Knowledge-Proof Based Versatile Smart Card Verification Protocol

DaeHun Nyang and JooSeok Song
Department of Computer Science Department, Yonsei University
SeodaemunGu ShinchonDong 134, Seoul 120-749, Korea
{nyang, jssong}@emerald.yonsei.ac.kr

ABSTRACT

We propose a zero-knowledge interactive proof based identification and signature scheme. The protocol is based on Euler's totient function and discrete logarithms over the ring Z/nZ, and can be applied to smart cards. A prover keeps a signed subgroup generator provided by a trusted center as its secret information. Our scheme has symmetricity in the sense that the same computational complexity and the same hardware both for Prover and for Verifier are required. Also, it requires minimal amount of computation and communications for secret information. The protocol is versatile enough to be applicable to digital signature scheme, multiple digital signature scheme and key exchange protocol. We outline those protocols to show the versatility of our protocol.

Categories and Subject Descriptors

F.2.m [**Theory of Computation**]: Miscellaneous; G.2.3 [**Mathematics of Computing**]: Applications

General Terms

Security

Keywords

Cryptography, Smart card, Identification, Zero-Knowledge Interactive Proof, Digital Signature, Multiple Signature, Key Exchange

1. INTRODUCTION

In a computerized communication society, it is necessary to provide an adequate method by which communicators can identify themselves to each other in an unforgeable manner. Since 1984, considerable attention has been paid to ZKIP[1, 6, 13], which is useful in identification and digital signature.

Identification schemes using ZKIP are generally based on the difficulty of computing the discrete logarithm modulo a large prime number, quadratic residuosity of numbers, modular square roots, and the factorization of a composite number. Many identification schemes have been proposed based on zero-knowledge proof[2, 5, 10, 12].

In this paper, we propose the concept of a "signed subgroup generator" and design a new interactive proof based identification scheme and digital signature. The scheme uses discrete logarithm over the ring Z/nZ and Euler's totient function to get its security, and is suitable for smart cards because it requires only one secret number and one accreditation to complete the interactive proof. Our identification scheme is superior to Guillous and Quisquater's scheme in that a verifier's computational burden is reduced to about half and it simplifies the hardware implementation. Besides those, our protocol can be easily modified to serve as a key exchange protocol and as a multiple signature scheme.

2. IDENTIFICATION SCHEME

2.1 Background

Before proceeding, we briefly explain zero-knowledge interactive proof. Interactive proof is defined by two interactive Turing machines (P, V). They have a few tapes to do their operation and communicate with each other. Prover(P) proves a certain knowledge to verifier(V), and the proof is called a witness. ZKIP induces a formal language L, and for an input x a prover proves $x \in L$. If $x \in L$, then, with very high probability, the verifier is convinced of this fact, after interacting with the prover. If $x \notin L$, then no matter what the prover does, with very high probability, he fails to fool the verifier(into believing that x is in L). The first condition is referred to as the *completeness*, while the second condition is referred to as *soundness*.

Informally, a proof system is called *zero-knowledge* when the prover tells the verifier nothing but that $x \in L$, even though the verifier tries to trick the prover into revealing something.

Our interactive protocol is a proof of knowledge of a predicate

$$P(s, x) = [s^x \equiv g \bmod pq]$$

where $\gcd(x, \lambda(pq)) = 1$, and g is a generator of a sufficiently large subgroup of Z/nZ(λ is the Carmichael function[8]).

We propose a new protocol where a trusted center signs "g" with a private-key corresponding to the public-key which is certified by a trusted center. Thus, each user keeps "the signed g" as its secret information, but does not know the private-key with which g is signed. "g" is a generator of a sufficiently large subgroup of Z/nZ.

2.2 The Identification Scheme for Smart Cards

Like other identification schemes, our protocol is composed of two phases. The first phase is the smart card issuing step, and the second is interactive proof.

A trusted center randomly chooses two large primes, p and q. The two primes are kept securely by the trusted center. We assume that $n = pq$ is a 768-bit or 1024-bit public number, and the factorization of n is still infeasible. Another public value g is also chosen by the trusted center. g is a generator of a sufficiently large subgroup of Z/nZ. It should be selected by the trusted center to satisfy

$$g^k \equiv 1 \bmod n$$

where k is the order of $g \bmod n$. k must not be smooth and must be large enough(say 160-bits) to be strong against attacks such as baby-step giant-step and index calculus. Girault devised a smart way to select g in [7], but we do not need to reveal k, which is a factor of $\phi(n)$($\phi(n)$ is Euler's totient function).

The trusted center publishes $(n = pq, g)$ in the public directory. The modulus n and the subgroup generator g will be shared among a group of users.

When an eligible user applies for a smart card, the trusted center issues a string I which contains information about the user and the card. And now the trusted center performs the following steps to compute the secret information s_i and a public-key v_i for user i.

- Preparatory steps for the center :
  1. For a user i who requests a trusted center of its secret value, the trusted center chooses v_i, where v_i is a prime and $\gcd(v_i, \lambda(n)) = 1$.
  2. The center computes
     $$s_i \equiv g^{1/v_i} \bmod n.$$
     s_i can be easily obtained by computing
     $$g^{d_i} \bmod n \text{ where } v_i d_i \equiv 1 \bmod \phi(n).$$
  3. The center generates a signature
     $$S = signature(I, v_i),$$
     and the certificate
     $$C(\text{User } i) = (I, v_i, S).$$
  4. A smart card containing the user's certificate is issued.

Here, the certificate can be generated by a trusted center using any well-known signature schemes or the signature scheme proposed in this paper. Rabin-like digital signature scheme or a low encryption exponent RSA can be used for a faster certificate verification[9]. They require only a few multiplications in the verification step.

After the preparatory steps, a prover has a secret number s_i and can prove its identity to a verifier by the following interactive proof.

- Interactive proof between a prover and a verifier :
  1. Prover sends its certificate $C(\text{Prover})$ to Verifier.
  2. Verifier checks Prover's certificate $C(\text{Prover}) = (I, v_i, S)$.
  3. Prover picks a random number $r \in \{2, \ldots, n-1\}$ and sends $x \equiv r^{v_i} \bmod n$ as a test to Verifier.
  4. Verifier sends a random number $e \in \{1, \ldots, 2^t\}$ as a question to Prover, where $2^t < v_i$. The size of 2^t represents a compromise between speed and security.
  5. Prover sends to Verifier as a witness:
     $$y \equiv r s_i^e \bmod n$$
  6. Verifier checks that
     $$x \equiv y^{v_i} g^{-e} \bmod n$$
     and accepts the proof as valid only when the above result is correct.

The above interactive proof requires only one security number and one accreditation for the proof of identity, but still provides the security level of 2^{-t}. To reduce the number of bits communicated, a prover sends only the first t bits of a hash function $h(x)$ to a verifier in step 3, and compares $h(x)$ with the first t bits of $y^{v_i} g^{-e} \bmod n$. Thus, the total number of communication bits is $2t + \log n$. The protocol requires an inverse operation to compute $g^{-e} \bmod n$, but the computation is already in the preprocessing stage.

2.3 Soundness and Completeness Proofs

For the validity and security of the proposed interactive proof, completeness and soundness of the protocol should be considered. In this section, we give formal proofs of the protocol.

It is trivial that the identification procedure succeeds if both a prover and a verifier follow the protocol as shown in theorem 1.

THEOREM 1 (COMPLETENESS). *If a prover and a verifier follow the protocol, the verifier always accepts prover's proof of identity.*

PROOF. By definition

$$y^{v_i} g^{-e} \equiv (r s_i^e)^{v_i} g^{-e} \equiv r^{v_i} g^e g^{-e} \equiv r^{v_i} \equiv x \bmod n$$

□

The proof that the cheating probability of a dishonest prover cannot be increased to higher than 2^{-t} is similar to Guillous and Quisquater's.

THEOREM 2 (SOUNDNESS). *Assume that a prover does not know the s_i and cannot compute in polynomial time the v_i-th root. If a verifier follows the protocol, he will accept the proof as valid with probability bounded by 2^{-t}.*

PROOF. To increase the probability of cheating, a prover must choose x in such a way that he can compute v_i-th roots y' and y'' of

$$x g^e \bmod n$$

for two questions e' and e''. However, the following procedure reveals s_i, and it contradicts the assumption that the v_i-th root cannot be computed in polynomial time without knowing s_i.

1. For two questions e' and e'', choose Bezout coefficients m and k such that

$$v_i m + (e' - e'')k = \pm 1$$

There always exist Bezout coefficients m and k, because of $\gcd(v_i, e' - e'') = 1$.

2. Compute the following and output:

$$(g^m \left(\frac{y'}{y''}\right)^k)^{\pm 1} \equiv (s_i^{v_i m} \left(\frac{y'}{y''}\right)^k)^{\pm 1} \equiv (s_i^{v_i m} s_i^{(e'-e'')k})^{\pm 1}$$

$$\equiv s_i \bmod n$$

□

If the question e is predicted by a prover, she can deceive the verifier by guessing e such that v_i divides $r + e$ and sending $x \equiv g^r \bmod n$ and $y \equiv g^{(r+e)/v_i} \bmod n$. A verifier will accept the proof of knowledge, because

$$y^{v_i} g^{-e} \equiv (g^{(r+e)/v_i})^{v_i} \equiv g^{r+e} g^{-e} \equiv x \bmod n.$$

However, the probability of this event is 2^{-t}, and for a sufficiently large t, it is almost impossible to guess a question e. There is a tradeoff between speed and security. If a security parameter t has a large value, the speed becomes slow. For the identification scheme, $t = 20$ is enough for the most applications, $t = 45$ for national security applications. For the digital signature, $t = 72$ is enough.

In the protocol, ($x \equiv r^{v_i} \bmod n$) is a random number, and y is masked by a random number r which cannot be computed by a verifier who does not know the factorization. Thus, all the messages from a prover to a verifier do not reveal any information about the prover's secret, s_i. The protocol "with one round", however, is not a zero-knowledge proof, because the running time of the simulator which produces transcripts with an identical probability distribution to those produced when Prover actually takes part in the protocol is $O(2^t)$. Technically for soundness, t must grow asymptotically faster than $\log n$, but the condition makes the running time of the simulator grow asymptotically faster than $(\log n)^c$, for a constant c. Thus, our protocol with one round is not a zero-knowledge proof. However, the protocol "with k-rounds ($k > 1$)" has surely the zero-knowledgeness. If we perform the protocol with k-rounds, the simulator runs in $O(2^t)$ and for the soundness, kt (not t) must grow asymptotically faster than $\log n$. If we chooses k large enough for kt to grow faster than $\log n$ for a t that grows slower than $\log n$, both the soundness and the zero-knowledgeness are obtained.

The protocol with k-rounds, however, is not efficient to lose the practical meaning. If we restrict $k = 1$, the protocol loses the zero-knowledgeness in theoretical sense but it gets efficiency instead. Perfect zero-knoledge in practice is hard to get, and only Fiat-Shamir's scheme among the referred ones in the paper obtains it. Practically, we don't need to insist the zero-knowledgeness in the asymptotic sense, sacrificing the efficiency. Proper selection of parameters such as k and t is enough to protect the secret information.

2.4 Why is v_i a Prime Number?

We show that the probability of cheating may be higher than 2^{-t}, unless v_i is a prime. For this, we assume that v_i is not a prime.

Suppose a dishonest prover i who has (s_i, v_i) such that $s_i^{v_i} \equiv g \bmod n$ and pretends to be a prover j. If $\gcd(v_i, v_j) \neq 1$ such as $v_i = la$, $v_j = lb$ and $b < 2^t$, then he/she can pretend to be prover j with probability $b^{-1} > 2^{-t}$ by guessing e such that b divides $e + r$, and sending $x \equiv g^r \bmod n$ and $y \equiv s_i^{(e+r)a/b} \bmod n$. Indeed,

$$y^{v_j} g^{-e} \equiv s_i^{(e+r)v_j a/b} g^{-e} \equiv s_i^{(e+r)lba/b} g^{-e} \equiv s_i^{v_i(e+r)} g^{-e}$$

$$\equiv g^r \equiv x \bmod n.$$

Thus, a dishonest prover i can cheat a verifier with probability higher than 2^{-t}. However, if v_i is a prime, this cannot happen.

2.5 Consideration on Common Modulus

The protocol in this paper uses the same security assumption as that of RSA. That is, the v-th root finding in Z/nZ is as difficult as the factorization. Simmons has shown that the protocol using common modulus in RSA fails as a privacy channel if a message was encrypted and sent to two or more receivers[14]. In that case, an outsider (not a subscriber/user) using only the public-key could decrypt the ciphertext with unacceptably high probability. Also, DeLaurentis has shown that the common modulus protocol is even more vulnerable to attack by insiders (subscriber/user) who can break the cryptosystem by using either a probabilistic or a deterministic method[3]. Knowledge of a single encryption/decryption pair is equivalent to factoring the modulus n probabilistically, and enables a subscriber to find other users' secret key deterministically. Thus, a protocol using RSA should not use common modulus.

There are two security aspects to be considered in the proposed scheme. One is whether a subscriber(a legal prover) can find d_i from $g^{1/v_i} \equiv g^{d_i} \bmod n$ and v_i. If he finds d_i, he can easily factor n by [3]. To find out d_i only from $g^{d_i} \bmod n$, he should solve a discrete logarithm problem. In fact, most sub-exponential algorithms to solve the discrete logarithm problem assume a finite field F_p or a Galois field $GF(p^m)$. The order of $g \bmod p$ or $g \bmod p^m$ is known a priori in these fields, whereas the order of $g \bmod n$ is not known. Thus, those algorithms are not applicable to find out d_i only from g^{d_i} in Z/nZ. Moreover, it is equivalent to breaking the RSA digital signature scheme to find d_i from $g^{d_i} \bmod n$ and v_i. The other to be considered is whether a subscriber can compute another subscriber's secret information $g^{1/v_j} \bmod n$ from its secret $g^{1/v_i} \bmod n$, where v_i, v_j are primes. Unless $v_i = av_j (a = 1, 2, \dots)$, one cannot compute $g^{1/v_j} \bmod n$ with g^{1/v_i} without the knowledge of the factorization[4].

3. DIGITAL SIGNATURES

This section treats digital signatures that are constructed from our identification scheme. Also, we show that the digital signature scheme is extendible to multiple digital signature scheme.

3.1 Digital Signature

Using a zero-knowledge based identification scheme and a hash function, one can construct a secure digital signature scheme[5]. The role of the trusted center in this case is the same as that in the identification scheme. In our identification scheme, random

number e contains no information, but its unpredictability prevents a prover from cheating. Switching the identification scheme to a signature scheme, we replace the verifier's role by the hash function h and obtain the following signature protocol.

- To sign a message M, Signer :
  1. Picks a random number, r, and computes
     $$x \equiv r^{v_i} \bmod n$$
     This computation is the preprocessing stage.
  2. Concatenates M and x, and hashes the result:
     $$e = h(M, x) \in \{0, \ldots, 2^t - 1\}$$
  3. Computes
     $$y \equiv r s_i^e \bmod n$$
     The signature is e and y; he/she sends these to Verifier with M, $C(\text{User } i)$.

- To verify Signer's signature on M, Verifier :
  1. Computes
     $$z \equiv y^{v_i} g^{-e} \bmod n$$
  2. Verifies whether $h(M, z)$ is e or not. Verifier accepts the message M as genuine if and only if e equals to $h(M, z)$.

THEOREM 3 (COMPLETENESS). *If Signer and Verifier follow their protocols, Verifier always accepts the signature as valid.*

PROOF. By definition,

$$z \equiv y^{v_i} g^{-e} \equiv (r s_i^e)^{v_i} g^{-e} \equiv r^{v_i} (s_i^{v_i})^e g^{-e} \equiv r^{v_i} g^e g^{-e} \equiv r^{v_i}$$

$$\equiv x \bmod n$$

and thus $h(M, z) = h(M, x)$. □

The probability distribution of the hash function $h(M, x)$ must be uniform with respect to random x. Also, $h(M, x)$ should be one-way with respect to M. The hash function which satisfies both criteria is highly resistant against known plaintext attacks and chosen plaintext attacks.

3.2 Multiple Signature

An extension of the signature scheme into the multiple signature scheme is presented. Multiple signatures are needed if many people want to sign the same document. An easy way to do this is to let each of them sign the same document separately, and concatenate the result to the signed document. With the scheme, the length and the time to sign a document is proportional to the number of people involved. Using the signature scheme designed in section 3.1, we can sign a document with more efficiency.

To sign a message, signers generate their own random numbers and use the product of those numbers as a commitment number. The random question is generated from the committed number, the message to be signed, and a hash function, so it is identical among signers. Verification process is similar to that of single signature scheme except that the public-key used in the verification is the product of v_i s.

To write a multiple signature and verify a document, each participant does the following procedure.

- To sign a message M, Signer i:
  1. Picks a random number, r_i, and computes
     $$x_i \equiv r_i^{\prod_{j=1}^{n} v_j} \bmod n$$
  2. Sends x_i to every other signers.
  3. Receives $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_n$ and computes
     $$x \equiv \prod_{j=1}^{n} x_j \bmod n$$
  4. Concatenates M and x, and hashes the result:
     $$e = h(M, x) \in \{0, \ldots, 2^t - 1\}$$
  5. Computes
     $$y_i \equiv r_i s_i^e \bmod n$$
     and sends y_i to every other signers.
  6. Receives $y_1, y_2, \ldots, y_{i-1}, y_{i+1}, y_{i+2}, \ldots, y_n$ and computes
     $$y \equiv \prod_{j=1}^{n} y_j \bmod n$$
     The signature is e and y; he/she sends these to Verifier with M and $I_1, I_2, \ldots, I_n$ which are correspond to each signer's public-key.

- To verify Signers' multiple signature on M, Verifier:
  1. Computes the following from $v_1 = f(I_1), v_2 = f(I_2), \ldots, v_n = f(I_n)$, where I_i is an identification word for Signer i:
     $$v = \prod_{i=1}^{n} v_i$$
  2. Computes
     $$z \equiv y^v g^{-e \sum_{j=1}^{n} v/v_j} \bmod n$$
  3. Verifies whether $h(M, z)$ is e or not. Verifier accepts the message M as genuine if and only if e equals to $h(M, z)$. If $e \neq h(M, z)$, there is at least one invalid signature.

Using our multiple signature scheme, the length of a signature is greatly reduced. Regardless of the number of people involved in signing procedure, the length of a signature is constantly $t + \log n$. For example, when n is a 512-bit number and the security parameter $t = 2^{72}$, total signature length of the scheme is $72 + 512$ bits, which is the same as that of the single signature. Even better, the number of signers does not increase the signature length.

As before, the completeness proof is followed by definition.

THEOREM 4 (COMPLETENESS). *If Signers and Verifier follow their protocols, Verifier always accepts the signature as valid.*

PROOF. Let v be $\prod_{i=1}^{n} v_i$. By definition,

$$
\begin{aligned}
& y^{\prod_{i=1}^{n} v_i} g^{-e \sum_{j=1}^{n} v/v_j} \\
& \equiv (y_1 y_2 \cdots y_n)^{\prod_{i=1}^{n} v_i} g^{-e \sum_{j=1}^{n} v/v_j} \\
& \equiv (r_1 r_2 \cdots r_n s_1^e s_2^e \cdots s_n^e)^{\prod_{i=1}^{n} v_i} g^{-e \sum_{j=1}^{n} v/v_j} \\
& \equiv (r_1 r_2 \cdots r_n)^{\prod_{i=1}^{n} v_i} (s_1^{\prod_{i=1}^{n} v_i} s_2^{\prod_{i=1}^{n} v_i} \cdots s_n^{\prod_{i=1}^{n} v_i})^e \\
& \times g^{-e \sum_{j=1}^{n} v/v_j} \\
& \equiv (r_1 r_2 \cdots r_n)^{\prod_{i=1}^{n} v_i} (g^{\prod_{i=1}^{n} v_i/v_1} g^{\prod_{i=1}^{n} v_i/v_2} \cdots g^{\prod_{i=1}^{n} v_i/v_n})^e \\
& \times g^{-e \sum_{j=1}^{n} v/v_j} \\
& \equiv r_1^{\prod_{i=1}^{n} v_i} r_2^{\prod_{i=1}^{n} v_i} \cdots r_n^{\prod_{i=1}^{n} v_i} \\
& \equiv x \bmod n \quad \square
\end{aligned}
$$

THEOREM 5 (SOUNDNESS). *Assume that a prover does not know the $\prod_{i=1}^{n} s_i$ and cannot compute in polynomial time the $\prod_{i=1}^{n} v_i$-th root. If a verifier follows the protocol, he/she will accept the proof as valid with probability bounded by 2^{-t}.*

PROOF. To increase the probability of cheating, a prover must choose x in such a way that he/she can compute $\prod_{i=1}^{n} v_i$-th roots y' and y'' of

$$x g^{e \sum_{j=1}^{n} v/v_j} \bmod n$$

for two questions e' and e''. However, the following procedure reveals $\prod_{i=1}^{n} s_i$, and it contradicts the assumption that the $\prod_{i=1}^{n} v_i$-th root cannot be computed in polynomial time without knowing $\prod_{i=1}^{n} s_i$.

1. Choose Bezout coefficients m, k such that

   $$m \prod_{i=1}^{n} v_i + (e' - e'')k = \pm 1, \text{where } v_i > (e' - e'')$$

   for $i = 1, 2, \ldots, n$.

   Since $\gcd(\prod_{i=1}^{n} v_i, e' - e'') = 1$, there exist k and m which satisfy the above condition.

2. Compute $\prod_{i=1}^{n} s_i$ as following :

   $$
   \begin{aligned}
   & (g^{(\prod_{i=1}^{n} v_i/v_1)m} g^{(\prod_{i=1}^{n} v_i/v_2)m} \cdots g^{(\prod_{i=1}^{n} v_i/v_n)m} \left(\frac{y'}{y''}\right)^k)^{\pm 1} \\
   & \equiv \left((s_1^{v_1})^{(\prod_{i=1}^{n} v_i/v_1)m} (s_2^{v_2})^{(\prod_{i=1}^{n} v_i/v_2)m} \right. \\
   & \left. \cdots (s_n^{v_n})^{(\prod_{i=1}^{n} v_i/v_n)m} s_1^{(e'-e'')k} s_2^{(e'-e'')k} \cdots s_n^{(e'-e'')k}\right)^{\pm 1} \\
   & \equiv \left((s_1 s_2 \cdots s_n)^{(\prod_{i=1}^{n} v_i)m + (e'-e'')k}\right)^{\pm 1} \\
   & \equiv s_1 s_2 \cdots s_n \bmod n
   \end{aligned}
   $$

$\square$

4. PERFORMANCE EVALUATION

In the identification protocol and the signature scheme, pre-computations can reduce the verifier's burden as well as the prover's. A prover can pre-compute the value $x \equiv r^{v_i} \bmod n$ in its idle time, and a verifier can prepare $g^{-e} \bmod n$ in advance. When the protocol starts, the only multiplications a prover should do are $s_i^e \bmod n$. Similarly, it is enough for a verifier to compute $y^{v_i} \bmod n$ and check the validity of the proof. This pre-computation does half the verifier's computation.

# of mults	Prover	Verifier	Bytes(x, e, y)
FS	18 (t(k+2)/2)	18 (t(k+2)/2)	12 + 3 + 256
Schnorr*	1	216 (1.5l + 0.25t)	3 + 3 + 17.5
GQ	36 (1.5t)	72 (3t)	3 + 3 + 64
OngSchnorr*	35 (t(k+2)/2 - 1)	37 (t(k+2)/2 + 1)	3 + 3 + 64
Ours*	36 (1.5t)	36 (1.5t)	3 + 3 + 64

Table 1: Comparison with the other identification schemes : * requires a certificate verification phase.

# of mults	Sig gen	Sig verify	Length(y, e)
RSA	750	≥ 2	64
Fiat-Shamir	45	45	576 + 9
Schnorr	1	228	17.5 + 9
GQ	108	216	64 + 9
Ong-Schnorr	107	109	64 + 9
Our Scheme	108	108	64 + 9

Table 2: Comparison with the other signature schemes

Comparisons with other schemes in terms of the number of multiplications and the required storage amount are shown in Table 1 and Table 2. Pre-computations are considered. We assume that identification schemes provide a security level of 2^{-24} and signature schemes 2^{-72}. Table 1 and Table 2 show the computational load of identification schemes and signature schemes, respectively. In the case of Schnorr's, Ong-Shnorr's and our scheme, the number of multiplications to verify the certificate is not counted. As stated in Section 2.2, either Rabin-like digital signature scheme or a low encryption exponent RSA adds only a couple of multiplications in the verification step[11, 9]. We assume the square and multiply algorithm is used to exponentiate a number. In Table 1, t means the security level and l means the number of bits of q in Schnorr's scheme. k is the number of Prover's secrets which is proportional to the amount of communication.

Our scheme requires the same computational complexity and almost the same hardware both for Prover and for Verifier. This symmetricity is useful in the environment where mutual identifications are required. In the signature generation process, our scheme requires only 14% of the number of multiplications required by RSA. In the verification phase, 50% of that of Schnorr's and Guillous and Quisquater's are enough. The amount of computation of Guillous and Quisquater's during the verification can be reduced by simultaneous multiple exponentiation, but it requires more complex hardware than ours.

5. CONCLUSION

In this paper, we propose a new interactive proof based identification scheme and a digital signature. The proposed scheme is very efficient both in verification and in proving steps owing to the pre-computable protocol structure. This property makes our

scheme much more appropriate for smart cards, and provides the computational symmetricity. In addition, our identification protocol requires minimal amount of storage for secret information and minimal amount of communications.

Our protocol can be modified such that a prover uses $x \equiv g^r \bmod n$ and $y \equiv s_i^{e+r} \bmod n$. In that case, a random number r should be chosen in the range $[1, \ldots, k]$, where k is the order of $g \bmod n$. With this modified version, a verifier can check the validity of the proof as in the same way as before.

$$y^{v_i} g^{-e} \equiv s_i^{(e+r)v_i} g^{-e} \equiv (s_i^{v_i})^{e+r} g^{-e} \equiv g^{e+r} g^{-e} \equiv x \bmod n$$

Soundness proof of the modified version is the same as the original one. However, this modification requires a prover to do much more multiplications, even though those are in the pre-computation phase.

Besides the digital signature, the protocol can be extended for the key-exchange when it is used for mutual identification. Instead of prover's sending "$x = r^{v_i} \bmod n$", the alternate version of the protocol is used.

1. User i commits $x_i \equiv g^{r_i} \bmod n$, where r_i is a random number.
2. User j sends a question and her commitment: e_j and $x_j \equiv g^{r_j} \bmod n$.
3. User i sends her answer and question:
$$y_i \equiv s^{e_j + r_i} \bmod n \text{ and } e_i$$
4. User j checks $x_i \equiv g^{-e_j} y^{v_i} \bmod n$. and sends her answer:
$$y_j \equiv s^{e_i + r_j} \bmod n$$
5. User i checks $x_j \equiv g^{-e_i} y^{v_j} \bmod n$.

After doing the mutual identification protocol, User i and j can obtain a common key, $K_{ij} = K_{ji}$.

$$K_{ij} \equiv x_j^{r_i} \equiv g^{r_j r_i} \equiv g^{r_i r_j} \equiv x_i^{r_j} \equiv K_{ji} \bmod n$$

6. REFERENCES

[1] L. Babai. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[2] T. Beth. Efficient zero-knowledge identification scheme. In *Advances in Cryptology : Proceedings of EuroCrypt 88, Lecture Notes in Computer Science, Springer-Verlag*, pages 77–86. IACR, 1988.

[3] J. M. DeLaurentis. A further weakness in the common modulus protocol for the rsa cryptoalgorithm. *Cryptologia*, 8(3):253–259, 1984.

[4] J.-H. Evertse. Which new rsa signatures can be computed from some given rsa signatures? In *Advances in Cryptology : Proceedings of EuroCrypt 90, Lecture Notes in Computer Science, Springer-Verlag*, pages 83–97. IACR, 1991.

[5] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology : Proceedings of Crypto 86, Lecture Notes in Computer Science, Springer-Verlag*, pages 186–194. IACR, 1987.

[6] D. C. Gilles Brassard and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37:156–189, 1988.

[7] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. In *Advances in Cryptology : Proceedings of EuroCrypt 90, Lecture Notes in Computer Science, Springer-Verlag*, pages 481–486. IACR, 1990.

[8] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.

[9] D. H. Nyang and J. S. Song. Fast digital signature scheme based on the quadratic residue problem. *Electronics Letters*, 33(3):205–206, 1997.

[10] J. Quisquater and L. Guillou. A paradoxical identity based signature scheme resulting from zero knowledge. In *Advances in Cryptology : Proceedings of EuroCrypt 88, Lecture Notes in Computer Science, Springer-Verlag*, pages 77–86. IACR, 1988.

[11] M. O. Rabin. Digitalized signatures and public key functions as intractable as factorization. In *Technical Report, MIT/LCS/TR212, Cambrideg, Mass.* MIT Lab., 1979.

[12] C. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology : Proceedings of Crypto 89, Lecture Notes in Computer Science, Springer-Verlag*, pages 239–251. IACR, 1989.

[13] S. M. Shafi Goldwasser and C. Rackoff. Knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[14] G. J. Simmons. A weak privacy protocol using the rsa cryptoalgorithm. *Cryptologia*, 7:180–182, 1983.