An Efficient Identification Scheme Based on Permuted Patterns *

Shahrokh Saeednia

Université Libre de Bruxelles, Département d'Informatique
CP 212, Boulevard du Triomphe, 1050 Bruxelles, Belgium
saeednia@ulb.ac.be

Abstract. This paper proposes a new identification scheme based on a hard partition problem rather than factoring or discrete logarithm problems. The new scheme minimizes at the same time the communication complexity and the computational cost required by the parties. Since only simple operations are needed for an identification, our scheme is well suited for smart cards with very limited processing power. With a "good" implementation, the scheme is much faster than the Fiat-Shamir or Shamir's PKP schemes.

Keywords: Identification, NP-completeness, Smart cards.

1 Introduction

Since the introduction of zero-knowledge proofs [3], many identification schemes have been proposed from which the Fiat-Shamir [1], the Guillou-Quisquater [4] and the Schnorr [6] schemes have attracted most attention. The security of these schemes is mainly based on the intractability assumption of either factoring large integers or the discrete logarithm problem, two closely related problems that actually constitute the basis of the public key cryptography. The major disadvantage of these schemes is the important computing cost required by modular multiplications of very large integers, which is significant when the computing power offered by the processor in use is low, as is the case of smart cards.

Many efforts have been devoted to create identification schemes using other hard problems allowing a fast implementation. In 1989, Shamir [7] proposed the first identification scheme that makes use of simple operations, while relying on some NP-complete problem, known as Permuted Kernels. The scheme needs 5 moves at each iteration and 20 iterations to achieve a security level close to 10^{-6}. In 1993, Stern [8] proposed another linear scheme based on Syndrome Decoding that is a three-move one, but requires about 34 iterations to attain the same security level as in Shamir's scheme. In 1996, another scheme is proposed by Pointcheval [5] based on the Permuted Perceptrons problem that uses the same framework as Stern's scheme but needs more than 48 iterations in order to be secure in practice.

* Patent pending. This work is supported by the Belgian Ministry of Wallone Region, grant no. 981/3743.

The problem with all these schemes is, however, the high communication rate between the prover and the verifier that considerably reduces the efficiency of the schemes. The problem is due to the number of iterations in the protocols, which is closely connected to the high probability of fraud at each iteration, namely, $1/2$ for Shamir's, $2/3$ for Stern's and $3/4$ for Pointcheval's schemes.

In the following sections, we propose the first linear scheme in which the probability of fraud at each iteration is not a constant value and depends on some parameters of the scheme. This allows reducing that probability and subsequently, the number of iterations. Some initial analyses show that we may choose system parameters in such a way that this probability be reduced to $1/100$ (so that, we only need 3 iterations to attain the security level of $1/1,000,000$), while a very fast implementation is assured. Since only basical operations such as byte-multiplications are needed, the scheme may be implemented efficiently on smart cards with very limited computing power.

2 Notations and Problems

We call

- a (k, p)-identity vector, a vector of size kp in which each value belonging to Z_p occurs exactly k times.
- a (k, p)-partitionable vector, a vector of size kp in which each value belonging to Z_p occurs a multiple of p times (possibly 0); in other words, the vector may be partitioned into k disjoint subvectors of size p, each of which containing the same value from Z_p.

So, any permutation of the vector

$$(\underbrace{0,\ldots,0}_{k},\underbrace{1,\ldots,1}_{k},\ldots,\underbrace{p-1,\ldots,p-1}_{k})$$

is a (k, p)-identity vector, and any permutation of one of the p^k vectors

$$(\underbrace{e_1,\ldots,e_1}_{p},\underbrace{e_2,\ldots,e_2}_{p},\ldots,\underbrace{e_k,\ldots,e_k}_{p})$$

is a (k, p)-partitionable vector, where $e_1, \ldots, e_k \in Z_p$ (not necessarily distinct).

For example, $(1, 1, 4, 1, 1, 4, 1, 4, 4, 1, 4, 1, 1, 1, 1)$ is a $(3, 5)$-partitionable vector, as it may be partitioned into $(1, 1, 1, 1, 1)$, $(1, 1, 1, 1, 1)$ and $(4, 4, 4, 4, 4)$.

Problem 1 :

Input: Integers $p \geq 2$ and $k \geq 1$, an $m \times n$ matrix H of integers in Z_p, where $n = kp$, and a vector Y of size m of integers in Z_p.

Question: Is there a (k, p)-identity vector X such that

$$Y = HX \pmod{p}. \tag{1}$$

The problem is NP-complete even for $p = 2$, since we can show that any instance of the Ssyndrome Decoding problem [2] (page 280) may be reduced to an instance of the above problem.

SYNDROME DECODING:
Input: An $m \times n$ matrix A of 0's and 1's, a vector Z of 0's and 1's and a positive integer ℓ.
Question: Is there a binary vector X of size n with ℓ 1's such that $AX \pmod 2 = Z$?

Reduction: Suppose that we have a polynomial-time algorithm for problem 1. Now given an input (A, Z, ℓ) for the syndrome decoding, we construct the binary matrix H as follows:

$$
\begin{array}{cccc}
 & n & n & \\
H = & \multicolumn{2}{c}{\left[\begin{array}{c|c} A & \begin{matrix}0\ldots\ldots0\\ \vdots \\ 0\ldots\ldots0\end{matrix} \\ \hline \begin{matrix}0\ldots\ldots0\\ \vdots\\ \vdots \\ 0\ldots\ldots0\end{matrix} & I \end{array}\right]} & \begin{matrix} m \\ \\ n\end{matrix}
\end{array}
$$

where I is the identity matrix of size n. We also construct the binary vector

$$Y = (Z \parallel (\underbrace{1, \ldots, 1}_{n-\ell}, (\underbrace{0, \ldots, 0}_{\ell})$$

of size $m + n$. If we apply the algorithm for problem 1 to the matrix H and the vector Y, we will obtain a $(n, 2)$-identity vector X as output, such that $HX \pmod 2 = Y$, if, of course, a solution exists. Then, the first n components of X specifies a solution to the original problem, since they contain obviously ℓ 1's and $n - \ell$ 0's. If there is no solution for problem 1, then the orignal problem has obviously no solution either.

Problem 2 :

Input: Integers $p \geq 2$ and $k \geq 2$, an $m \times n$ matrix H of integers in Z_p, where $n = kp$, and a vector Y of size m of integers in Z_p.

Question: Is there a (k, p)-partitionable vector X such that $Y = HX \pmod p$.

Note that, unlike the problem 1, we require here that $k > 1$. The problem of finding a (k, p)-partitionable vector is somehow different from the problem 1, because here we are not limited to find a permutation of a fixed vector, but a vector whose entries are from a particular set of values. What makes this problem hard is the predetermined number (that is a multiple of p) of each value appearing in the solution. Even for $k = 2$, the problem is NP-complete, as any instance of the 3-Dimensional Matching problem [2] (page 221) may be transformed to problem 2.

3-DIMENSIONAL MATCHING (3DM):
Input: Set $T \subseteq U \times V \times W$, where U, V and W are disjoint sets having the same number ℓ of elements.
Question: Is there a subset $T' \subseteq T$ such that $|T'| = \ell$ and no two elements of T' agree in any coordinate?

Reduction: First, let us noticee that any instance of 3DM may be seen as a binary matrix A with 3ℓ rows and $|T|$ columns, where each element of T is represented in a column of A:

$$\text{for } 1 \le i \le \ell \qquad a_{ij} = \begin{cases} 1 & \text{if } t_{j1} = u_i \\ 0 & \text{otherwise} \end{cases} \qquad 1 \le j \le |T|$$

$$\text{for } \ell + 1 \le i \le 2\ell \qquad a_{ij} = \begin{cases} 1 & \text{if } t_{j2} = v_{i-\ell} \\ 0 & \text{otherwise} \end{cases} \qquad 1 \le j \le |T|$$

$$\text{for } 2\ell + 1 \le i \le 3\ell \qquad a_{ij} = \begin{cases} 1 & \text{if } t_{j3} = w_{i-2\ell} \\ 0 & \text{otherwise} \end{cases} \qquad 1 \le j \le |T|$$

In this case, the problem is to find ℓ columns whose sum is the vector $(1, \dots, 1)$. In other words, we have to find a binary vector X_0 with ℓ 1's such that $AX_0 = (1, \dots, 1)$.

Now, given a matrix A of this form, we construct the matrix H as

$$H = \begin{array}{cc|c} \overset{|T|}{\longleftrightarrow} & \overset{|T|}{\longleftrightarrow} & \\ \begin{array}{|c|}\hline \\ A \\ \\ \hline 0 \dots\dots 0 \\ \vdots \\ \vdots \\ 0 \dots\dots 0 \\ \hline \end{array} & \begin{array}{c|}\hline 0 \dots\dots 0 \\ \vdots \\ 0 \dots\dots 0 \\ \hline \\ I \\ \\ \hline \end{array} & \begin{array}{c} \\ 3\ell \\ \\ \\ |T| \\ \\ \end{array} \end{array}$$

and the vector $Y = (Y_0 \parallel Y_1)$, where

$$Y_0 = (\underbrace{1, \ldots, 1}_{3\ell}) \qquad \text{and} \qquad Y_1 = (\underbrace{0, \ldots, 0}_{\ell}, \underbrace{1, \ldots, 1}_{|T|-\ell})$$

This is precisely an instance of problem 2 with $k = 2$ and $p = |T|$. Now, suppose that we have a polynomial-time algorithm for problem 2 that outputs (if a solution exists) a $(2, |T|)$-partitionable vector X such that $HX \pmod{|T|} = Y$. Let us denote $X = (X_0 \parallel X_1)$, where $|X_0| = |X_1| = |T|$.

Because of the presence of 0's and 1's in $X_1 = Y_1$, we know that X has exactly $|T|$ 1's and $|T|$ 0's (see the definition of (k, p)-partitionable vectors). So, considering that

$$X_1 = (\underbrace{0, \ldots, 0}_{\ell}, \underbrace{1, \ldots, 1}_{|T|-\ell})$$

X_0 has obviously ℓ 1's and $|T| - \ell$ 0's.

On the other hand, because of the modulus $|T|$ (and considering the number of columns of A that is also $|T|$), one may be sure that no row of A has more than one 1 in the positions corresponding to 1's in X_0, otherwise $AX_0 \pmod{|T|}$ would not equal Y_0. This precisely means that X_0 is a solution for the 3DM problem.

It is easy to see that, if the problem 2 does not have any solution, the original 3DM problem has no solution either.

Remark: If $k \geq m$, the problem 2 may be solved in polynomial time. In fact, the problem consists of finding k integers $e_1, \ldots, e_k \in Z_p$ and k disjoint subsets $S_1, \ldots, S_k$ of the set $(1, 2, \ldots, kp)$, each of which having p elements, such that $\forall i = 1, \ldots, m$

$$\sum_{\ell=1}^{k} e_\ell \left(\sum_{j \in S_\ell} h_{ij}\right) \pmod{p} = y_i$$

Obviously, one can try to solve the problem by choosing an arbitrary partition and form a system of equations with m equations and k unknowns (the e_i's). When $k \geq m$, this system may be solved in polynomial time by attributing arbitrary values to $k - m$ unknowns and solving the resulting $m \times m$ system. On the Contrary, when $k < m$, one should solve a system in which the number of unknowns is less than that of the equations. So, the system may have a solution or not, depending on the chosen partition.

We may also combine the two problems as follows, which is more suitable for the security analysis of our scheme.

The main problem :

Input: Integers $p \geq 2$ and $k \geq 1$, an $m \times n$ matrix H of integers in Z_p, where $n = kp$, and a vector Y of size m of integers in Z_p.

Question: Is there a vector X as a combination of a (ℓ, p)-identity vector and a $(k-\ell, p)$-partitionable vector, for any value of $0 \leq \ell \leq k$, such that $Y = HX \pmod p$.

One may observe that the vector X is such that, for some permutation τ

$$X_{\tau^{-1}} = ((\underbrace{e_1, \ldots, e_1}_{p}, \ldots, \underbrace{e_{k-\ell}, \ldots, e_{k-\ell}}_{p}), \underbrace{(0, \ldots, p-1), \ldots, (0, \ldots, p-1)}_{\ell \text{ times}}).$$

It is easy to see that this problem is also NP-complete, since the problem 1 may obviously be reduced to it.

3 The Scheme

Initialization: The authority chooses a relatively small odd prime p and an $m \times n$ matrix H of integers in Z_p, where $n = kp$ for some fixed $k \geq 2$.

Key Generation: Every user chooses a random permutation π over $\{1, \ldots, n\}$ as his secret key, forms the (k, p)-identity vector

$$X = (\underbrace{0, \ldots, 0}_{k}, \underbrace{1, \ldots, 1}_{k}, \ldots, \underbrace{p-1, \ldots, p-1}_{k})_\pi$$

and computes his public key as $Y = HX \pmod p$.

Protocol:

1. $\bar{A}$ chooses a (k, p)-partitionable vector R by randomly and independently selecting k numbers $e_1, \ldots, e_k \in Z_p$ and p permutations $\sigma_1, \ldots, \sigma_p$ over $\{1, \ldots, k\}$ and computing
$$R = ((e_1, \ldots, e_k)_{\sigma_1}, \ldots, (e_1, \ldots, e_k)_{\sigma_p})_\pi$$
and sends $Z = HR \pmod p$ to $\bar{B}$.
2. $\bar{B}$ selects a random number $0 < c < p$ and sends it to $\bar{A}$.
3. $\bar{A}$ sends $V = R + cX \pmod p$ to $\bar{B}$.
4. $\bar{B}$ verifies whether V is a (k, p)-identity vector and whether $Z + cY \equiv HV \pmod p$.

If after t rounds, A correctly answered to all $\bar{B}$'s questions, $\bar{B}$ accepts A.

4 Security assessment

4.1 Completeness

Theorem 1. *If A follows the protocol, $\bar{B}$ always accepts A's proof.*

Proof. There exists a permutation τ such that

$$R_{\tau^{-1}} = (\underbrace{e_1, \ldots, e_1}_{p}, \ldots, \underbrace{e_k, \ldots, e_k}_{p})$$

and

$$X_{\tau^{-1}} = (\underbrace{(0, \ldots, p-1), \ldots, (0, \ldots, p-1)}_{k \text{ times}}).$$

Hence, since p is prime,

$$cX_{\tau^{-1}} = ((0, \ldots, p-1)_\delta, \ldots, (0, \ldots, p-1)_\delta)$$

for some permutation δ. Clearly, $R_{\tau^{-1}} + cX_{\tau^{-1}}$ is a (k,p)-identity vector and any permutation of the latter remains so. Thus, $(R_{\tau^{-1}} + cX_{\tau^{-1}})_\tau = R + cX$ is also (k,p)-identity.

The second part of the verification, i.e., the fact that $Z + cY \equiv H(R + cX) \pmod{p}$ is trivial.

4.2 Security from the verifier's point of view (Soundness)

As usual, a cheating prover may guess the verifier's question c in advance and choose a (k,p)-identity vector V, compute $Z = HV - cY$ and send it in step 1. Evidently, if the verifier's question happens to be c, then the cheating prover can successfully answer to it. However, the probability of this event is just $1/(p-1)$ for one round and $(\frac{1}{p-1})^t$ for the whole protocol.

To answer all verifier's possible questions, an impersonator may clearly try to find X or a similar (k,p)-identity vector satisfying (1). However, if he finds a combination of a (ℓ,p)-identity vector and a $(k-\ell,p)$-partitionable vector for any value of ℓ (for $\ell = 0$, it will be a (k,p)-partitionable vector), he can still answer correctly to all verifier's questions. In fact, it suffices to determine a permutation τ, such that

$$X_{\tau^{-1}} = ((\underbrace{e_1, \ldots, e_1}_{p}, \ldots, \underbrace{e_{k-\ell}, \ldots, e_{k-\ell}}_{p}), \underbrace{(0, \ldots, p-1), \ldots, (0, \ldots, p-1)}_{\ell \text{ times}})$$

and set R in step 1 of the protocol as

$$R = (\underbrace{(0, \ldots, p-1), \ldots, (0, \ldots, p-1)}_{k-\ell \text{ times}}, (\underbrace{e_1, \ldots, e_1}_{p}, \ldots, \underbrace{e_\ell, \ldots, e_\ell}_{p}))_\tau$$

This guarantees that for any $0 < c < p$, $cX + R$ is a (k,p)-identity vector. However, as we mentionned above, finding such vectors is as hard as finding a (k,p)-identity vector satisfying (1).

There are, however, particular vectors X allowing to answer to a number $1 < q < p-1$ of verifier's questions. In other words, for some X there exist a vector R such that $R + cX$ is a (k,p)-identity vector for q values of c out of $p-1$. The maximum value of q depends on k and p. A simulation shows that, when $k = 1$ and p is prime, q is at most $\frac{p-1}{2}$, while for $k > 1$ or when p is composite, q may be larger, but never equal to $p-1$.

Example 1. For $p = 13$ and $k = 1$, if

$$X = (2, 0, 0, 4, 4, 7, 7, 8, 8, 9, 9, 10, 10)$$

we may find

$$R = (3, 3, 1, 3, 5, 3, 8, 9, 3, 10, 3, 3, 11)$$

which allows that $R+cX$ be a $(1, 13)$-identity vector for $c = 1, 2, 4, 8, 10, 11$. It is easy to verify that for no other value of $0 < c < p-1$, $R+cX$ is a $(1, 13)$-identity vector.

In the following, we will show that finding a solution of (1) allowing to answer to more than one verifier's question is computaionally equivalent to finding an identity or a partitionable vector or a combination of them satisfying (1). In particular, we examine the case when we are looking for an X satisfying (1) and for which $q = 2$, and we show that even finding such a vector is difficult.

To do so, we may adopt two approaches:

- Choose two arbitrary (k, p)-identity vectors V_1 and V_2 corresponding to arbitrary questions c_1 and c_2, and compute X as $(V_1 - V_2)/(c_1 - c_2)$. In this case, if any permutation δ of X is a solution of (1) then we can answer to two questions out of $p - 1$, namely by $(V_1)_\delta$ and $(V_2)_\delta$.
  In fact, it suffices to compute $R = (V_1 - c_1X)_\delta$ and send $Z = HR$ to the verifier in step 1 of the protocol. Now, if the verifier's question happens to be c_1 or c_2, then (k, p)-identity vectors V_1 and V_2 will be the appropriate answers, respectively.
  However, finding a permutation of such an X that satisfies (1) is actually an instance of the Permuted Kernels problem, which is known to be NP-complete in the strong sense [7].
- Compute an X such that $Y = HX$ and search two (k, p)-identity vectors V_1 and V_2 such that $V_1 - V_2 = (c_1 - c_2)X$ for some $0 < c_1, c_2 < p$[1]. In this case, we may answer to verifier's questions c_1 and c_2. However, finding V_1 and V_2 is actually an instance of the Numerical Two-Dimensional Matching that is also NP-complete in the strong sense [2] (page 224).

Note that, in the latter case, X may contain subvectors that are (ℓ, p)-identity or (j, p)-partitionable, with $j + \ell < k$. In this case, we may reduce the problem to a smaller one by just considering the subvector with $k - (\ell + j)$ elements of X that are neither in the (ℓ, p)-identity nor in the (j, p)-partitionable subvectors.

4.3 Security from the prover's point of view (Zero-knowledge)

The above protocol is not zero-knowledge, because R (and consequently V) is at last permuted with π. In fact, a cheating verifier who receives $V = R + cX$ would know that

$$V_{\pi^{-1}} = R_{\pi^{-1}} + cX_{\pi^{-1}} = ((e_1, \ldots, e_k)_{\sigma_1}, \ldots, (e_1, \ldots, e_k)_{\sigma_p}) + c(\underbrace{0, \ldots, 0}_{k}, \ldots, \underbrace{p-1, \ldots, p-1}_{k})$$

[1] The necessary condition in order that such V_1 and V_2 exist is that $\sum_{i=1}^{n} x_i \equiv 0 \pmod{p}$.

without knowing the partial permutations σ_i and the k values e_j. So, in order to find π or an equivalent permutation, he may compute a (k,p)-identity vector, that we call $V_{\pi'^{-1}}$, by attributing values to e_j's and selecting some permutations for σ_i's, which allows to derive π' afterward from it and V. Now, the correctness of π' may be verified by checking whether

$$H \cdot (\underbrace{0, \ldots, 0}_{k}, \ldots, \underbrace{p-1, \ldots, p-1}_{k})_{\pi'} \pmod p = Y.$$

The number of all possibilities in this case is straightforwardly $p^k(k!)^p$ that is also the number of possible choices for R. As we can see, this quantity grows exponentially with both p and k, so that for appropriate values of them, finding π becomes infeasible in practice (see the following section).

Note that, the fact that the e_i's are chosen independently in the protocol is of particular importance, because otherwise, the knowledge of V would reveal some significant information about X. For example, if we require that e_j's be all different (that is only possible when $k \leq p$), then no pair of entries in V corresponding to equal values in X may never be equal. Consequently, by fixing a position ℓ in V and discarding all positions that have the same value as V_ℓ, after considering a polynomial number of vectors V, only k positions remain that correspond obviously to those of equal values in X.

If we continue this way, we can sequentially identify all the p subvectors of X, each of them having the same value, without knowing their exact value. Hence, trying $p!$ permutations, X may be determined. The attack would obviously fail for not too small values of p, but may succeed if p is small, whatever the value of k.

To conclude this section, let us notice that if we interchange the role of X and R in the protocol, i.e., if X is a (k,p)-partitionable vector and R is a (k,p)-identity one, not only the protocol functions correctly, but also this allows us to consider one additional verifier's questions ($c = 0$). The reason why we did not adopt this choice is just because the protocol would no longer remain secure as it leaks enough information to retrieve X from some values of V.

5 Parameters and Performances

First of all, we recall that the number of iterations t is closely connected to the value of p. As mentionned above, a cheating prover may prepare to answer correctly to 1 verifier's questions out of $p-1$ possible questions. So, t should be determined such that $(p-1)^t$ is sufficiently high. For instance, for $p = 101$ we need 3 iterations to achieve a security level of 10^{-6}.

On the other hand, the number of possible vectors R relies on the choice of k and p. In fact, the number of basic (k,p)-partitionable vectors (i.e., without considering partial permutations σ_i) is p^k, while the number of permutations in each subvector (to which σ_i is applied) is $k!$. This means that, the total number of possible (k,p)-partitionable vectors is $p^k(k!)^p$. We propose to choose p and k

such that this number be at least 2^{512}, which is sufficiently large for our purposes. For example, for the above choice of p and with $k = 5$, this number is almost 2^{733}.

We also propose to take $m = \lceil n/2 \rceil = \lceil kp/2 \rceil$. Thus, H will be a large matrix and it seems that we deal with a large matrix product HR. However, since we know the number of occurrences of each value e_i in R (that is p), we may reduce the matrix product HR into an $m \times k$ matrix product, by means of k passes on R, which minimize the computational cost required by the prover[2].

The matrix product algorithm works as follows:

```
Z = (0,...,0)
FOR j = 1 to k DO
    W = (0,...,0)
    FOR i = 1 to n DO
        IF r_i = e_j THEN
            FOR ℓ = 1 to m DO      w_ℓ := w_ℓ + H_ℓi      END
        END
    END
    FOR i = 1 to m DO      z_i := z_i + (w_i × e_j)      END
END
```

Although this algorithm performs kn tests (that are not necessary in classical matrix product methods), the number of multiplications is reduced by $1/p$, which allows us to consider large matrices.

For the above values, a matrix product of this form needs $mk = 1265$ multiplications of 7-bit integers, in our case. Since we only need 3 iterations, the prover has to perform 3795 byte-multiplications in the whole protocol that, is less than what is needed for one multiplication of two 512-bit integers. If we consider the kn tests, our scheme with only one key is more than 10 times faster than the Fiat-Shamir scheme (with 5 keys), and may be efficiently implemented on smart cards with limited processing power.

The following table shows the security level, the number of possible R's and the number of byte-multiplications for some choices of p and k.

	n	m	t	$p^k(k!)^p$	Security level	Byte multiplications
$p = 17$, $k = 30$	510	255	5	$\approx 2^{1950}$	2^{-20}	38250
$p = 37$, $k = 14$	518	259	4	$\approx 2^{1415}$	6^{-8}	14504
$p = 101$, $k = 5$	505	253	3	$\approx 2^{737}$	10^{-6}	3795

To compare with other linear schemes, let us recall that, unlike all previous proposals, in our scheme the probability of fraud at each iteration, is a function of p and may be decreased arbirarily, by choosing larger values of p. In the following table, we consider our scheme with $p = 101$ and $k = 5$ and we compare it with Stern's syndrome decoding [8], Shamir's permuted kernels [7] and Poincheval's

[2] The same technique allows the verifier to compute HV efficiently.

permuted perceptrons [5] protocols. We can see that our scheme is by far much more efficient than those schemes, from the computation and communication points of view.

	SDP Stern	PKP Shamir	PPP Pointcheval	New Scheme
Public key (bits)	256	296	144	1771
Secret key (bits)	512	384	117	3535
Bits sent by round	954	832	896	5306
Byte-multiplications by round (prover)	1260	1369	42676	1265
Number of rounds	34	20	48	3
Total bits sent	32436	16640	43008	15918
Total byte-multiplications (prover)	42840	27380	2048448	3795
Security level	$970,740^{-1}$	$968,420^{-1}$	$993,252^{-1}$	$1,000,000^{-1}$

Note that the number of byte-multiplications, in the case of Stern's and Pointcheval's schemes, is computed by transforming bit operation timings into byte-multiplication clocks.

Since we believe that most permutations are suitable for our scheme, instead of a mapping, we may choose the secret key as a vector of $n = 505$ bits that may be used by a very simple permutation algorithm to form X and R. The storage of the public key needs 1771 bits, but may be reduced to 128 bits if we consider $f(Y)$ rather than Y itself, where f is a hash function. In this case, the verifier should check, in the last step, whether $f(HV - Z \pmod p)$ equals the public key.

As is the case of all other linear schemes, the storage of the common matrix H is not necessary in our scheme. Since the above matrix product algorithm performs operations on columns of H at the same time, entries of each column of H may be generated by a simple pseudo-random function with i $(i = 1, \ldots, n)$ as the argument.

6 Conclusion

We proposed a new identification scheme whose security depends on a new partition problem. Unlike all previously proposed linear schemes, the probability of fraud at each iteration of our scheme is not constant and may be decreased by considering larger values of one of the system parameters. The scheme uses only simple operations and needs a few iterations to attain a sufficiently high security level. This precisely means that, it minimizes both computation and communication complexity. To compare with existing identification schemes, our scheme is about 10 times faster than the Fiat-Shamir scheme that is one of the fastest and the most used scheme in practice.

References

1. A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems", *Advances in Cryptology* (Proceedings of *Crypto* '86), Lecture Notes in Computer Science, vol. 263, Springer-Verlag, 1987, pp. 186-194
2. M. R. Garey and D. S. Johnson, "Computers and intractability, a guide to the theory of NP-completeness", W. H. Freeman & Co, 1979
3. S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof systems", *SIAM J. Comp.*, vol. 18, 1989, pp. 186-208
4. L. Guillou and J. J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory" *Advances in Cryptology* (Proceedings of *EuroCrypt* '88), Lecture Notes in Computer Science, vol. 339, Springer-Verlag, 1989, pp. 123-128
5. D. Pointcheval, " A new identification scheme based on the perceptrons problem" *Advances in Cryptology* (Proceedings of *EuroCrypt* '95), Lecture Notes in Computer Science, vol. 921, Springer-Verlag, 1995, pp. 319-328
6. C. Schnorr, "Efficient signature generation by smart cards", *Journal of Cryptology*, vol. 4, no. 3, 1991, pp. 161-174
7. A. Shamir, "An efficient identification scheme based on permuted kernels" *Advances in Cryptology* (Proceedings of *Crypto* '89), Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1990, pp. 606-609
8. J. Stern, "A new identification scheme based on syndrome decoding" *Advances in Cryptology* (Proceedings of *Crypto* '93), Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1994, pp. 13-21