{e} \neq e$ then compute the corresponding $\overline{y} = \overline{y}(\widetilde{A}\,,\, RA\,,\, \overline{e}\,,\, v\,)$ and *output* $\log_\alpha v := \frac{y - \overline{y}}{\overline{e} - e} \pmod q$.

Time analysis. Let $S_{\tilde{A},v} > 2^{-t+1}$. We arrange for fixed $\widetilde{A}$ and v the success bits $S_{\tilde{A},v}(RA,e)$ in a matrix with rows RA and columns e. A row RA is called *heavy* if the fraction of 1–entries is at least $S_{\tilde{A},v} \,/\, 2$. At least half of the 1–entries are in heavy rows since the number of 1–entries in non–heavy rows is at most $S_{\tilde{A},v} \cdot \#\text{rows} \cdot \#\text{columns} / 2$. Thus the row RA that succeeds in step 1 is heavy with probability at least 1/2. A heavy row has at least two 1–entries.

We abbreviate $\varepsilon = S_{\tilde{A},v}$. The probability that step 1 probes $i\,\varepsilon^{-1}$ random RA without finding an 1–entry is at most $(1-\varepsilon)^{i/\varepsilon} < 2.7^{-i}$. Thus the average number of probes for the loop of step 1 is

$$\leq \sum_{i=1}^{\infty} i\,\varepsilon^{-i}\, 2.7^{-i+1} \quad = \quad O(\varepsilon^{-1})\ .$$

We have with probability at least 1/2 that $u \geq \varepsilon^{-1}/2$. The row RA is heavy with probability at least 1/2. If these two cases happen then step 2 finds a successful $\overline{e}$ with probability $\geq 1-(1-\varepsilon/2)^{2/\varepsilon} > 1-2.7^{-1}$, and we have $e \neq \overline{e}$ with probability $\geq 1/2$. Thus AL terminates after one iteration of steps 1 and 2 with probability

$$\geq \frac{1}{4}(1-2.7^{-1})\frac{1}{2} \quad > \quad 0.07.$$

The probability that AL performs exactly i iterations is at most 0.93^{i-1}. Alltogether we see that the average number of probes for AL is at most

$$O\left(5\,\varepsilon^{-1} \sum_{i=0}^{\infty} 0.93^{i-1}\, t\right) \;=\; O(\varepsilon^{-1}).$$

This proves the claim. **Q.E.D.**

The above proof shows that two authentications with the same x and distinct challenges $e, \overline{e}$ together reveal the secret s.

The argument above can be extended to show that the authentication protocol is a proof of knowledge, in the sense of Feige, Fiat, Shamir (1987), showing that user A knows $s = \log_\alpha v$.

The verifier B is free to choose the bit string e in step 3 of the authentication protocol, thus he can choose e in order to spy upon A's method for authentication. The informal (but non rigorous) reason that A reveals no information is that the numbers x and y are random. The random number x reveals no information. It is unlikely that the number y reveals any useful information because y is superposed by the discrete logarithm of x, $y = \log_\alpha x + es \pmod q$ and the cryptanalyst cannot infer $r = \log_\alpha x$ from x. The scheme is not zero–knowledge because the triple (x, y, e) may be a particular solution of the equation $x = \alpha^y v^e \pmod p$ due to the fact that the choice of e may depend on x.

Minimizing the number of communication bits. Using a hash function h we can reduce the amount of communication for authentication. A can send in step 2 the t bit string $h(x) = h(x, 0)$ instead of x and B computes in step 5 $\overline{x} := \alpha^y v^e \pmod p$ and checks that $h(x) = h(\overline{x})$. It is not necessary that h is a one–way function because $x = \alpha^r \pmod p$ is already the result of a one–way function. To achieve the security level 2^t the bit string $h(x)$ must be at least t bits long. No particular attack is known for the function $h(x)$ consisting of the t least significant bits of x. The number of communication bits is $2t + 140$ plus the bits for (I, v) and S. The pair $(y, h(x))$ is a signature of the empty message with respect to the following signature scheme.

Protocol for signature generation.

To sign message m with the private key s perform the following steps:

1. *Preprocessing* (see section 3). Pick a random number $r \in \{1, \dots, q\}$ and compute $x := \alpha^r \pmod p$.
2. Compute $e := h(x, m) \in \{0, \dots, 2^t - 1\}$.
3. Compute $y := r + se \pmod q$ and output the *signature* (e, y).

Protocol for signature verification.

To verify the signature (e, y) for message m with public key v compute $\overline{x} = \alpha^y v^e \pmod p$ and check that $e = h(\overline{x}, m)$.

A signature (e, y) is accepted if it withstands verification. A signature generated according to the protocol is always accepted since we have

$$x = \alpha^r = \alpha^{r+se} v^e = \alpha^y v^e \pmod{p} .$$

With $t = 72$ and $q \approx 2^{140}$ the signature (e, y) is 212 bits long.

Efficiency. The work for signature generation consists mainly of the preprocessing (see section 3) and the computation of $se \pmod{q}$ where the numbers s and e are about 140 and $t = 72$ bits long. The latter multiplication is negligible compared to a modular multiplication in the RSA–scheme.

Signature verification consists mainly of the computation of $\overline{x} = \alpha^y v^e \pmod{p}$ which can be done on the average using $1.5\ell + 0.25t$ multiplications modulo p where $\ell = \lceil \log_2 q \rceil$ is the bit length of q. For this let y and e have the binary representations

$$y = \sum_{i=0}^{\ell-1} y_i \, 2^i \ , \quad e = \sum_{i=0}^{\ell-1} e_i \, 2^i \quad \text{with } y_i, e_i \in \{0,1\} \ , \quad e_i = 0 \quad \text{for } i \geq t \ .$$

We compute αv in advance and we obtain $\overline{x}$ as follows

1. $z := 1$,
2. $z := z^2 \alpha^{y_i} v^{e_i} \pmod{p}$ for $i = \ell - 1, \ldots, 1$,
3. $\overline{x} := z$.

This computation requires at most $\ell + t - 1 + \sum_{i=t}^{\ell} y_i$ modular multiplications. If half of the bits y_i with $i \geq t$ are zero, and $e_i = y_i = 0$ holds for one fourth of the $i < t$, then there are at most $\ell + 0.5(\ell - t) + 0.75t = 1.5\ell + 0.25t$ modular multiplications.

Coexistence of the authentication and the signature scheme. Some precaution has to be taken if the authentication and the signature scheme are both used with the same α and p. In this case it is incorrect to transmit the entire witness x in the authentication protocol. This is because the verifier may pose as exam e the hashing $h(x, m)$ of witness x with any message m. Then the proof of identity (x, y) yields a signature for message m. This attack can be thwarted by transmitting in the authentication protocol instead of x a hash value (e.g. 72 bits) of x.

The choice of the prime q. The prime q must be at least 140 bits long in order to sustain a security level of 2^{72} steps. This is because $\log_\alpha(x) \in \{1, \ldots, q\}$ can be found in $O(\sqrt{q})$ steps by the baby step giant step method. In order to compute $u, v \leq \lceil \sqrt{q} \rceil$ such that $\log_\alpha(x) = u + \lceil \sqrt{q} \rceil v$ we enumerate the sets $S_1 = \{\alpha^u (\bmod p) \,|\, 0 \leq u \leq \lceil \sqrt{q} \rceil\}$ and $S_2 = \left\{x\alpha^{-\lceil \sqrt{q} \rceil v} \,(\bmod p) \,|\, 0 \leq v \leq \lceil \sqrt{q} \rceil\right\}$ and we search for a common element $\alpha^u = x\alpha^{-\lceil \sqrt{q} \rceil v} \,(\bmod p)$. The generation of S_1 and S_2 takes 2^{71} modular multiplications. Sorting and merging S_1 and S_2 requires more than 2^{72} steps.

More general groups. It is possible to implement the above signature and authentication scheme using a finite group G other than the subgroup $\mathbb{Z}_p^*$ of units in $\mathbb{Z}_p$. We can use any finite group with an efficient multiplication algorithm and having the property that the discrete logarithm is infeasible to compute. In the general case we have $\alpha \in G$ and the order q of α must have some prime factor that is larger than 2^{140}. In case that the order q is publicly known the modification of our basic scheme is straightforward. If we are only given an upper bound M for q then we choose in the preprocessing phase a random number r in the interval $\{1, \ldots, M\}$ and we dispense with the reduction modulo q in the protocols for authentication and signature generation. Examples of suitable groups are e.g. class groups and elliptic curves $E(K)$ over a finite field K.

The choice of the hash function h. We distinguish two types of attacks:

a) Given a message m find a signature for m.

b) *Chosen message attack.* Sign an unsigned message m of your choice.

We call a function h *one–way* if for all but a negligible fraction of the output values it is infeasible to invert h. The function h is called *collision–free* if it is infeasible to generate two inputs with matching outputs.

Attack a) requires to solve the multivariate congruence

$$h(\alpha^y v^e \,(\bmod p),\, m) = e$$

in y and e. No method has been found to solve such a congruence since this type of congruence came up in connection with the ElGamal scheme. In order to thwart the attack a) the function $h(x, m)$ must be almost *uniform* with respect to x in the following sense. For every message m, every $e \in \{0, \ldots, 2^t - 1\}$ and random $x \in \mathbb{Z}_p^*$ the probability $prob_x[h(x, m) = e]$ must be close to 2^{-t}. Otherwise, in case that for fixed m, e the event $h(x, m) = e$ has nonnegligible probability with respect to random number x, the cryptanalyst can compute $\overline{x} := \alpha^y v^e \pmod{p}$ for random y–values until the equality $e = h(\overline{x}, m)$ holds. The equality yields a signature (y, e) for message m. If $h(x, m)$ is uniformly distributed with respect to random x then this attack requires about 2^t steps.

Attack b) can be launched if we are given many pairs (y_i, e_i) so that the functions $h(x_i, \cdot)$ with $x_i = \alpha^{y_i} v^{e_i} (\mathrm{mod} p)$ all coincide. Given (y_i, e_i) for $i = 1, \ldots, 2^{t/2}$ the cryptanalyst can generate messages m_j for $j = 1, \ldots, 2^{t/2}$ and check whether there exist i, j such that $h(x_i, m_j) = e_i$. In this case he has found a signature (y_i, e_i) for message m_j. The probability of success is about $2^{t/2} 2^{t/2} 2^{-t} = 1$. Given the pairs (y_i, e_i) the work load for the attack is about $2^{t/2} \log{(t/2)}$ steps. For this the cryptanalyst sorts the sets $S_1 = \{e_i \text{ for } i = 1, \ldots, 2^{t/2}\}$ and $S_2 = \{h(x_i, m_j) \text{ for } j = 1, \ldots, 2^{t/2}\}$ and searches for a joint element by merging the sets S_1 and S_2. It is important that by assumption $h(x_i, m_j)$ does not depend on i. In order to thwart this attack the function $h(x, m)$ must depend on at least 140 bits of the number x.

In order to thwart the chosen message attack the function $h(x, m)$ must, for all but a negligible fraction of x, be *one–way* in the argument m. Otherwise the cryptanalyst can choose y, e arbitrarily, he computes $\overline{x} := \alpha^y v^e \pmod{p}$ and solves $e = h(\overline{x}, m)$ for m. This yields a signature for message m.

It seems not necessary that the function $h(x, m)$ is collision–free with respect to m. Suppose the cryptanalyst finds messages m and m' such that $h(x, m) = h(x, m')$ for some $x = \alpha^y \pmod{p}$. If he asks for a signature for m' then this signature is based on an arbitrary random number x' and cannot simply be used to sign m. The equality $h(x,m) = h(x,m')$ only helps to sign m if a signature (y,e) for m' is given using this particular x, i.e. $x = \alpha^y v^e \pmod p$. But if $h(x,m)$ is one–way in m then it is difficult to solve $h(x,m) = h(x,m')$ for given x, m'.

By the same reason the analyst cannot simply attack using a weak x, where $h(x,m)$ is not one–way in m. For this attack he needs to know a valid signature with x. Since the signature protocol generates x as the result of a one–way function it seems to be sufficient that the fraction of weak x is negligible. Even though a negligible fraction of weak x does not seem to hurt the scheme we strongly recommend to use a hash–function $h(x,m)$ that is one–way in m for each fixed x.

Comparison with ElGamal signatures. An ElGamal signature (y,x) for the message m and keys v,s with $v = \alpha^{-s} \pmod p$ satisfies the equation $\alpha^m = v^x x^y \pmod p$ and can be generated from a random number r by setting $x := \alpha^r \pmod p$ and by computing y from the equation

$$ry - sx = m \pmod{p-1}\ . \tag{1}$$

We replace in equation (1) x by the hash value $e = h(x,m)$. Then we can eliminate the right side m in equation (1). We further simplify (1) through replacing the product ry by $y - r$ and $p-1$ by q. This transforms (1) into the new equation $y = r + es \pmod q$. The new signatures are much shorter.

Relationship to the Beth authentication scheme. Beth (1988) proposes an authentication scheme in which the user's private key y is part of the KAC's ElGamal–signature (x,y) for the user's identification number I. The KAC produces the signature (x,y) when it registers a legitimate user. Let $\overline{s}, \overline{v}$ be the KAC's private and public ElGamal keys. We have $\overline{v} = \alpha^{-\overline{s}} \pmod p$ and $\alpha^I = \overline{v}^x x^y \pmod p$. Now x and y are taken for the user's public and private keys. In order to authenticate himself to a third party it is sufficient that the user proves knowledge of y. Knowledge of y means knowledge of the KAC's signature for the identification number I. Only the KAC can produce this signature. According to the ElGamal protocol, y is a well defined discrete logarithm, $y = \log_x(\alpha^I \overline{v}^{-x})$. In the

Beth scheme a user proves knowledge of y using the parallel variant of protocol 1 in Chaum, Evertse, Graaf (1988).

The construction of Beth saves a separate signature by the KAC for the user's public key. It also saves a separate transmission of this signature in the authentication protocol as well as its separate verification. The penalty for this two fold. The user must reveal its secret key to the KAC. The authentication test is less efficient, the verifier has to perform three exponentiations. Beth does not consider signatures related to his authentication scheme. He works with $GF(q)$, in particular with $GF(2^n)$ instead with $\mathbb{Z}_p$.

Girault (1990) has proposed a variant of our scheme that is identity based and which does not reveal the user's secret key to the KAC.

3 The Performance of the Signature Scheme.

We wish to achieve a security level of 2^{72} operations, i.e. the best known method for forging a signature/authentication should require at least 2^{72} steps. In order to obtain the security level 2^{72} we choose $q \geq 2^{140}$, $t = 72$ and $p \geq 2^{512}$. The number of multiplication steps and the length of the message dependent part of signatures are independent of the bit length of p. Only the length of the public key depends on p. We compare the performance of the new scheme to the Fiat–Shamir scheme ($k = 9$, $t = 8$) the RSA–scheme and the GQ–scheme of Guillou and Quisquater.

# of multiplications	new scheme $t = 72$	Fiat–Shamir $k = 9,\ t = 8$	RSA	GQ
signature generation (without preprocessing)	0	44^*	$750^{*\#}$	180^*
preprocessing	210^{**}	0	0	0
signature verification$^+$	228^*	44^*	> 2	108^*

*) Can be reduced by optimization. Standard optimizations either use exponents with small Hamming weight or use short addition chains.

**) Can be greatly reduced using the preprocessing algorithm of section 4 provided that this algorithm is secure.

#) Computing modulo each prime factor of the RSA–modulus reduces these modular multiplications to multiplications with twice shorter numbers.

+) This does not include the verification of the pair (I, v) consisting of the user's public key v and identification string I.

Fast algorithms for signature verification exist for the RSA–scheme with small exponent and for the Micali–Shamir variant of the Fiat–Shamir scheme. The new scheme is most efficient for signature generation. Recently Ong and Schnorr (1990) have proposed another variant of the Fiat–Shamir scheme. For this variant signatures can be generated using about 13 modular multiplications.

# bytes for the new scheme	
p	64 (32, resp. see below)
q	17.5
public key v	64
private key s	17.5

Given the prime q we can choose the prime p so that

$$2^{255} < 2^{512} - p < 2^{256} \ .$$

The particular form of p simplifies the arithmetic modulo p and allows to store p with only 32 bytes. The particular form of p does provide no advantage in any of the known discrete logarithm algorithms. This also holds for the number field sieve algorithm by A.K. Lenstra, H.W. Lenstra, Manasse and Pollard (1990), see Gordon (1990) and for the cubic sieve algorithm by Coppersmith, Odlyzko and Schroeppel (1986).

 # bytes for complete signatures. If the KAC also uses the new signature scheme then its signature S for (I, v) is also of the form (e, y) and is 26.5 bytes long. Then a complete signature consisting of I, v, S, e, y is only about 127 bytes long:

identification string I	10	bytes
public key v	64	bytes
the KAC's signature S	26.5	bytes
message dependent part of signature (e, y)	26.5	bytes
	127	bytes

Signatures for the new scheme are much shorter than for other schemes. Fiat–Shamir signatures with $k = 9,\ t = 8$ are 531 bytes long. A signature consists of I (10 bytes) $e_{1,1}, \ldots, e_{9,8} \in \{0, 1\}$ (9 bytes) and $y_1, \ldots, y_8 \in \mathbb{Z}_N$ ($8 \cdot 64 = 512$ bytes). Signatures in the RSA scheme are 202 bytes long. A signature consists of I (10 bytes) the users modulus (64 bytes) the KAC's signature of the users modulus (64 bytes) and the message dependent part of the signature (64 bytes).

The number of communication bytes for authentication.
We consider the parameters $k = 5,\ t = 4$ with security level 2^{-20}.

I	10		bytes
v	64		bytes
S	26.5		bytes
x(h(x))	64	(9)	bytes
e	2.5		bytes
y	17.5		bytes
	84.5	(129.5)	bytes

The amount of communication for the new scheme is much less than for the Fiat–Shamir authentication scheme. We compare to the Fiat–Shamir scheme with $k = 5,\ t = 4$ and security level 2^{-20}. The Fiat–Shamir authentication requires to exchange 524.5 (304.5) bytes of information. This information consists of I (10 bytes) $y_1, \ldots, y_4 \in \mathbb{Z}_N$ (256 bytes),

$e_{1,1}, \ldots, e_{4,5} \in \{0,1\}$ (2.5 bytes) $x_1, \ldots, x_4 \in \mathbb{Z}_N$ ($4 \cdot 64 = 256$ bytes). Using hash values $h(x_1), \ldots, h(x_4)$ the latter part of the communication reduces to $4 \cdot 9 = 36$ bytes.

4 Preprocessing the random number exponentiation.

The preprocessing for authentication/signature generation consists of an exponentiation $r \mapsto \alpha^r (\mathrm{mod} p)$ of a random number $r \in \{1, \ldots, q\}$. If q is 140 bits long this exponentiation can be done using 210 multiplications modulo p. The exponentiation of random numbers constitutes the core of other crypto schemes as well, as e.g. the schemes of El–Gamal (1985), Beth (1988) and G"unther (1989). In this section we propose a very efficient algorithm that simulates the exponentiation of a random number modulo p. If proven to be secure this algorithm can be used in the preprocessing phase of our scheme and in the other crypto schemes as well.

The smart card stores a collection of k independent random pairs (r_i, x_i) for $i = 1, \ldots, k$ such that $x_i = \alpha^{r_i} \pmod p$ where the numbers r_i are independent random numbers in $\{1, \ldots, q\}$. Initially these pairs can be generated by the KAC. For every signature/authentication the card uses a random combination (r, x) of these pairs and subsequently rejuvenates the collection of pairs by combining randomly selected pairs. We use a random combination (r, x) in order to release minimum information on the pairs (r_i, x_i) $i = 1, \ldots, k$. For each signature generation we randomize the pairs (r_i, x_i) so that no useful information can be collected on the long run.

It is not necessary to publish the preprocessing algorithm. Each smart card can have its own secret algorithm for preprocessing. Even though the preprocessing algorithm may be private it is important to know whether a cryptographically secure preprocessing algorithm exists. For this we propose a specific *example algorithm* and give some evidence that it is secure even if the algorithm is public. The algorithm performs an *internal randomization* using a random permutation of the numbers $1, \ldots, k$. After a few rounds of preprocessing the new pairs $(r_1, x_1), \ldots, (r_k, x_k)$ will be quasi–independent from the present pairs.

Preprocessing algorithm.

Initiation. Load r_i, x_i for $i = 1, \dots, k$, $\nu := 1$ (ν is the *round number*).

1. Pick a random permutation a of $\{1, \dots, k\}$.
2. $r := r_\nu + 2r_{\nu-1} \pmod q$, $x := x_\nu x_{\nu-1}^2 \pmod p$, $u := r$, $z := x$ (here $\nu - 1 \in \{1, \dots, k\}$ is the residue of $\nu - 1 (\bmod k)$).
   Keep r, x for the next signature.
3. FOR $i = k, \dots, 1$ DO $[u := r_{a(i)} + 2u \pmod q$, $z := x_{a(i)} z^2 \pmod p]$
4. $r_\nu := u$, $x_\nu := z$, $\nu := \nu + 1 \pmod k$, go to 1 for the next round.

To simplify subsequent discussions we denote $a(k+1) = \nu$ and $a(k+2) = \nu - 1 \pmod k$. Then one round of preprocessing performs

$$r_\nu := \sum_{i=1}^{k+2} r_{a(i)} 2^{i-1} \pmod q, \qquad x_\nu := \prod_{i=1}^{k+2} x_{a(i)}^{2^{i-1}} \pmod p\ .$$

REMARKS.

1) One round of preprocessing takes only $2k+2$ multiplications modulo p, $k+1$ additions modulo q and $k+1$ shifts.
2) In practical applications the numbers $a(1), \dots, a(k)$ are generated by a pseudo–random number generator. This does not weaken the cryptographic security provided that the random generator is perfect.
3) It has been shown in Schnorr (1990) that if the initial numbers $(r_1, \dots, r_k)$ are uniformly distributed over $\{1, \dots, q\}$ then the uniform distribution of $(r_1, \dots, r_k)$ is preserved throughout the preprocessing and that any k consecutive r–values, to be used for k consecutive signatures, are also uniformly distributed.
4) The above preprocessing algorithm has been proposed in Schnorr (1990) with $k = 8$ and with arbitrary numbers $a(1), \dots, a(d) \in \{1, \dots, k\}$ and $d \leq k$. De Rooij (1991) has pointed out that this preprocessing is vulnerable if it is possible to choose in round ν all numbers $a(1), \dots, a(k)$ to be either ν or $(\nu - 1) \bmod k$. His attack is

thwarted by the requirement that the numbers $a(1), \dots, a(k)$ form a permutation of $1, \dots, k$.

5) No attack is known that constructs the secret key s from signatures generated with the above preprocessing algorithm and which uses less than 2^{72} steps in case that $k = 8$.

Acknowledgement. I wish to thank J. Hastad, S. Micali and J. Pollard for their comments and the unknown referees for their suggestions.

References.

Beth, T.: Efficient Zero–Knowledge Identification Scheme for Smart Cards. Advances in Cryptology - EUROCRYPT'88, Lecture Notes in Computer Science **330** (1988), pp. 77–86.

Brickell, E.F. and McCurley, K.S.: An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. Advances in Cryptology – EUROCRYPT'90, Lecture Notes in Computer Science **473** (1991), pp. 63 – 71.

Chaum, D., Evertse, J.H. and van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and some Generalizations. Advances in Cryptology - EUROCRYPT'87, Lecture Notes in Computer Science **304** (1988), pp. 127–141.

Coppersmith, D., Odlyzko, A. and Schroeppel, R.: Discrete Logarithms in $GF(p)$. Algorithmica **1** (1986), pp. 1–15.

ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory **31** (1985), pp. 469–472.

Feige, U., Fiat, A. and Shamir, A.: Zero–Knowledge Proofs of Identity. Proceedings of STOC 1987, pp. 210–217, and J. Cryptology **1** (1988), pp. 77–95.

Fiat, A. and Shamir, A.: How to Prove Yourself: Practical Solutions of Identification and Signature Problems. Advances in Cryptology - CRYPTO'86, Lecture Notes in Computer Science **263** (1987), pp. 186–194.

Girault, M. : An Identity–Based Identification Scheme Based on Discrete Logarithms. Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science **473** (1991), pp. 481 – 486.

Goldwasser, S., Micali, S. and Rackoff, C.: Knowledge Complexity of Interactive Proof Systems. Proceedings of STOC 1985, pp. 291–304.

Gordon, D. : Discrete logarithms in $GF(p)$ using the number field sieve. TR. Sandia Laboratories (1990).

Guillou, L.S. and Quisquater, J.J.: A Practical Zero–Knowledge Protocol fitted to Security Microprocessor minimizing both Transmission and Memory. Advances in Cryptology - EUROCRYPT'88, Lecture Notes in Computer Sciences, **330** (1988), pp. 123–128.

Günther, C.G.: An Identity–Based Key–Exchange Protocol. Advances in Cryptology - EUROCRYPT'89, Lecture Notes in Computer Science **434** (1990), pp. 29–37.

Lenstra, A.K., Lenstra, H.W.,Jr., Manasse, M.S. and POLLARD, J.M.: The Number Field Sieve. Proceedings of STOC 1990.

Ong, H. and Schnorr, C.P.: Fast Signature Generation with a Fiat Shamir–like Scheme. Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science, **473** (1991), pp. 432 – 440.

Rabin, M.O.: Digital Signatures and Public–Key Functions as intractable as Factorization. Technical Report MIT/LCS/TR–212 (1978).

Rivest, R., Shamir, A. and Adleman, L.: A Method for Obtaining digital Signatures and Public Key Cryptosystems. Comm. ACM **21** (1978), pp.120–126.

de Rooij, P.J.N. : On the Security of the Schnorr Scheme using Preprocessing. Proceedings EUROCRYPT'91.

Schnorr, C.P. : Efficient Identification and Signatures for Smart Cards. Advances in Cryptology - CRYPTO'89. Lecture Notes in Computer Science **435**, (1990), pp. 239 – 252.

References.

Beth, T.: Efficient Zero–Knowledge Identification Scheme for Smart Cards. Advances in Cryptology - EUROCRYPT'88, Lecture Notes in Computer Science **330** (1988), pp. 77–86.

Brickell, E.F. and McCurley, K.S.: An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. Advances in Cryptology – EUROCRYPT'90, Lecture Notes in Computer Science **473** (1991), pp. 63 – 71.

Chaum, D., Evertse, J.H. and van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and some Generalizations. Advances in Cryptology - EUROCRYPT'87, Lecture Notes in Computer Science **304** (1988), pp. 127–141.

Coppersmith, D., Odlyzko, A. and Schroeppel, R.: Discrete Logarithms in $GF(p)$. Algorithmica **1** (1986), pp. 1–15.

ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory **31** (1985), pp. 469–472.

Feige, U., Fiat, A. and Shamir, A.: Zero–Knowledge Proofs of Identity. Proceedings of STOC 1987, pp. 210–217, and J. Cryptology **1** (1988), pp. 77–95.

Fiat, A. and Shamir, A.: How to Prove Yourself: Practical Solutions of Identification and Signature Problems. Advances in Cryptology - CRYPTO'86, Lecture Notes in Computer Science **263** (1987), pp. 186–194.

Girault, M. : An Identity–Based Identification Scheme Based on Discrete Logarithms. Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science **473** (1991), pp. 481 – 486.

Goldwasser, S., Micali, S. and Rackoff, C.: Knowledge Complexity of Interactive Proof Systems. Proceedings of STOC 1985, pp. 291–304.

Gordon, D. : Discrete logarithms in $GF(p)$ using the number field sieve. TR. Sandia Laboratories (1990).

Guillou, L.S. and Quisquater, J.J.: A Practical Zero–Knowledge Protocol fitted to Security Microprocessor minimizing both Transmission and Memory. Advances in Cryptology - EUROCRYPT'88, Lecture Notes in Computer Sciences, **330** (1988), pp. 123–128.

Günther, C.G.: An Identity–Based Key–Exchange Protocol. Advances in Cryptology - EUROCRYPT'89, Lecture Notes in Computer Science **434** (1990), pp. 29–37.

Lenstra, A.K., Lenstra, H.W.,Jr., Manasse, M.S. and POLLARD, J.M.: The Number Field Sieve. Proceedings of STOC 1990.

Ong, H. and Schnorr, C.P.: Fast Signature Generation with a Fiat Shamir–like Scheme. Advances in Cryptology - EUROCRYPT'90, Lecture Notes in Computer Science, **473** (1991), pp. 432 – 440.

Rabin, M.O.: Digital Signatures and Public–Key Functions as intractable as Factorization. Technical Report MIT/LCS/TR–212 (1978).

Rivest, R., Shamir, A. and Adleman, L.: A Method for Obtaining digital Signatures and Public Key Cryptosystems. Comm. ACM **21** (1978), pp.120–126.

de Rooij, P.J.N. : On the Security of the Schnorr Scheme using Preprocessing. Proceedings EUROCRYPT'91.

Schnorr, C.P. : Efficient Identification and Signatures for Smart Cards. Advances in Cryptology - CRYPTO'89. Lecture Notes in Computer Science **435**, (1990), pp. 239 – 252.

$$
\begin{array}{ccc}
 & \alpha, q, p, h & \\
 & \swarrow \qquad\qquad \searrow & \\
I, s, v, (S) & \xrightarrow{I, v, (S)} & \text{check } \ I, v, (S) \\
\text{pick random } \ r & & \\
x := \alpha^r \pmod{p} & \xrightarrow{h(x)} & \\
 & \xleftarrow{e} & \text{pick random } \ e \\
y := r + se \pmod{q} & \xrightarrow{y} & \overline{x} := \alpha^y v^e \pmod{p} \\
 & & \text{check that } h(x) = h(\overline{x}) \\
 & & \\
\text{PROVER} & & \text{VERIFIER}
\end{array}
$$

Figure 1: AUTHENTICATION

$$
\begin{array}{ccc}
 & \alpha, q, p, h & \\
 & \text{message } \ m & \\
 & \swarrow \qquad\qquad \searrow & \\
I, s, v, (S) & \xrightarrow{I, v, (S)} & \text{check } \ I, v, (S) \\
\text{pick random } \ r & & \\
x := \alpha^r \pmod{p} & & \\
e := h(x, m) & & \\
y := r + se \pmod{q} & \xrightarrow{e, y} & \overline{x} := \alpha^y v^e \pmod{p} \\
 & & \text{check that } \ e = h(\overline{x}, m) \\
 & & \\
\text{SIGNATURE GENERATION} & & \text{SIGNATURE VERIFICATION}
\end{array}
$$

Figure 2: SIGNATURE GENERATION AND VERIFICATION