EFFICIENT IDENTIFICATION AND SIGNATURES FOR SMART CARDS [§]

C.P. Schnorr
Universität Frankfurt

1. Introduction

We present an efficient interactive identification scheme and a related signature scheme that are based on discrete logarithms and which are particularly suited for smart cards. Previous cryptoschemes, based on the discrete logarithm, have been proposed by El Gamal (1985), Chaum, Evertse, Graaf (1988), Beth (1988) and Günter (1989). The new scheme comprises the following novel features.

(1) We propose an efficient algorithm to preprocess the exponentiation of random numbers. This preprocessing makes signature generation very fast. It also improves the efficiency of the other discrete log-cryptosystems. The preprocessing algorithm is based on two fundamental principles *local randomization* and *internal randomization*.
(2) We use a prime modulus p such that p-1 has a prime factor q of appropriate 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, i.e. it is less than half the length of RSA and Fiat-Shamir signatures. The number of communication bits of the identification scheme is less than half that of other schemes.

The new scheme minimizes the work to be done by the smart card for generating a signature or for proving its identity. This is important since the power of current processors for smart cards is rather limited. Previous signature schemes require many modular multiplications for signature generation. In the new scheme signature generation costs about 12 modular multiplications, and these multiplications do not depend on the message/identification, i.e. they can be done in preprocessing mode during the idle time of the processor.

The security of the scheme relies on the one-way property of the exponentiation $y \mapsto \alpha^y \pmod p$, i.e. we assume that discrete logarithms with

§ european patent application 89103290.6 from 24.2.1989 base α are difficult to compute. The security of the preprocessing is established by information theoretic arguments.

This abstract is organised as follows. We present in section 2 a version of the signature scheme that uses exponentiation of a random integer. In section 3 we propose an efficient algorithm that simulates this exponentiation. We study its security in section 4. The performance of the scheme is exemplified in section 5.

2. The identification 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 $\{1,...,n\}$.

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 $h : \mathbb{Z}_q \times \mathbb{Z} \to \{0,...,2^t-1\}$,
- its own private and public key.

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

COMMENTS. The KAC's own keys are used for signing the public keys issued by the KAC. The KAC can use for its own signatures any public key signature scheme, e.g. RSA, Fiat-Shamir, Rabin or the new scheme presented here. The hash function h is only used for signatures and is not needed for identification. The function h outputs random numbers in $\{0,...,2^t-1\}$; for the choice of the function h see the end of section 2. The *security number* t can depend on the application intended, we consider $t = 72$. The scheme is designed such that forging a signature or an identification requires, with $t = 72$, about 2^{72} steps.

Registration of users. When a user comes to the KAC for registration the KAC verifies its identity, generates an identification string I (containing name, address, ID-number etc.) and signs the pair (I,v) consisting of I and the user's public key v. The user can generate himself his private key s and the corresponding public key v.

The user's private and public key. Every user has a private key s which is a random number in $\{1,2,...,q\}$. 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 protocol is related to protocol 1 in Chaum, Evertse, Graaf (1988); it condenses this protocol to a single round.

The identification protocol

(Prover A proves its identity to verifier B)

1. *Initiation.* A sends to B its identification string I and its public key v. B checks v by verifying KAC's signature transmitted by A.
2. *Preprocessing.* A picks a random number $r \in \{1,...,q-1\}$, computes $x := \alpha^r$ (mod p), and sends x to B (see section 3 for an efficient simulation of this exponentiation).
3. B sends a random number $e \in \{0,...,2^t-1\}$ to A.
4. A sends to B $y := r + se$ (mod q) .
5. *Identification test.* B checks that $x = \alpha^y v^e$ (mod p) and accepts A's proof of identity iff equality holds.

Obviously if A and 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 identification. The proof (x,y) (the exam e, resp.) is called *straight* if A (B, resp.) has followed the protocol, otherwise the proof (exam, resp.) is called *crooked.*

A fraudulent A can cheat by guessing the correct e and sending the crooked proof

$$x := \alpha^r v^e \text{ (mod p)}, \quad y := r .$$

The probability of success for this attack is 2^{-t}. By the following proposition this success rate cannot be increased unless computing $\log_\alpha v$ is easy.

Proposition 2.1 *Suppose there is a probabilistic algorithm AL with time bound $|AL|$ which takes for input a public key* v *and withstands, with probability* $\varepsilon > 2^{-t+2}$, *the identification test for a straight exam. Then the discrete logarithm of* v *can be computed in time* $O(|AL|/\varepsilon)$ *and constant, positive probability.*

Proof. This is similar to Theorem 5 in Feige, Fiat, Shamir (1987). The following algorithm AL' computes $\log_\alpha v$.

1. Repeat the following steps at most $1/\varepsilon$ times: generate x the same way as does algorithm AL, pick a random e' in $\{0,...,2^t-1\}$ and check whether AL passes the identification test for (x,e'); if AL succeeds then fix x and go to 2.
2. Probe $1/\varepsilon$ random numbers e" in $\{0,...,2^t-1\}$. If algorithm AL passes the identification test for some e" that is distinct from e' then go to 3 and otherwise stop.
3. Choose the numbers y', y" which AL submits to the identification test in response to e', e". (y'-y" is the discrete logarithm of $v^{e''-e'}$ (mod p).)
4. Output (y'-y")/(e"-e') (mod q) .

We bound from below the success probability of this algorithm. The algorithm finds in step 1 a passing pair (x,e') with probability at least $\frac{1}{2}$. With probability at least $\frac{1}{2}$, the x chosen in step 1, has the property that AL withstands the identification test for at least a $\frac{1}{2}$ ε-fraction of all $e \in \{0,\dots,2^t-1\}$. For such an x step 2 finds a passing number e" that is distinct from e' with probability at least

$$1 - (1-\varepsilon/2)^{1/\varepsilon} > 1 - 2.7^{-1/2} > 0.3 .$$

This shows that the success probability of the algorithm is at least 0.3/4.

□

The verifier B is free to choose the bit string e in step 3 of the identification protocol, thus he can try to choose e in order to obtain useful information from A. 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. Furthermore it is unlikely that the number y reveals any useful information because y is superposed by the discrete logarithm of x, $y = \log_\alpha x + e \cdot s \pmod q$, and the cryptanalyst cannot infer $r = \log_\alpha x$ from x. The scheme is not zero-knowledge because the tripel (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. We can reduce the number of communication bits for identification. For this A sends in step 2 h(x) (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. We can take for h(x) the t least significant bits of x. The total number of communication bits for h(x),e,y is 2t + 140 which is less than half that of other schemes. The transmission of e is not necessary, e can be fixed to h(x). Then 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 using 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 and public key v compute $\overline{x} = \alpha^y v^e \pmod p$ and check that $e = h(\overline{x},m)$ *(signature test).*

A signature (e,y) is considered to be *valid* if it withstands the signature test. A signature generated according to the protocol is always valid since

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

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(mod q) where the numbers s and e are about 140 and t = 72 bits long. The latter multiplication is negligible compared with a modular multiplication in the RSA-scheme.

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

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

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

```
1. $i := l$ , $z := 1$ ,
2. while $i \geq 0$ do [$i := i-1$, $z := z^2 \alpha^{y_i} v^{e_i} \pmod p$] ,
3. $\bar{x} := z$ .
```

This computation requires at most $l + t - 1 + \sum_{i=t}^{l} y_i$ 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 $1.5\, l + 0.25\, t$ modular multiplications.

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

$$ry - sx = m \pmod{p-1} \qquad (1)$$

We replace in equation (1) x by the hash value $e = h(x,m)$. Then we can dispense with the right side m in equation (1) which we make zero. We further simplify (1) in that we replace 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.

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 $S_1 = \{\alpha^u \pmod p \mid 0 \leq u \leq \lceil \sqrt{q} \rceil\}$ and $S_2 = \{x\, \alpha^{-\lceil \sqrt{q} \rceil v} \pmod p \mid 0 \leq v \leq \lceil \sqrt{q} \rceil\}$ and we search for a common element $\alpha^u = x\alpha^{-\lceil \sqrt{q} \rceil v} \pmod p$.

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 by using signatures of messages of your choice.

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 $\text{prob}_x[h(x,m) = e]$ must be near to 2^{-t}. Otherwise, in case that for fixed m,e the event $h(x,m) = e$ has nonnegligible probability with respect to random x, the cryptanalyst can compute $\overline{x} := \alpha^y r^e \pmod p$ for arbitrary 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.

In order to thwart the chosen message attack the function $h(x,m)$ must 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. Then he has found a signature for an arbitrary message m.

It is 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 a new 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 such that $x = \alpha^y v^e \pmod p$. 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'.

3. Preprocessing the random number exponentiation

We describe an efficient method for preprocessing the random numbers r and $x := \alpha^r \pmod p$, that are used for signature generation. This preprocessing mode also applies to other discrete log-cryptosystems such as the schemes by ElGamal (1985), Beth (1988) and Günter (1989).

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/identification 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. We give an *example* of a preprocessing algorithm that demonstrates the method. It uses a *security parameter* d, for all practical purposes d and k can be fairly small integers, for this paper we assume that $6 \le d,k$.

Preprocessing algorithm

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

```
1. Pick random numbers $a(0),...,a(d-3) \in \{1,...,k\}$, $a(d-2) := a(d) := \nu-1 \pmod k$,
   $a(d-1) := \nu$.
2.       $r_\nu := \sum_{i=0}^{d} r_{a(i)} 2^i \pmod q$ ,  $x_\nu := \prod_{i=0}^{d} x_{a(i)}^{2^i} \pmod p$ ,
   (Below we give a detailed algorithm for this computation.)
3. Keep for the next signature/identification the pair r, x with
         $r := r_\nu^{old} + 2 \cdot r_{\nu-1} \pmod q$,  $x := x_\nu^{old} \cdot x_{\nu-1}^2 \pmod p$.
4. $\nu := \nu+1 \pmod k$ , go to 1 for the next round.
```

REMARKS. 1. By the choice of a(d-1) the preprocessing preserves the uniform distribution on $(r_1,...,r_k)$.
2. The setting $a(d) := \nu-1 \pmod k$ has the effect that step 2 shifts the binary representation of $r_{\nu-1}$ for d positions to the left and subsequently adds it to r_ν. Theorem 4.2 relies on the choice of a(d-1). Lemma 4.3 relies on the choice a(d), and Theorem 4.4 relies on the choice of a(d-2), a(d-1) and a(d).
3. The preprocessing algorithm must not be public. Each smart card can have its own secret algorithm for preprocessing. There are many variations of the above technique. It is possible to take for $(r_{a(i)}, x_{a(i)})$ with $0 \le i < d-2$ the key pair (-s,v).

We describe step 2 of the preprocessing algorithm in detail. Step 2 can be done using only 2d multiplications modulo p, d additions modulo q and d shifts.

Step 2 of the preprocessing algorithm.

```
1. $u := r_{a(d)}$ ,  $z := x_{a(d)}$ ,  $i := d-1$ .
2. while $i \ge 0$ do [$u := 2u + r_{a(i)} \pmod q$ ,  $z := z^2 x_{a(i)} \pmod p$ ,
                     if $i = d-1$ then ($r := u, x := z$) ,  $i := i-1$] .
3. $r_\nu := u$ ,  $x_\nu := z$ .
```

4. Cryptanalysis of preprocessing

The preprocessing algorithm combines two fundamental principles *local randomization* and *internal randomization*. The pairs (r,x) that are used for signatures are locally random in the sense that every k consecutive pairs are independent, see Theorem 4.2. The random indices a(0),...,a(d-3) perform an internal randomization. The principles of local and of internal randomization are complementary and can also be used for the construction of pseudo-random number generators and hash functions.

Notations. We denote the number a(i) of round ν as $a(i,\nu)$. Let T_ν be the k×k integer matrix that describes the transformation of the numbers $r_1,\dots,r_k$ in round ν of the preprocessing algorithm, i.e. step 2 of round ν performs $r^T := T_\nu\, r^T \pmod q$ where $r = (r_1,\dots,r_k)$. For $j \geq 0$ let r_j^* be the number r after j rounds. The sequence of r-values that is used for signatures is $r_1^*, r_2^*, \dots, r_j^*$.

Lemma 4.1 *If the initial vector* $(r_1,\dots,r_k)$ *is uniformly distributed over* $\{1,\dots,q\}^k$ *then this distribution is preserved throughout the preprocessing provided that* $2^d < q$ *.*

Proof. T_ν is the identity matrix except for row ν. Row ν is determined by the transformation of r_ν in step 2:

$$r_\nu := r_\nu\,(\det T_\nu) + \sum_{a(i,\nu)\neq\nu} r_{a(i,\nu)}\, 2^i \pmod q$$

where $\det T_\nu = \sum_{a(i,\nu)=\nu} 2^i$. It follows from $a(d-1,\nu) = \nu$ and $a(d,\nu) \neq \nu$ that $\det T_\nu$ is a nonzero integer and thus $1 \leq \det T_\nu < 2^d < q$. We see that T_ν is invertible modulo q. Therefore T_ν preserves the uniform distribution on $\{1,\dots,q\}^k$. □

A similar argument proves the next theorem.

Theorem 4.2 *If the initial vector* $(r_1,\dots,r_k)$ *is uniformly distributed over* $\{1,\dots,q\}^k$ *then for all* $j \geq 0$ *and for all numbers* $a(i,\nu)$, $0 \leq i \leq d-3$, $\nu \leq k+j$ *the vector* $(r_{1+j}^*,\dots,r_{k+j}^*)$ *is, for sufficiently large q, uniformly distributed over* $\{1,\dots,q\}^k$.

It is an open problem whether the vector $(r_{i_1}^*,\dots,r_{i_k}^*)$ is uniformly distributed for all indices $1 \leq i_1 < i_2 \cdots < i_k$. We believe that this holds for all but a negligible fraction of the instances for $a(i,\nu)$, $1 \leq \nu \leq i_k$.

Because of Theorem 4.2 the cryptanalyst can only attack a sequence of more than k consecutive signatures/identifications. The set of the first k+1 signatures can be attacked by guessing the numbers $a(0),\dots,a(d-3)$ of the first k rounds. Given these numbers and the first k+1 signatures the cryptanalyst can determine the secret key s and the initial numbers $r_1,\dots,r_k$ by solving a system of k+1 linear equations modulo q. This attack requires an exhaustive search over $k^{(d-2)k}$ cases.

Let r_ν^{new} be the number r_ν after ν rounds of preprocessing. If q and the numbers $a(0),\dots,a(d-3)$ for ν rounds are fixed then the number r_ν^{new} is a function of the initial numbers $r_1,\dots,r_k$ which is linear over $\mathbb{Z}_q$.

Lemma 4.3 *Pairwise distinct instances for the numbers* $a(0),\dots,a(d-3)$ *of* ν *rounds generate, for sufficiently large q, pairwise distinct linear functions* $r_\nu^{new} = r_\nu^{new}(r_1,\dots,r_k)$ *depending on the initial numbers* $r_1,\dots,r_k$ *and* q.

Proof. Let $S_\nu := T_\nu\, T_{\nu-1} \cdots T_1$ be the product matrix that describes the transformation on r for the first ν rounds of preprocessing. This is an integer matrix that does not depend on q. The dominant row (i.e. the row with the maximal entry) of S_ν is the row $\nu(\text{mod } k)$, call this row vector s_ν. We show how to decipher all numbers $a(i)$ of the first ν rounds from s_ν. To simplify the argument let $a(i,1)$ for $i=0,\dots,d$ be pairwise distinct. Then the j-largest entry of s_ν is in column $a(d-j+1,1)$ for $j=0,\dots,d$. (In general we can determine from the relative size of the largest entries of s_ν which of the numbers $a(i,1)$ coincide.) This clearly holds for $\nu = 1$ and the induction step from $\nu - 1$ to ν follows from $a(d,\nu) = \nu-1 \pmod k$. This shows how to obtain from s_ν the matrix T_1. Given the matrix T_1 we form the vector $s_\nu T_1^{-1}$ which is the dominant row of the matrix $T_\nu\, T_{\nu-1} \cdots T_2$ that corresponds to $\nu-1$ rounds starting with round number 2. Thus we can decipher in the same way the numbers $a(i,2)$ for $i=1,\dots,d$ and the matrix T_2 from $s_\nu T_1^{-1}$. Recursively we obtain from s_ν all numbers $a(i)$ of the first ν rounds. Now the claim follows from the equation

$$r_\nu^{new} = s_\nu\, r^T \pmod q$$

where $r = (r_1,\dots,r_k)$ is the initial r-vector. □

For random input $(r_1,\dots,r_k) \in (\mathbb{Z}_q)^k$ two distinct linear functions over $\mathbb{Z}_q$ give the same output with probability $1/q$. Therefore if the number of choices for $a(0),\dots,a(d-3)$ over ν rounds is about q then the number r_ν^{new} is completely randomised by the numbers $a(0),\dots,a(d-3)$ of ν rounds, and thus r_ν^{new} is quasi-independent of $r_1,\dots,r_k$.

Let $\mathbf{a}$ be the vector $\mathbf{a} = (a(i,\nu) \mid i=0,\dots,d-3,\ \nu=1,\dots,k)$. The number r_{k+1}^* is determined by $r_1,\dots,r_k$, q and $\mathbf{a}$. We know from Theorem 4.2 that the linear transformation $(r_1,\dots,r_k) \mapsto (r_1^*,\dots,r_k^*)$ is invertible modulo q. Therefore we have a function $r_{k+1}^* = r_{k+1}^*(r_1^*,\dots,r_k^*,q,\mathbf{a})$ that is linear in $r_1^*,\dots,r_k^*$ over $\mathbb{Z}_q$. By the next theorem distinct instances of $\mathbf{a}$ yield, for sufficiently large q, distinct functions r_{k+1}^* in $r_1^*,\dots,r_k^*$.

Theorem 4.4 *Pairwise distinct instances for the numbers* $a(0),\dots,a(d-3)$ *of the first k rounds generate, for sufficiently large q, pairwise distinct linear functions* r_{k+1}^* *depending on* $r_1^*,\dots,r_k^*$.

Proof. We show that distinct vectors $\mathbf{a}$ generate, for sufficiently large q, distinct linear functions $r_{k+1}^*(r_1,\dots,r_k,q,\mathbf{a})$ where the inputs are the initial numbers

$r_1,\dots,r_k$. Let s^*_{k+1} be the coefficient vector of the linear function $r^*_{k+1}(r_1,\dots,r_k,q,\mathbf{a})$, i.e. $r^*_{k+1}(r_1,\dots,r_k,q,\mathbf{a}) = s^*_{k+1}\,\mathbf{r}^T \pmod q$ with $\mathbf{r} = (r_1,\dots,r_k)$. By the method in the proof of Lemma 4.3 we can decipher from s^*_{k+1} all numbers a(i) of the first k rounds.

Now the claim follows from the choice $a(d-2,\nu) = \nu-1 \pmod k$. It follows by an argument that is similar but more involved than the one for the proof of Lemma 4.3. □

The fastest attack to the preprocessing algorithm that we are aware of enumerates the linear functions $r^*_{k+1}(r^*_1,\dots,r^*_k,q,\mathbf{a})$ that have high probability; the probability space is the set of all vectors $\mathbf{a}$. For the security level 2^{72} it is necessary that the maximal probability for these linear functions is not much larger than 2^{-72}. In order to break the preprocessing it is sufficient to guess two functions $r^*_{k+1}(r^*_1,\dots,r^*_k,q,\mathbf{a})$ and $r^*_{k+2}(r^*_2,\dots,r^*_{k+1},q,\mathbf{a})$. Given these two functions we can uncover the secret key s from the first k+2 signatures by solving a system of linear equations.

We finally consider attacks on arbitrarily many signatures from a different point of view. The problem to recover the secret key s and the initial numbers $r_1,\dots,r_k$ when the first n signatures are given, can be put into the following form.

Given integers $y_1,\dots,y_n \in \{1,\dots,q\}$ and $e_1,\dots,e_n \in \mathbb{Z}$

Find integers $s, r_1,\dots,r_k \in \{1,\dots,q\}$ such that there exist integers $t_{i,j}$, $0 \le t_{i,j} < 2^{i(d+1)}$, satisfying $y_i = e_i s + \sum_{j=1}^{k} t_{i,j}\, r_j \pmod q \quad i=1,\dots,n$. (4.1)

The searched integers $t_{i,j}$ are from the linear transformation $(r_1,\dots,r_k) \mapsto (r^*_1,\dots,r^*_n)$, hence $0 \le t_{i,j} \le 2^{i(d+1)}$. If $k^{(d-2)k} > q$ the equation (4.1) is, for almost all $y_1,e_1,\dots,y_n,e_n,s,r_1,\dots,r_k$, solvable for $t_{i,j}$ such that $0 \le t_{i,j} < 2^{i(d+1)}$. This makes this attack useless. However if k and d are small the solvability of equation (4.1) with $0 \le t_{i,j} < 2^{i(d+1)}$ may characterize the searched numbers $r_1,\dots,r_k$. It is interesting to determine the complexity of finding $r_1,\dots,r_k$ such that (4.1) is solvable with "small" integers $t_{i,j}$. It seems that this problem is more difficult than the knapsack problem since in our case all knapsack items s and $r_1,\dots,r_k$ are unknown.

Conclusion. There is a trade-off between the parameters k and d. It is sufficient to have $q \ge 2^{140}$, $k = 8$ and $d = 6$, then $k^{(d-2)k} = 2^{96}$. It is possible to further reduce k and d but we must have $k^{(d-2)k} \ge 2^{72}$.

5. 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 signatures/identification should require at least 2^{72} steps. In order to obtain the security level 2^{72} we choose $q \geq 2^{140}$ and $t = 72$. We choose for the preprocessing algorithm, the parameters $k = 8$, and $d = 6$. For the new scheme the number of multiplication steps and the length of signatures are independent of the bit length of p. Only the length of the public key depends on p. For this we assume that p is 512 bits long. We compare the performance of the new scheme to the Fiat-Shamir scheme (k=8, t=9) the RSA-scheme and the GQ-scheme of Guillou and Quisquater.

# of multiplications	new scheme t=72	Fiat-Shamir k=8 , t=9	RSA	GQ
signature generation (without preprocessing)	0	45*	750*	216*
preprocessing	12*	0	0	0
signature verification	228*	45*	≥ 2	108*

*) can be reduced by optimization

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.

# bytes for the new scheme	
system parameters p,q	82.5 (26, resp. see below)
" α	64
public key v	64
private key s	18.5
signature (e,y)	26.5
preprocessing (r_i,x_i) i=1,...,8 (6, resp.)	660 (495, resp. see below)

We can choose particular primes q and p such that

$$|q-2^{140}| \leq 2^{40} , \quad |p-2^{512}| \leq 2^{170} .$$

The particular form simplifies the arithmetic modulo q and modulo p, and requires only 26 bytes to store p and q. We are not aware of any disadvantage of this particular form for p and q. In total about 800 (635, resp.) bytes EEPROM are sufficient to store p,q,v,e,y and (r_i,x_i) for i=1,...,8 (6, resp.), α is not needed for signature generation. About 192 bytes RAM are necessary to perform modular multiplications with a 512 bit modulus p. The program for signature generation requires less than 500 bytes ROM.

Optimization. We give a variant of the preprocessing algorithm that uses only k=6 pairs (r_i,x_i) and which require on the average 12.76 modular multiplications per round. First let k=6 and let (r_7,x_7) be the pair $(-s,v)$.

Optimized preprocessing

```
1. $r := r_{\nu-1} + r_\nu \pmod q$ , $x := x_{\nu-1} \cdot x_\nu \pmod p$ ,
   keep the pair r, x for the next signature/identification,
   $u := r + r_{\nu-1} \pmod q$ , $z := x \cdot x_{\nu-1} \pmod p$
2. for $j = 1,\dots,4$ do
   [pick with probability $7\cdot3/29$, $7/29$, $1/29$ resp.
   2 , 1 , 0 resp. distinct random numbers $a \in \{1,\dots,7\}$ .
   $u := 2u + \sum_a r_a \pmod q$ , $z := z^2 \prod_a x_a \pmod p$].
3. $r_\nu := u$, $x_\nu := z$, $\nu := \nu+1 \pmod 7$, go to 1 for the next round.
```

The number of possible transformations per round is about $[7\cdot3 + 7 + 1]^4 = 29^4$. The number of possible transformations over 6 rounds is about $29^{4\cdot6} \approx 2^{116}$ which is sufficiently large to perform an internal randomization. The average number of modular multiplications is $6 + 4(2\cdot7\cdot3 + 7) / 29 \approx 12.76$.

We can further reduce either the number of pairs (r_i,x_i) or the number of modular multiplications by inserting write operations into step 2 of the preprocessing. We can at the end of the inner loop of step 2 decide, based on a coin flip, whether to replace some pair (r_a,x_a) by (u,z). This will increase the number of possible transformations per round. However this variant will only be practical if write operations are sufficiently fast.

Acknowledgement I wish to thank J. Hastad for his criticism of the previous version of the preprocessing algrithm.

References

BETH, T.: A Fiat-Shamir-like authentication protocol for the ElGamal scheme. Proceedings of Eurocrypt' 88, Lecture Notes in Computer Science 330, (1988) pp. 77-86.

CHAUM, D., EVERTSE, J.H. and van de GRAAF, J.: An Improved protocol for Demonstration Possession of Discrete Logarithms and some Generalizations. Proceedings of Eurocrypt' 87, Lecture Notes in Computer Science 304, (1988), pp. 127-141.

COPPERSMITH, D., ODLYZKO, A. and SCHROEPPEL, R.: Discrete Logarithms. 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.

FIAT, A. and SHAMIR, A.: How to Prove Yourself: Practical Solutions of Identification and Signature Problems. Proceedings of Crypto 1986, in Lecture Notes in Computer Science (Ed. A. Odlyzko), Springer Verlag, 263, (1987) pp. 186-194.

GOLDWASSER, S., MICALI, S. and RACKOFF, C.: Knowledge Complexity of Interactive Proof Systems. Proceedings of STOC 1985, pp. 291-304.

GÜNTER, C.G.: Diffie-Hellman and ElGamal protocols with one single authentication key. Abstracts of Eurocrypt' 89, Houthalen (Belgium) April 1989.

MICALI, S. and SHAMIR, A.: An Improvement of the Fiat-Shamir Identification and Signature Scheme. Crypto 1988.

QUISQUATER, J.J. and GUILLOU, L.S.: A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. Proceedings Eurocrypt' 88. Springer Verlag, Lecture Notes in Computer Sciences, vol. 330, (1988), pp. 123-128.

RABIN, M.O.: Digital signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212 (1978).

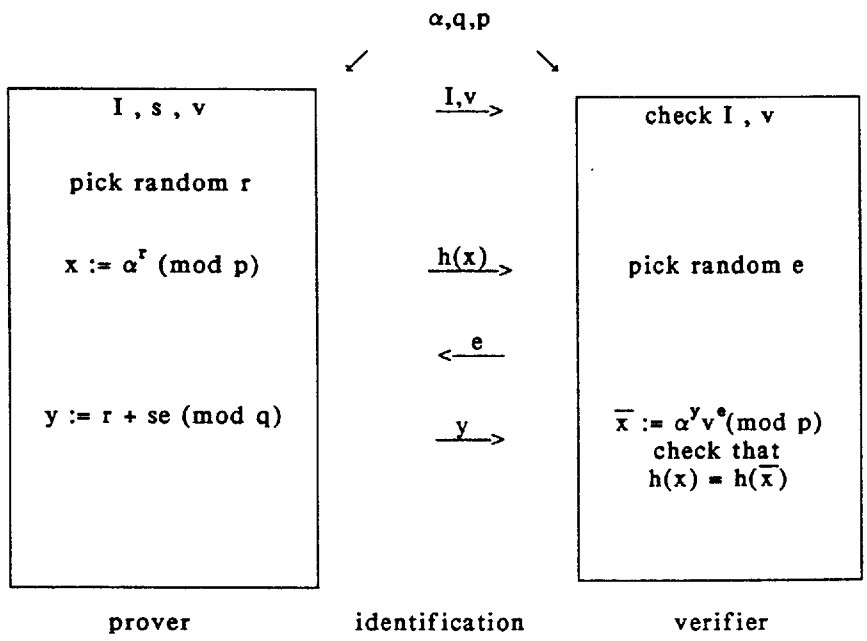
α,q,p
I,s,v
pick random r
$x := \alpha^r \pmod p$
$y := r + se \pmod q$
$\xrightarrow{I,v}$
$\xrightarrow{h(x)}$
$\xleftarrow{e}$
$\xrightarrow{y}$
check I, v
pick random e
$\overline{x} := \alpha^y v^e \pmod p$
check that
$h(x) = h(\overline{x})$
prover
identification
verifier

FIG. 1

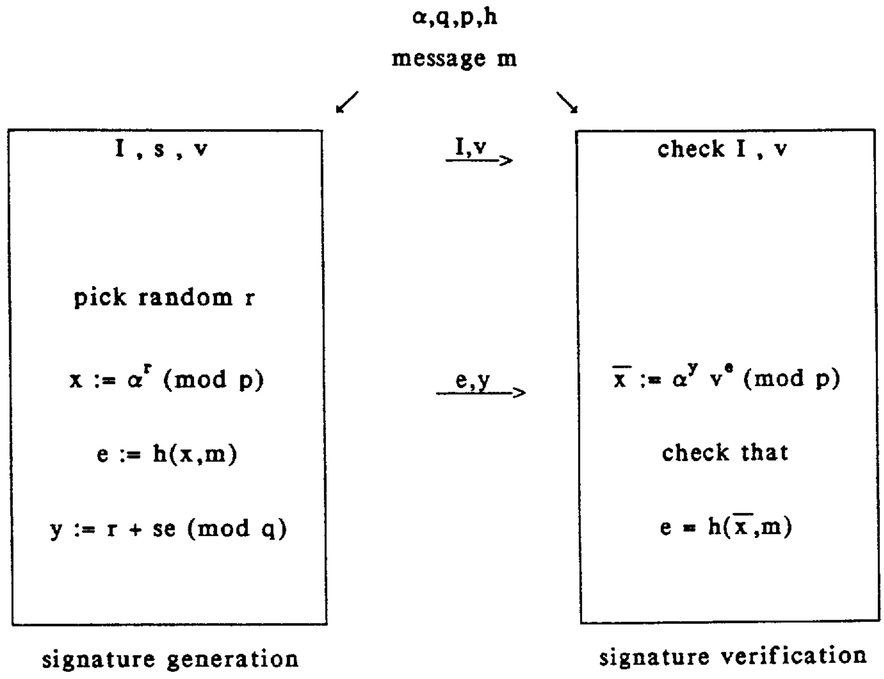
α,q,p,h
message m
I,s,v
pick random r
$x := \alpha^r \pmod p$
$e := h(x,m)$
$y := r + se \pmod q$
$\xrightarrow{I,v}$
$\xrightarrow{e,y}$
check I, v
$\overline{x} := \alpha^y v^e \pmod p$
check that
$e = h(\overline{x},m)$
signature generation
signature verification

FIG. 2