A new paradigm for public key identification

Jacques Stern,
Laboratoire d'informatique,
École Normale Supérieure

Abstract

The present article investigates the possibility of designing zero-knowledge identification schemes based on hard problems from coding theory. Zero-knowledge proofs were introduced in 1985, in a paper by Goldwasser, Micali and Rackoff ([16]). Their practical significance was soon demonstrated in the work of Fiat and Shamir ([11]), who turned zero-knowledge proofs of quadratic residuosity into efficient means of establishing user identities. In the present paper, we propose a new identification scheme, based on error-correcting codes, which is zero-knowledge and seems of practical value. Furthermore, we describe several variants, including one which has an *identity based* character. The security of our schemes depends on the hardness of finding a word of given syndrome and prescribed (small) weight with respect to some randomly generated binary linear error-correcting code. This is, of course, not the first attempt to design a cryptographic scheme using tools from coding theory. The difference is that identification protocols do not follow the public key paradigm based on trap-door functions and described in the seminal Diffie-Hellman paper ([8]). Rather, they only require one-way functions, which opens the way to using, in a rather direct manner, simple combinatorial problems of the kind provided by coding theory. The resulting schemes compare favourably to their number-theoretic analogues.

0 Introduction

Modern cryptography is concerned with algorithms and schemes which ensure confidentiality, integrity and proof of origin for digital communications. In conventional cryptosystems, these various functionalities are provided in a setting where the transmitter and the receiver share a common key, whose secrecy is requested for proper operation. A major breakthrough took place in 1976 with the appearance of public-key cryptography ([8]). In their paper, Diffie and Hellman proposed a new concept, allowing the use of two matching keys, one for encryption and a different one for decryption. The main novel character of the concept is that the encryption key need not be kept secret. Shortly afterwards, Rivest, Shamir and Adleman invented the celebrated RSA algorithm ([29]). This algorithm is a public key system making heavy use of operations modulo a large integer n obtained by multiplying together two prime numbers and whose security is related to difficulty of factoring n. Since then, nearly all new cryptographic schemes have been based on hard problems from number

theory, despite the fact that this produces a significant computing load. One of the most noticeable exception is the well-known encryption scheme introduced by McEliece, which uses Goppa codes (see [25]). Even if the question of finding appropriate alternative techniques is considered a major open problem in the area of public key cryptography, little progress has been made.

Subsequent research in the area has been aimed at achieving simpler functionalities at a lower cost in terms of computing load. This research has been quite successful in the setting of identification, where a user attempts to convince another entity of his identity by means of an on-line communication. Of course, the transaction should not give enough information to allow anyone else to misrepresent himself as the legitimate user, including the entity carrying the identification process. A major step forward in this area was made with zero-knowledge proofs, introduced in 1985, in a paper by Goldwasser, Micali and Rackoff ([16]) and whose practical significance for public key identification was soon demonstrated in the work of Fiat and Shamir ([11]). Still, zero-knowledge based techniques have continued to rely on number theory, even though the new protocols do not exactly follow the basic public key paradigm invented by Diffie and Hellman and requiring trap-door functions ([8]). Rather, they are based on one-way functions, which is a less stringent requirement and which opens the way to using simpler techniques, more combinatorial in spirit. In 1989, there were two attempts to build identification protocols based on simple operations (see [33, 24]). The first one relied on the intractability of some coding problem but, unfortunately, was not really practical, due to its heavy communication load. The other one was based on the so-called Permuted Kernel problem (PKP). It achieved limited computing time both on the user's side and on the verifier's side and low communication complexity. Furthermore, it was well suited to the environment of small portable devices using 8 bit microprocessors.

In the present paper, we propose a new identification scheme, based on the hardness of a problem from coding theory which we call the syndrome decoding problem (SD). Besides coding theory, the security only relies on the very simple cryptographic primitive of hashing and thus offers a true alternative to number-theoretic protocols. When formalized at a suitable level of generality, this scheme is zero-knowledge. Furthermore, it seems of truly practical value: as for the PKP scheme, discussed above, it has limited computing load and communication complexity. Also, since it only uses the logical operations induced by coding and the efficient operations involved in hashing, it could easily be partially implemented in hardware, with a definite gain in terms of efficiency. The results in this paper have been announced in [35].

The paper is organized as follows: section 1 discusses the underlying hard problem from coding theory on which the scheme is based. This is in the framework of complexity theory. The identification protocol itself appears in section 2. Next, in section 3, we give formal proofs of the zero-knowledge property. Section 4 provides a more informal treatment of several variants of the basic scheme. We close the paper by mentioning various other combinatorial problems that lead to related schemes.

1 The underlying hard problem

As was mentioned in the introduction, an identification scheme uses a specific one-way function. Actually, a *one way-function* is not a mere function but a family of functions f_n, depending on an integer n, which is usually thought as a *security parameter*. In order to simplify matters, we will always assume that the domain D_n of f_n is such that, for suitable strictly increasing functions $k(n)$ and $l(n)$, $l(n) \geq k(n)$, D_n is included in $\{0,1\}^{l(n)}$ and is the image of $\{0,1\}^{k(n)}$ by a one-to-one polynomial-time function. This is a particular case of what is called polynomially-samplable. Note that our conventions imply that for $n \neq m$, D_n and D_m are disjoint.

Definition 1 *A collection of functions* $\{f_n : D_n \mapsto \{0,1\}^{r(n)}\}$ *is called* ***strongly one-way*** *if the following two conditions hold:*
i)there exists a polynomial-time algorithm F *that, on input* $x \in D_n$*, always outputs* $f_n(x)$*.*
ii) for every probabilistic polynomial-time algorithm A*, every* $c > 0$ *and all sufficiently large* n*'s,*

$$\Pr(A(f_n(X_n)) \in f_n^{-1}(f_n(X_n))) < \frac{1}{n^c}$$

where X_n *is a random variable uniformly distributed over* D_n*.*

Remark. Our notation possibly needs explanation: f_n^{-1} applied to some value y refers to the set of all pre-images of y.

Of course, the practical relevance of the above definition is questionable as is the meaning of proofs based on it. In practice, we have only constants and not parameters going to infinity. This opens up a whole potential discussion on the role of complexity-based arguments in cryptography and is not specific to the scheme presented here nor to the theorems proved in order to assess its security. There are various ways to answer this objection. Firstly, we may argue, based on experiments, that the asymptotics are actually relevant in the proposed range of parameters. This is what is usually done for schemes based e.g. on the hardness of factoring and parameters for these schemes are finely tuned, taking into account the latest progress of factoring algorithms. We will develop similar argument for our scheme further down in the present section. Secondly, we may view complexity-theoretic proofs as a mere technique to validate the design of cryptographic protocols: with this modest approach, schemes that are supported by some type of mathematical proof receive some evidence that their design is not flawed but the word proof cannot be applied without precaution to the practical implementations. Finally, we may use a specific lower bound on the security of the concrete one-way function under consideration. Accordingly, the reductions used in the proofs have to be analyzed for precise running time and lowering of the security. Since it entails extra technical and notational difficulties we have chosen not to systematically undertake this kind of analysis in a paper aimed at a large audience and to keep it for future work.

We now turn to error-correcting codes. An (n,k,d) binary linear code is a linear subspace of $\{0,1\}^n$ of dimension k, whose non zero elements have weight at least d. Here, $\{0,1\}^n$ is viewed as a vector space over the two-element field, with the usual bitwise addition and scalar multiplication. Members of $\{0,1\}^n$ are sometimes called *words* and the (Hamming)

weight of a word x, denoted by $w_H(x)$ is the number of ones it includes. The *information rate* of an (n, k, d) binary linear code is the ratio k/n and the code can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors, where $\lfloor t \rfloor$ denotes, as usual, the integer part of a real number t. An (n, k, d) binary linear code can also be defined by its parity check matrix which is an m-by-n binary matrix $(m = n - k)$, with the property that, for each vector of the code, the product (mod 2) of the matrix by the vector is zero. In the matrix vector product, the element of the code is a column vector and lies on the right of the matrix. Actually, this product can be computed for any vector and is called the syndrome. If the vector is not in the code, the syndrome is non zero.

It is well known that the question of finding the closest codeword to a vector is hard. It is also difficult to find a word of given weight from its syndrome's value (see [2]). More precisely, the following problem, stated in the style of [12], is NP-complete:

Instance An $m \times n$ binary matrix $H = (h_{ij})$, a binary non-zero vector $y = (y_1, \ldots, y_m)$, and a positive integer w.

Question Is there a binary vector $x = (x_1, \ldots, x_n)$ with no more than w 1's such that, for $1 \leq i \leq m$, $\sum_{j=1}^{n} h_{ij} \cdot x_j \equiv y_i \pmod 2$?

Comment: The variant in which we ask for an x with *exactly* w 1's is NP-complete, even for $y = (0, 0, \ldots, 0)$. If we drop the word "exactly", the question becomes open.

NP-completeness ensures that there is no polynomial-time algorithm for solving a problem in the worst case; however many NP-complete problems can be attacked after a suitable preprocessing phase or can be suitably approximated or else can be efficiently solved on average. The first issue has been discussed in a related work of Bruck and Naor ([5]), where they present a linear code, such that even with a large amount of pre-processing (based on the parity matrix only), it is still hard to produce a minimum-weight word leading to this syndrome. Non-approximability results for the minimum distance of a code appear in [1]. As for the the hardness of random instances, the question has been investigated by various researchers, especially for families of random codes with a constant information rate. Such codes can be obtained by randomly filling up a parity check matrix of the appropriate dimension with zeros and ones. It is known that these codes almost surely satisfy the Gilbert-Varshamov bound ([23]) and therefore that they can correct a constant fraction of the length of the codewords. More accurately, we let $\rho = k/n$ and we define a function $\lambda = GV(\rho)$ by the relation $1 - \rho = H_2(\lambda)$, with $0 < \lambda < 1/2$ and $H_2(x) = -x \log_2 x - (1 - x) \log_2(1 - x)$. Then, for any $\delta < \lambda$, random codes with information rate ρ almost surely decode $\left\lfloor \frac{\delta n}{2} \right\rfloor$ errors. Still, it is difficult do decode with respect to these codes and it even looks equally hard to exhibit codewords achieving the minimum distance or to find words of given syndrome whose weight is close to the minimum distance. Several probabilistic algorithms have been proposed that solve these problems (see [19, 34, 6]) but their running time is exponential. In practical terms, it appears relatively easy to design efficient probabilistic algorithms which find words of very low weight and given syndrome, and similarly for words of average weight, but the probability of success decreases exponentially with the prescribed weight, the most difficult case arising for values close to the Gilbert-Varshamov bound.

In order to encapture the notion of hardness that was just described, we make the following definition, where $\mathcal{M}(p \times q)$ denotes the set of binary matrices with p rows and q columns::

Definition 2 *Let θ, δ be in $(0,1)$; The $SD(\theta,\delta)$ collection is the set of functions $\{f_n\}$ such that:*
$D_n = \{(M,x) | M \in \mathcal{M}(\lfloor \theta n \rfloor \times n), x \in \{0,1\}^n, w_H(x) = \lfloor \delta n \rfloor\}$
$f_n : \ (M,x) \ \longmapsto \ (M, M \cdot x)$

With this definition, we can formally state the intractability assumption on which the identification scheme proposed in this paper is built. It expresses the hardness of what we call the *Syndrome Decoding problem.*

Intractability assumption *Let θ be in $(0,1)$. Then, for all δ such that $0 < \delta < 1/2$ and $H_2(\delta) < \theta$, the $SD(\theta,\delta)$ collection of functions is strongly one-way.*

As was already pointed out, the above intractability assumption does not tell much about the choice of actual parameters that guarantee that the concrete problem of finding short codewords is beyond the limits of current computing technology. A survey of known algorithms for solving this problem appears in [6] with a discussion of their possible implementations and of their actual performances. We refer the reader to this paper and we only briefly comment on some figures taken from [6], for the case $\theta = 1/2$. When $n = 256$ and $m = 128$ finding a word of weight close to 30 from its syndrome is possible and takes a few hours on a workstation. On the other hand, if we set $n = 512$, $m = 256$ and look for a word of weight 56, the workfactor for the search based on the best algorithm can be estimated as 2^{70}. Even if it is difficult to make exact comparisons, it seems that this involves a computing effort similar to what might be needed for factoring a 512 bit integer. Obviously, larger dimensions yield better security.

2 The identification scheme

2.1 Description of the scheme

The purpose of the scheme is to allow the identification of users in a system run by a central trusted authority. The proposed scheme uses a fixed $(m \times n)$-matrix H over the two-element field. This matrix is common to all users and is originally built randomly by the authority. Ideally, this means that each bit of the matrix is chosen uniformly and independantly, even if practicality may impose the use of some form of pseudo-random bit generator, as will be discussed later. H is used as a parity-check matrix and, as observed in the previous section, it provides a linear binary code with a good correcting power.

Upon registration, each user U receives a secret key s_U, chosen at random by the authority among all n-bit words with a prescribed number p of 1's. This prescribed number p is also part of the system. The public identification of the user is computed as

$$i_U = H.s_U$$

This public identification is made available in some form of directory or is certified by means of a digital signature of the authority. It may further be linked to the actual identity of the user. This allows a registered participant who wants to access a given ressource to submit his public identity in order to undertake an identification session. Once the correctness of the public identity has been checked by the entity controlling the ressource, either through the directory or by means of the certificate, the interactive identification protocol can take place.

The identification scheme relies on the notion of a *commitment*. A commitment is an electronic way to temporarily hide a sequence of bits u that cannot be changed. It is actually a two-stage process: a commit stage and a decommit stage. In the commit phase, the user transmits the image $\langle u \rangle$ of u via some public cryptographic hash function. Later, during the decommit phase, the user simply reveals the value of u and the computation of $\langle u \rangle$ can be checked. Commitments are usually achieved by hash functions. This is a standard cryptographic tool and, from a practical point of view, several hash functions such that MD5 (see [30]) or SHA (see [31]) can be used. The usual property required for hash functions is *collision-freeness*: it should be impossible to forge different sequence of bits u and v such that $\langle u \rangle = \langle v \rangle$. The existence of collision-free hash functions is a widely believed assumption and can be suitably stated in the context of complexity theory by considering a collision free family, since the hash function depends not only on its input u, but also on the security parameter n. Note that the image of u by a collision-free hash function binds the value of u by preventing the user to announce another string v at the decommit stage. Still, it does not ensure the stronger property that no partial information on the committed string u can be recovered. A drastic way to ensure this property is to model the commit function $\langle u \rangle$ by a truly random function. This hypothesis has already been used by various authors and is nicely developed in [4]. The underlying complexity-theoretic model is the Turing machine with random oracle. Another possibility is to add randomness to the hash function: instead of computing the image of u by a collision-free hash function h, one computes the image of $\rho || \rho \oplus u$, where ρ is a randomly chosen string with the same length as u and where $||$ denotes concatenation. We will call this technique *random hashing*.

We now describe the basic interactive protocol that enables any user U (which we call the prover) to identify himself to another entity (which we call the verifier). The protocol includes r rounds, each of these being performed as follows:

1. The prover picks a random n-bit word y together with a random permutation σ of the integers $\{1 \cdots n\}$ and sends commitments c_1, c_2, c_3 respectively as

$$c_1 = \langle \sigma || H(y) \rangle$$
$$c_2 = \langle y.\sigma \rangle$$
$$c_3 = \langle (y \oplus s_U).\sigma \rangle$$

   to the verifier. A permutation σ is being considered in this setting as a vector of bits which encodes it; also note that $y.\sigma$ refers to the image of y under permutation σ.

2. The verifier sends a random element b of $\{0, 1, 2\}$.

3. If b is 0, then, the prover returns y and σ. If b is 1 then, the prover reveals $y \oplus s$ and σ. Finally, if b equals 2, then the prover discloses both $y.\sigma$ and $s_U.\sigma$.

4. If b equals 0, the verifier checks that commitments c_1 and c_2, which were made in step 1, have been computed honestly. More accurately, let $\tilde{y}$ and $\tilde{\sigma}$ be the answers received from the prover at step 3, then the equations to check are as follows:

$$c_1 = \langle \tilde{\sigma} || H(\tilde{y}) \rangle$$

$$c_2 = \langle \tilde{y}.\tilde{\sigma} \rangle$$

If b equals 1, the verifier checks that commitments c_1 and c_3 were correct. Again, let us denote by $\tilde{y}$ and $\tilde{\sigma}$ the answers received from the prover at step 3. Note that from the equation

$$H(y) = H(y \oplus s_U) \oplus i_U$$

it follows that the first check amounts to the equation

$$c_1 = \langle \tilde{\sigma} || H(\tilde{y}) \oplus i_U \rangle$$

As for the second check, it reads

$$c_3 = \langle \tilde{y}.\tilde{\sigma} \rangle$$

Finally, if b is 2, the verifier checks the weight property and commitments c_2 and c_3. Denoting by $\tilde{y}$ and $\tilde{s}$ the answers received at step 3, this corresponds to verifying that $w_H(\tilde{s})$ has the prescribed value p and that the following equations hold

$$c_2 = \langle \tilde{y} \rangle$$

$$c_3 = \langle \tilde{y} \oplus \tilde{s} \rangle$$

It should be understood that the description that was just given refers to the case when commitments are simply achieved by hashing. When random hashing is used as explained above, then each commitment c_j, $j = 1, 2, 3$ uses a random string ρ_j, $j = 1, 2, 3$. Step 3 should then be replaced by the following

- If b is 0, then the prover returns y, σ and ρ_1, ρ_2. If b is 1, then the prover reveals $y \oplus s$, σ and ρ_1, ρ_3. Finally, if b equals 2, then, the prover discloses $y.\sigma$, $s_U.\sigma$ and ρ_2, ρ_3.

Obvious changes are also needed in step 4, in order to properly decommit.

The number r of consecutive rounds depends on the required level of security and will be discussed further on as well as the values of the parameters n, k, p.

2.2 Practical versions of the scheme

We first briefly discuss security issues from an informal point of view. A more formal approach will be taken in the next section. Clearly, the security of the scheme relies on the difficulty of inverting the function

$$s \longrightarrow H(s)$$

when its arguments are restricted to valid secret keys. Section 1 already gave some evidence of this difficulty. In order to propose minimum size parameters, we note that any inversion algorithm solves the problem of finding a word of weight p in the code consisting of all words x such that $H(x)$ is 0 or i. We next use the precise asymptotic evaluation of best algorithms computing codewords of small weight, recently given in [6] and confirmed by experiments in moderate sizes. This leads to the following possible sizes:

- $n = 512, m = 256, p = 56$
- $n = 768, m = 384, p = 84$
- $n = 1024, m = 512, p = 110$

These values correspond to codes with information rate 1/2. The value of p is slightly below the Gilbert-Varshamov bound. This is in accordance with the observations made in section 1. From the estimations of [6], the workfactor for the known algorithms that might reveal the secret is above $O(\Omega^k)$ where Ω is about 1.18 for the chosen values. For the minimum parameters quoted above, this yields a value close to 2^{70}, which can be considered as unfeasible. Of course, larger parameters and trade-offs between the size of the code and its dimension can be considered. Also note that the parameters here are being chosen by considering only a time estimate for the most direct attack, namely finding a secret key s_U from H and i_U. This ignores the fact that, in the reduction of the security of the scheme to inverting H, which will be given in the next section, there may be some degradation of the security level and this could mean that larger parameters are advisable. This type of analysis has already been considered in section 1 and, as was noted there, is not specific to the cryptographic scheme introduced in this paper. The only argument that we can offer is that we know of no other way to break the scheme than solving the underlying hard SD problem.

In order to counterfeit a given identity without knowing the secret key, various strategies can be used.

1. Having only y and σ ready for the verifier's query and replacing the unknown s_U by some arbitrary vector t of weight p, for the computation of the various commitments. Thus, the false prover hopes that b is 0 or 2. In the first case, he can simply disclose y and σ and in the second case, he returns $y.\sigma$ and $t.\sigma$. On the other hand, he is unable to answer when $b = 1$ and, as a result, the probability of success is $(2/3)^r$, where r is the number of rounds.

2. A similar strategy can be defined with $y \oplus s$ in place of y. In this case, the false prover hopes that b is 1 or 2

3. Having σ and both y and $y \oplus t$ ready where t is some element such that $H(t) = i$, distinct from s_U and whose weight is not p. This strategy expects that b is 0 or 1 and again yields the same probability of success.

It is fairly clear that shifting between one strategy to another has also the same probability of success.

We close this section by various remarks related to the actual performances of our scheme.

1. It might be thought that the proposed scheme requires a large amount of memory. This is not accurate: on one hand, because the operations to perform are very simple, they can be implemented in hardware in a quite efficient way; on the other hand, if the scheme is implemented, partially or totally, in software, it is not neccessary to store all of H. One can only store words corresponding to some chosen locations and extend these by a fixed software random number generator.

2. The communication complexity of the protocol is comparable to what it is in the Fiat-Shamir scheme when only one key is used. If we assume that permutations come from a seed of 120 bits via a pseudo-random generator and that hash values are 128 bits long, we obtain an average number of bits per round which is slightly above 1000 bits when $n = 512$. There is a trick that can save one hash value. It consists in replacing in step 1 c_1, c_2, c_3 by $\langle c_1||c_2||c_3\rangle$. This requires providing the missing hash value at step 3, which becomes

   (a) If b is 0, then the prover returns y, σ and c_3. If b is 1, then the prover reveals $y \oplus s$, σ and c_2. Finally, if b equals 2, then the prover discloses $y.\sigma$, $s_U.\sigma$ and c_1.

   Step 4 has also to be modified accordingly. The trick saves 128 bits per round and the resulting communication complexity is close to 900 bits on average.

3. The security of the scheme can be increased by taking n, k and p larger. The figures $n = 512$, $k = 256$ are to be considered as a minimum.

4. The heaviest part of the computing load of the prover (which is usually a portable device with a limited computing power) is the computation of $H(y)$, which is done in step 1. This load can be drastically reduced by extending the protocol to a 5-pass version which will be discussed further on in this paper.

5. Considering that the probability of any cheating strategy is bounded above by $(2/3)^r$, where r is the number of rounds, we see that the basic protocol has to be repeated 35 times in order to achieve a level of security of 10^{-6}. A key difference between the proposed schemes and previous proposals is the fact that a single round offers only security 1/3 instead of 1/2. There is a variant of our protocol that achieves security 1/2. It will be discussed further on in this paper

6. As is the case for Shamir's PKP, our scheme is not identity based. This means that public keys necessarily have to be made available by means of a directory or that they have to be certified by the issuing authority. We will consider below a variant of this scheme with an identity-based character.

7. Following [11], our identification scheme can be turned into a signature scheme as follows:

   - prepare r commitments c_1^j, c_2^j, c_3^j, $j = 1, \cdots r$ according to the instructions for step 1 and hash them together with the message m

- use the prepared commitments for step 1 and replace the verifier queries at step 2 using the successive digits of the hash value computed above in a base-3 representation
- compute the answers at step 3 as prescribed
- issue the transcript of all communications sent during the r steps as the signature of m

Note that the quantity $(3/2)^r$ now represents the workfactor needed to forge a signature: the attacker can try $(3/2)^r$ variants c_i until, by trial and error, he finds one whose hash value provides questions that he is prepared to answer. Accordingly, a larger value of r is required , say $r \geq 120$, in order to make the attack unfeasible. This leads to rather long signatures. Still, if suitably optimized, the technique might prove useful in specific scenarios, since verification is quite fast.

2.3 Comparison with other schemes

We now undertake a comparison with known zero-knowledge protocols. This is not an easy task especially if we want to address practicality: efficiency is actually dependant on the type of platform on which the protocols are implemented and on the degree of optimization of the software. Since we have not done large scale experiments with the numerous known schemes, we will restrict ourselves to a few remarks. Several parameters are meaningful when comparing schemes: the time for identification (on the prover's side), the time for verification (on the verifier's side), the communication complexity, the key size and the security level. In order to keep the discussion simple, we will ignore the latter by considering that the minimum proposed parameters for our scheme, namely $n = 512, m = 256, p = 56$ offer the same guarantee, as far as the underlying problem is concerned, as the number-theoretic analogues based on 512-bit numbers. We will also adjust the respective number of rounds so as to obtain a level of security of 10^{-6}. In the case of our scheme, this means 35 rounds.

Computing time. The various identification protocols can be organized in three groups.

- The schemes by Guillou and Quisquater ([17]), Ohta and Okamoto ([27]), Schnorr ([32]), Okamoto ([26]) all use modular exponentiation modulo a large number n, both on the prover's side and on the verifier's side. It is worth noting that, when n is 512 bits, this operation requires 768 modular multiplications on average and leads to significant computing time even on large machines: for example, on a SUN SPARC 10, about 70 milliseconds are needed for the prover and 120 milliseconds for the verifier. In a very limited computing environment, such as the one provided by smart cards, the operation simply cannot be performed unless some specific arithmetical co-processor is added. With such a device on board, 768 modular multiplications are done in approximately 500 milliseconds.
- The schemes proposed in [11, 9] and usually called the Fiat-Shamir schemes, only use modular multiplication. They are based on one or several secret keys, whose respective square modulo some fixed large number n is public. When a single key is

used, the number of modular multiplications to perform is 1.5 on average (both for the prover and for the verifier). This leads to 30 multiplications on each side for a full identification with security level 10^{-6}. This is presumably of the same order of magnitude as the 35 boolean matrix multiplications needed for our scheme. When several keys are used, the number of multiplications in each round increases but the number of rounds simultaneously decreases: with five keys and the same security level of 10^{-6}, only 14 multiplications are needed on average.

- The scheme proposed by Shamir in [24] and known as the PKP scheme, from the underlying "permuted kernel problem", on which it relies and the present scheme are quite similar in performances. Both can be implemented on a standard smart card with no arithmetical co-processor. In the case of our scheme, an implementation has been done on an SGS Thomson ST16623 card with 224 bytes of RAM, of which 140 bytes only are used for SD. One round is computed by the card in 800 ms.

Communication complexity. Clearly, the protocol proposed in the present paper cannot compete with the various schemes using modular exponentiation and mentioned above. These schemes can usually be performed in a single round, even if this is at the cost of formally losing the zero-knowledge character that will be discussed in the next section. This means that the overall communication stays in the kilobit range. The same is true for the Fiat-Shamir scheme with a large number of keys. For the single key case, the communication of each round is close to 1000 bits, which is very comparable to what we have. The figures for PKP are similar so that these protocols only differ by the number of rounds needed.

Key size. In number theoretic protocols, the key length is usually of the same size as the modulus n. This means 512 bits in the context that we discuss. It is higher for the multiple key Fiat-Shamir scheme, since each key has to be stored. On the other hand, some algorithms based on the discrete logarithm problem may have a shorter secret key: for example, in the Schnorr scheme, the secret key can be made as small as, say, 140 to 160 bits. In our proposal, the public key is 256 bits and the secret key is a vector of 512 bits with hamming weight 56. The latter can be coded as a word of 512 bits but more compact encodings, close to 256 bits, are also possible, for example by suitably decribing the successive gaps between the one's locations. The key size for PKP is similar. It is interesting to note that smaller key sizes are possible as demonstrated by a recent result of the author (see [36])

From the above discussion, we see that the identitification scheme presented in this paper will presumably appear attractive in severely limited computing environments such as those offered by smart cards. In this setting, it might be easier to implement than the single-key Fiat-Shamir protocol or the PKP scheme, that have similar performances. The slightly larger number of rounds could be an acceptable overhead in situations when identification is executed as a background task, access to the ressource being granted beforehand, on a provisional basis.

3 The zero-knowledge property

Following [9], the security of identification schemes is based on three properties:

- **The completeness property** asserts that the execution of the protocol between a prover who has the correct secret and a verifier is successful with overwhelming probability.

- **The soundness property** encaptures the notion of *knowledge*: in an identification protocol, the (secret) knowledge of the prover is demonstrated by the interaction. This means that any machine which successfully performs identification, can be suitably "modified" so that it outputs a possible secret key. This goes through the use of a *knowledge extractor*, which is given the prover's program and may run it as a subroutine in order to finally extract the secret key. In practical terms, this has the consequence that an intruder who has not been given a secret key, cannot identify himself as a regular user, provided that the computation of the secret key from the matching public key is actually hard.

- **The zero-knowledge property**, also called the *simulation property* guarantees that the execution of the protocol does not leak any information on the secret key, even if the verifier is replaced by a another machine with a somehow biased strategy aiming at extracting data from the prover. This goes through the construction of a *simulator*, i.e. a machine which recreates (at least statistically) the communication between the prover and the verifier without being given access to the secret key. In practical term, zero-knowledge ensures that repeated executions of the protocol do not provide any kind of useful information that might help an intruder to misrepresent himself as a regular user.

We now turn to formal proofs of security. This is in the framework of complexity theory. The arguments in this section will follow those in [9] and we will also use similar notations. Secret keys and public keys are related by a fixed polynomial-time predicate. Users are polynomial-time probabilistic Turing machines with a special tape, called the knowledge tape, on which they store their secret key. To remain consistent with our previous notations, we call them provers. Verifiers are also polynomial time probabilistic machines. Interaction is modelled by having the prover and the verifier share their input tape as well as another tape called the *communication tape* on which they alternately write up the messages that they broadcast to each other. Interaction ends up when the verifier enters a final state outputting one bit indicating the decision to accept or reject.

Our scheme can actually be described within this framework: the overall system in which identification is performed is modelled by a probabilistic polynomial time algorithm which, on input n, produces a random parity check matrix H of size $\lfloor \theta n \rfloor \times n$. Here θ is a fixed parameter, which has been set to $1/2$ in the practical examples given above. The algorithm also produces, on request, at most polynomially many secret key-public key pairs (s, i). These keys are such that $H(s) = i$ and s has weight $p = \lfloor \delta n \rfloor$, δ being another fixed parameter $< 1/2$ with $H_2(\delta) < \theta$ or equivalently $\delta < GV(1 - \theta)$. The protocol of section 1 is interactively executed between a prover P with public key i and a verifier V, both having

the common input (H, i). Although several users are around, we will focus on the security of the identification scheme where, at any time, there is only one protocol executing. We will successively consider the three relevant properties (completeness, soundness and simulation) and, for each one, we will recall the formal definition before entering the proofs.

3.1 Completeness

Consider an interactive protocol based on a polynomial-time predicate $\Pi(I, S)$ and executed by a prover P and a verifier V on the common input I. Let $ACC(P, V, I)$ denote the verifier's decision, where 1 stands for acceptance. Completeness asserts that, when P is given on its knowledge tape an S such that $\Pi(I, S)$ holds, then V accepts with overwhelming probability. More formally,

$$\forall c\, \exists n\, \forall I\, (|I| > n \rightarrow Pr\{ACC(P, V, I)\} > 1 - |I|^{-c})$$

In our scheme, the input I is the pair (H, i) and the polynomial-time predicate consists of the two relations $H(s) = i$ and $w_H(s) = p$. It is easily seen that, when the prover is given the appropriate s, $ACC(P, V, I)$ is always one.

3.2 Soundness

The formal definition of soundness is quite intricate. Furthermore several variants have been considered in the recent literature. The definition presented in [9] refers to the situation where the prover is a probabilistic polynomial-time machine with access to a knowledge tape. The knowledge extractor is given the prover's program and the auxiliary input contained on the knowledge tape. The starting hypothesis for using the extractor is the assumption that the prover has a non negligible probability I^{-c} of convincing the verifier on some input I, formally:

$$Pr\{ACC(P, V, I)\} > |I|^{-c}$$

The conclusion is that the extractor outputs with overwhelming probability an S such that $\Pi(I, S)$ holds. Note that the extractor is dependant on the constant c. The definition from [10] is different in that the extractor runs in *expected polynomial time* rather than polynomial time but treats provers more uniformly: for any given a and large enough I, it outputs an S such that $\Pi(I, S)$ holds, with probability $> Pr\{ACC(P, V, I)\} - |I|^{-a}$. Another definition appears in [3]. It also treats the prover uniformly but allows a *knowledge error* κ to appear in the success probability for the extractor: namely, the extractor outputs an S such that $\Pi(I, S)$ holds, within an expected number of steps bounded by

$$\frac{|I|^c}{Pr\{ACC(P, V, I)\} - \kappa(|I|)}$$

This definition allows to prove soundness for a single round of an iterative protocol: for example the knowledge error of our scheme with $r = 1$ is $2/3$.

We feel that the subtleties involved in the various definitions of soundness will only be of interest to the zero-knowledge experts and we do not go further on this topic referring to [9, 10, 3] for more information. We will take a much simpler path here by only considering the

case of a polynomial time probabilistic machine $\tilde{P}$ which operates with an empty knowledge tape, i.e. without the secret. Assuming that such a machine has a nonnegligible probability of success, we will build another machine outputting, with overwhelming probability a solution s of the equation $H(s) = i$, with the prescribed weight. We call such a solution an *acceptable key*. As above, nonnegligible means bounded from below by the inverse of a polynomial in n. Observe that the existence of a machine outputting acceptable keys contradicts our intractability assumption $SD(\theta, \delta)$. Thus, our approach already provides a strong argument towards the security of our scheme. Furthermore, the reader familiar with [9, 10, 3] will easily translate our statements into the respective formalism of these papers. In order to carry the proofs through, we will make use of a property of the family of hash functions we use, namely *collision freeness*. We first give a formal version of the security bounds stated in section 2.

Theorem 1 *Assume that some probabilistic polynomial-time adversary $\tilde{P}$ is accepted with probability $\geq (2/3)^r + \epsilon$, $\epsilon > 0$, after playing a constant number r of rounds of the identification protocol. Then there exists a polynomial-time probabilistic machine which outputs an acceptable key s from the public data or else finds collisions for the hash function, with overwhelming probability.*

Proof Consider the tree $T(\omega)$ of all 3^r executions corresponding to all possible questions of the verifier when the adversary has a fixed random tape ω. Let

$$\alpha = Pr\{T(\omega) \text{ has a vertex with 3 sons }\}$$

where the probability is taken over ω. If α is $< \epsilon$, then it is easily seen that the probability of success of the adversary is bounded by $(2/3)^r + \epsilon$: $(2/3)^r$ comes from the case where $T(\omega)$ has no vertex with 3 sons and ϵ from the other case. Thus α is at least ϵ and by resetting the adversary $1/\epsilon$ times, one finds, with constant probability, an execution tree $T(\omega)$ with a vertex having 3 sons. Repeating again, the probability can be made very close to one. Now a vertex with 3 sons corresponds to a situation where 3 commitments c_1, c_2, c_3 have been made and where the adversary can provide answers to the 3 possible queries of the verifier. Consider the answer y_0, σ_0 to the question $b = 0$, the answer w_1 (representing the expected value $y \oplus s$), σ_1 to the question $b = 1$ and, finally, the answer z_2 (representing $y.\sigma$), t_2 (representing $s.\sigma$) to the question $b = 2$. Because commitment c_1 is consistent with the answers to both $b = 0$ and $b = 1$, we have

$$c_1 = \langle \sigma_0, H(y_0) \rangle = \langle \sigma_1, H(w_1) \oplus i \rangle$$

we conclude that either a collision for the hash function has been found or else that $\sigma_0 = \sigma_1$ and $H(y_0) = H(w_1) \oplus i$. Similar arguments show that, unless a collision has been found, $z_2 = y_0.\sigma_0$ and $z_2 \oplus t_2 = w_1.\sigma_1$. We note that, since the third answer is accepted, t_2 has the prescribed weight. Writing σ in place of $\sigma_0 = \sigma_1$, we get

$$t_2 = z_2 \oplus (t_2 \oplus z_2) = (y_0 \oplus w_1).\sigma$$

so that $y_0 \oplus w_1$ also has the prescribed weight. Now, $H(y_0 \oplus w_1) = H(y_0) \oplus H(w_1) = i$ and we conclude that $y_0 \oplus w_1$ is an acceptable key.

The drawback of the algorithm described in the above proof is that the computing time is exponential in r: one can find an acceptable key with constant probability only by running the adversary $1/\epsilon$ times for randomly chosen ω and exploring each time the full execution tree $T(\omega)$ for all possible queries of the verifier. We now describe a more efficient way of using the adversary.

1. randomly select the content ω of the random tape of $\tilde{P}$. This defines an execution tree $T(\omega)$.
2. randomly select a sequence for the verifier's queries. This defines a branch b of $T(\omega)$.
3. visit all vertices along branch b. If a vertex ν of $T(\omega)$ with three sons is found at level i, then output ω, b, i, else return to step 1

Lemma 1 *Under the hypotheses of theorem 1, the probability of success of each step of the improved algorithm is bounded from below by* $\epsilon^3/10$.

Consider the set X defined by

$$X = \{\omega | T(\omega) \text{has at least } 2^r + \tfrac{\epsilon}{2}3^r \text{ branches}\}$$

We claim that X has probability at least $\epsilon/2$. Otherwise, we can bound the overall probability of success of the adversary $\tilde{P}$ by distinguishing the case $\omega \in X$ and the case $\omega \notin X$. The first case contributes by at most $\epsilon/2$ and the second by $(\frac{2}{3})^r + \epsilon/2$ and taking the sum yields a contradiction. Now, when ω is in X, we can consider the subtree of $T(\omega)$ consisting of successful executions. Since $\omega \in X$, this subtree has at least $2^r + \frac{\epsilon}{2}3^r$ branches. For any index i, $0 \leq i \leq r$ we let n_i denote the number of vertices at level i and for $0 \leq i < r$, we set $\alpha_i = n_{i+1}/n_i$. We have

$$\prod_{i=0}^{r-1} \alpha_i \geq 2^r + \frac{\epsilon}{2}3^r$$

Taking logarithms, this yields

$$\sum_{i=0}^{r-1} \log(\alpha_i) \geq \log(2^r + \frac{\epsilon}{2}3^r) \geq \log((1 - \frac{\epsilon}{2})2^r + \frac{\epsilon}{2}3^r)$$

Using a convexity inequality, this is

$$\geq (1 - \frac{\epsilon}{2})r + \frac{\epsilon}{2}r \log 3$$

Hence one of the $\log(\alpha_i)$s has to exceed $1 + \frac{\epsilon}{2}(\log 3 - 1)$, which implies

$$\alpha_i \geq 2.2^{\frac{\epsilon}{2}(\log 3 - 1)} \geq 2 + \epsilon \ln 2(\log 3 - 1)$$

Now, if we let n_{i2} and n_{i3} respectively denote the number of vertices of $T(\omega)$ with at most two sons and with three sons, we get

$$\alpha_i \leq \frac{2n_{i2} + 3n_{i3}}{n_{i2} + n_{i3}} = 2 + \frac{n_{i3}}{n_i}$$

and this shows that the proportion of vertices with 3 sons at level i is larger than $\epsilon \ln 2(\log 3 - 1)$. Putting together the inequalities we see that

- we get $\omega \in X$ with probability $\geq \epsilon/2$
- we then get a successful branch b with (conditional) probability $\geq (2/3)^r + \epsilon/2$
- we finally get a vertex with three nodes with (conditional) probability $\geq \epsilon \ln 2(\log 3 - 1)$

Taking the product, we get the lower bound $\frac{\ln 2(\log 3-1)}{4}(\epsilon)^3$ which is $\geq \frac{1}{10}\epsilon^3$

Together with the proof of theorem 1 the probabilistic estimates given by the lemma show that repeating the basic step $10/(\epsilon)^3$ times reveals an acceptable key with constant probability $\simeq 1 - 1/e$.

It should be noted that, lemma 1 actually provides some concrete security estimates: assume for example that we aim at a security level which is ϵ and that we are concerned with an attacker which performs a huge preprocessing step with running time T and has a subsequent small running time t during the interaction. This is a reasonable scenario since we are dealing with identification: there could be a time-out device for bounding t. Now, using the attacker repeatedly, we find an acceptable key in time $T + \frac{10t}{\epsilon^3}$. This can be compared with the time needed to attack the SD problem by the best known algorithms (see [6]). The figures should be convincing enough for codes of size 1024 although they cannot really justify the smaller parameter size that we suggest. But the same is true of all proofs that support various number-theoretic schemes from the literature.

Lemma 1 can be read as proving soundness with knowledge error $(2/3)^r$ in the sense of [3]. The following result achieves soundness in the sense of [9], provided that the number of rounds is not too small.

Theorem 2 *Assume that some probabilistic polynomial time adversary $\tilde{P}$ is accepted with non negligible probability after playing a number of rounds $r(n)$ of the identification protocol that is such that $\log(n) = o(r(n))$. Assume further that the hash function is collision free. Then there exists a polynomial-time probabilistic machine which outputs an acceptable key from the public data with overwhelming probability.*

Proof We use the above lemma. If $\pi(n)$ is the probability of success of $\tilde{P}$, then, since $\log n$ is an $o(r)$, we have, for n large enough $(2/3)^r < \frac{\pi(n)}{2}$. Setting $\epsilon = \frac{\pi(n)}{2}$and observing that $10/(\epsilon)^3$ is polynomially bounded, we see that a vertex with three sons will be found with overwhelming probability by operating $\tilde{P}$ only a polynomial number of times.

Remark. The hypothesis on the hash function is really needed. In [13], Marc Girault and the author have shown that, if collisions can be efficiently produced, then very dangerous attacks against the scheme can be mounted. In more practical terms, this has the consequence that 64-bit hash values cannot be considered.

3.3 The simulation property

We now turn to the zero-knowledge aspect of the protocol. As explained above, this property guarantees that the execution of the protocol does not leak any information on the secret key, even if the verifier is replaced by another machine $\tilde{V}$ with a somehow biased strategy aiming at extracting data from the prover. Following [9], we give a more precise definition:

Definition 3 *An interactive protocol between two polynomial-time machines P and V is zero-knowledge if, for every polynomial time machine $\tilde{V}$, there exists a machine S which generates, in expected polynomial time, an output having the same probability distribution as the content of the communication tape produced during the interaction of P and $\tilde{V}$.*

The expert reader will observe that the definition just given only describes a specific form of zero-knowledge, usually called *perfect zero-knowledge*. Other related definitions exist: *statistical* zero-knowledge is concerned with the situation where the simulated distribution $\mathcal{S}_n$ is indistinguishable from the original distribution $\mathcal{D}_n$, which means

$$\forall c \, \exists n \, \forall I \, \forall X \, (|I| > n \rightarrow Pr\{\mathcal{S}_{|I|}(X) - \mathcal{D}_{|I|}(X)\} < |I|^{-c})$$

Computational zero-knowledge applies to the case when the two distributions cannot be distinguished (in the same asymptotic sense) by a polynomial-time machine outputting one bit, usually called a polynomial-time test.

Thus, we need to build a *simulator*, i.e. a machine $S(\tilde{V})$, which recreates the communication between $\tilde{V}$ and P in expected polynomial-time. This uses the idea of resettable simulation from [16]: at the beginning of each round the simulator chooses at random one of the three cheating strategies described in section 2.2 and prepares the initial commitments c_1, c_2, c_3 according to the chosen strategy. Now, each strategy allows to successfully answer two of the three challenges issued by $\tilde{V}$. In case $\tilde{V}$ asks the wrong question, the simulator resets the machine for the current round. The simulator clearly runs in expected polynomial time $\frac{3r}{2}$. The main technical difficulty of the proof comes from the hash function: since the inputs for the commitments c_1, c_2, c_3 come from related data, it appears difficult to claim that they will be indistinguishable from those created by the cheating strategy, at least without any further assumption. A way to circumvent this difficulty is to assume that the hash function is actually a random function. This hypothesis has already been used in [11] and is nicely developed in [4]. The underlying complexity-theoretic model is the Turing machine with random oracle: both P and V are probabilistic oracle Turing machines, the oracle providing, upon request, specific values for the hash function. Success probabilities are taken over the random choices of the different machines and over the oracle. As pointed out in [4], the simulator has to include a simulation of the oracle, which may appear difficult since the oracle is an infinite object. The solution proposed in [4] is to allow the simulator to prescribe a small (polynomial-time) piece of the oracle and have the rest filled at random. We thus get a proof of the following.

Theorem 3 *In the random oracle model, the SD protocol is zero-knowledge.*

Note that the oracle zero-knowledge definition of [4] is primarily used to show that the corresponding model allows to do zero-knowledge proofs in one round, which is impossible in the usual model. We use the random oracle setting in a different way since it models a further property of cryptographic hash functions (besides collision-freeness). Such a property seems to be needed to perform the simulation of identification protocols which use hashing, such as SD or the PKP scheme from [24]. Since the latter does not include proofs, we can only suspect that it had the same setting in mind. In the case of SD, there is another option to prove zero-knowledge: it consists in modifying the commitment step by using

random hashing, as explained in section 2. Recall that this simply involves changing $\langle x \rangle$ into $\langle \rho || \rho \oplus x \rangle$, where ρ is a randomly chosen string with the same length as x.

Theorem 4 *When random hashing is used, the SD protocol is zero-knowledge.*

In view of the proof of theorem 3, it is enough to show that the commitments c_1, c_2, c_3 follow the same distribution when they come from a legitimate user P and when they are produced by any of the three cheating strategies. We only analyze the first strategy and leave the two remaining cases to the reader. It consists in choosing y and σ at random and replacing the unknown s_U by some arbitrary vector t of weight p. Commitments are computed as follows

$$c_1 = \langle \rho_1 || \rho_1 \oplus (\sigma || H(y)) \rangle$$
$$c_2 = \langle \rho_2 || \rho_2 \oplus (y.\sigma) \rangle$$
$$c_3 = \langle \rho_3 || \rho_3 \oplus ((y \oplus t).\sigma) \rangle$$

Now, the mapping

$$(\rho_1, \rho_2, \rho_3, \sigma, y) :\longrightarrow (\rho_1, \rho_2, \rho_3 \oplus ((s_U \oplus t).\sigma), \sigma, y)$$

is a permutation of the underlying probability space which tranforms the commitments c_1, c_2, c_3 into the corresponding commitments generated by the legitimate user. This is enough to conclude that both distributions of commitments are alike and to complete the proof of theorem 4.

4 Variants of the scheme

We now close up the theoretical evaluation of our scheme and discuss several variants at a practically oriented level, i.e. without formal proofs.

4.1 A variant which minimizes the computing load

In order to minimize the computing load, we introduce a 5-pass variant. This variant depends on a new parameter q.

- Step 1 is the same except that commitment c_1 is replaced by $\langle \sigma \rangle$. Thus $H(y)$ is not computed at this stage.
- After step 1, the verifier sends back a choice of q indices from $\{1 \cdots m\}$ (these refer to a choice of q rows of the matrix H).
- The prover answers by sending the list of bits $b_1, \cdots, b_q$ corresponding to the selected indices of the vector $H(y)$. This consitutes a kind of partial commitment for $H(y)$.
- The rest of the protocol is similar (with obvious changes for the checking step).

Of course this opens up new strategies for cheating: basically, one will try to have both y and $y \oplus t$ ready where t is some element of weight p such that $H(t)$ differs from i on a small number of bits, say h. This will increase the probability of success by an amount which is close to $\frac{1}{3}(1-\frac{h}{k})^q$. In the case $n = 512$, $m = 256$, $p = 56$, $q = 64$, $h = 15$, this extra amount is roughly 0.007 and the loss can be compensated by adding only one extra round of the protocol.

Of course, the new strategy becomes more and more successful as h decreases; for example, making $h = 4$ and keeping all other figures unchanged increases the probability of cheating successfully to 0.78. But it can be shown that finding a t as above is equivalent to finding a word s' of weight at most $p + h$ with a given syndrome $H(s') = i$ and it is believed that, when h is very small, this remains unfeasable. Of course, many other trade-offs between n, k, p, h, q are possible.

4.2 A variant which minimizes the number of rounds.

In this variant, the secret key s is replaced by a simplex code generated by $s_1, \cdots, s_v$. Recall that a simplex code of dimension v has all its non zero codewords of weight 2^{v-1} (see [22]). It is easy to construct such a code with length $2^m - 1$ and to extend the length to any larger value n. The corresponding public key is the sequence $H(s_1), \cdots, H(s_v)$.

It is unknown whether or not it is much easier to recover the family of secret vectors than to recover a single one. As a set of minimal values, we recommend $v = 7$ together with $n = 576$ and $m = 288$. This ensures consistency with our previous estimates.

We now describe one 5-pass round of a protocol that achieves identification.

1. The prover picks a random n-bit word y together with a random permutation σ of the integers $\{1 \cdots n\}$ and sends commitments c_1, c_2 respectively for $\langle \sigma, H(y) \rangle$, $\langle y.\sigma, s_1.\sigma, \cdots, s_v.\sigma \rangle$ to the verifier.

2. The verifier sends a random binary vector $b_1, \cdots, b_v$.

3. The prover computes
$$z = (y \oplus \bigoplus_{j=1}^{v} b_j s_j).\sigma$$
and sends z to the verifier

4. The verifier responds with a one bit challenge b.

5. If b is 0, the prover reveals σ. If b is 1, the prover discloses $y.\sigma$ as well as the full sequence $s_1.\sigma, \cdots, s_v.\sigma$.

6. If b equals 0, the verifier checks that commitment c_1 has been computed honestly.Note that $H(y)$ can be recovered from $H(z.\sigma^{-1})$, the sequence of public keys and the binary vector issued at step 2.

   If b equals 1, the verifier checks that commitment c_2 was correct, that the computation of z is consistent and that $s_1, \cdots, s_v$ actually form a simplex code of the required weight.

As before, this basic round can be repeated. Using arguments similar to those in section 3, it can be shown that the probability of success of a single round, when no information about the secret keys is known, cannot significantly excess $\frac{1+2^{v-1}}{2^v}$, which is essentially 1/2. On the other hand, it is clear that the communication complexity is worse than in the single-key case.

4.3 An identity based version

One attractive feature of the Fiat-Shamir scheme is that the public key can be derived from the user's identity, thus avoiding the need to link both by some signature from the issuing authority. Neither Shamir's PKP scheme nor our basic scheme have this feature. We now investigate various modifications that can turn our scheme into an identity-based scheme. We note that our identity based versions are very sensitive to the possible disclosure of secret keys: for example, if several users pool their keys, then the overall system becomes weak. As a consequence these versions should be restricted to scenarios where the secret data are not available to the (physical) users. This is presumably the case if tamper-proof devices are used.

A first possibility is to use a set of t simplex codes of dimension m. If $s_1, \cdots, s_v$ is the first of these codes, then m bits can define a specific key

$$\bigoplus_{j=1}^{v} b_j s_j$$

and therefore, assuming that the identity of a user is given by tv bits, one can define t secret keys for each user. Now, these secret keys can be used randomly to perform identification. The verifier has to store tv vectors of k bits (the images of the basis vectors of the codes), which is much less than a full directory of users. We suggest $v = 7$ $t = 6$ as a reasonable implementation.

The other possibility that we describe is a bit more intricate and is more a suggestion for further research. It uses a "master code" consisting of $2t$ vectors $s_1, \cdots, s_{2t}$ whose one-bits only cover a subset T of the possible n locations. We assume that the identity Id_U of each user is given by a balanced sequence of $2t$ bits $e_1, \cdots, e_{2t}$, i.e. a sequence with t zeros and t ones. We let $F(Id_U)$ be the vector space generated by those $H(s_j)$'s such that $e_j = 1$. The public key i_U of the user should satisfy $i \in F(Id_U)$. Key generation goes as follows: by Gaussian elimination, it is possible to find a linear combination s_U of the s_j's, $e_j = 1$ whose weight p is slightly below $\frac{|T|-t}{2}$. This is the secret key of the user, computed by the issuing authority.

The security of this variant is more difficult to analyze. Let F be the vector space generated by of all of the $H(s_j)$'s and let C be the code consisting of all words x such that $H(x) \in F$. Typically, it can be observed that C has a vector with a small number of ones located within the (unknown) set T. The dimensions should be designed in order that the weight of this vector is large enough. We suggest, as a working example, $|T| = 272$, $t = 40$, $p = 112$, $n = 1024$, $m = 512$.

5 Conclusion.

Before concluding the paper, we briefly mention another possible scheme that can be devised by replacing the $\{0,1\}$-matrix H of SD by a matrix over a finite field with an extremely small number q of elements (typically 3, 5 or 7). In this situation, the weight constraint is replaced by the constraint that the secret solution s to the equation $H(s) = i$ consists entirely of zeros and ones. Thus the underlying difficult problem is a modular knapsack. Although it is known that knapsacks can be attacked by methods based on lattice reduction (see [21, 7]), it is clear also that these methods do not apply to the modular case, at least when the modulus q is very small. Possible values for the scheme are (with the same notations as above)

- $n = 196$, $m = 128$, $q = 3$
- $n = 384$, $m = 256$, $q = 3$
- $n = 128$, $m = 64$, $q = 5$
- $n = 192$, $m = 96$, $q = 5$

One round of the protocol is performed as follows:

1. The prover picks a random vector y with coefficients from the q-element field, together with a random permutation σ of the integers $\{1 \cdots n\}$ and sends commitments c_1, c_2, c_3 respectively for $\langle \sigma, H(y) \rangle$, $\langle y.\sigma \rangle$ and $\langle (y + s \bmod q).\sigma \rangle$.
2. The verifier sends a random element b of $\{0, 1, 2\}$.
3. If b is 0, the prover reveals y and σ. If b is 1, the prover reveals $y + s \bmod q$ and σ. Finally, if b equals 2, the prover discloses $y.\sigma$ and $s.\sigma$.
4. The verifier makes the obvious checks.

The above protocol seems to indicate that our scheme is not an isolated example but is related to a fairly general paradigm that produces identification protocols from hard combinatorial problems. Besides Shamir's PKP and the present SD, other proposals have been made that belong to the same family: one was put forward by Pointcheval (see [28]) using the so-called perceptron's problem; another one, due to the author ([36]), is based on the problem of solving linear equations modulo a small prime, the unknowns being subject to the condition that they belong to some prescribed subset. A specific feature of the latter is that it achieves very small key-length both for the public and the secret key.

As a conclusion, let us repeat what we feel is the main achievement of the present paper: by defining a new practical identification scheme based on the syndrome decoding problem (SD), which only uses very simple operations, we believe that we have widened the range of techniques that can be applied in crytography.

References

[1] S. Arora, L. Babai, J. Stern and Z. Sweedyk. The hardness of approximate optima in lattices, codes and systems of linear equations, *Proc. 34th Ann. Symp. on Foundations of Computer Science*, (1993), 724–733.

[2] E. R. Berlekamp, R. J. Mc Eliece and H. C. A. Van Tilborg. On the inherent intractability of certain coding problems, *IEEE Trans. Inform. Theory,* (1978) 384–386.

[3] M. Bellare and O. Goldreich. On defining proofs of knowledge, *Proceedings of Crypto 92*, Lecture Notes in Computer Science 740, 390–420.

[4] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. *Proceedings of the 1st ACM Conference on Computer and Communications Security*, (1993), 62–73.

[5] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing, *IEEE Trans. Inform. Theory,* IT-36(2) (1980), 381–385.

[6] F. Chabaud. On the security of some cryptosystems based on error correcting codes. *Proceedings of Eurocrypt 94*, Lecture Notes in Computer Science 950, 131–139.

[7] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C. P. Schnorr and J. Stern. Improved low-density subset sum algorithms, *Computational Complexity*, **2**, (1992), 11–128.

[8] W. Diffie and M. E. Hellman. New Directions in Cryptography, *IEEE Trans. Inform. Theory*, IT-22, (1976), 644–654.

[9] U. Feige, A. Fiat and A. Shamir, Zero-knowledge proofs of identity, *Proc. 19th ACM Symp. Theory of Computing*, (1987), 210–217, and *J. Cryptology*, **1** (1988), 77–95.

[10] U. Feige and A. Shamir. Witness indistinguishability and witness hiding protocols. *Proc. 22nd ACM Symp. Theory of Computing*, (1990), 416–426.

[11] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems, *Proceedings of Crypto 86*, Lecture Notes in Computer Science 263, 181–187.

[12] M. R. Garey and D. S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* W. H. Freeman and Co, 1979.

[13] M. Girault and J. Stern. On the length of cryptographic hash values used in identification schemes, *Proceedings of Crypto 94*, Lecture Notes in Computer Science 839, 202–215.

[14] O. Goldreich, R. Impagliazzo, L.A. Levin, R. Venkatesan and D. Zuckerman. Security preserving amplification of hardness, *Proc. 31st Ann. Symp. on Foundations of Computer Science*, (1990), 318–326.

[15] O. Goldreich. *Foundations of cryptography (Fragments of a book).* Weizmann Institut of Science, 1995.

[16] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems, *Proc. 17th ACM Symp. Theory of Computing*, (1985), 291–304.

[17] L.S. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory, *Proceedings of Eurocrypt 88*, Lecture Notes in Computer Science 339, 123–128.

[18] R. Impagliazzo, L. Levin and M. Luby. Pseudo-random generation form one-way functions, *Proc. 21st ACM Symp. Theory of Computing*, (1989), 12–24.

[19] J . S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Trans. Inform. Theory*, IT-34(5): 1354–1359.

[20] L. Levin. One-way functions and pseudo-random generators, *Combinatorica*, 7 (1987), 357–363.

[21] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems, *J. Assoc. Comp. Mach.* **32** (1985), 229–246.

[22] F. J. MacWilliams and N. J. A. Sloane. The theory of error-correcting codes, North-Holland, Amsterdam-New-York-Oxford (1977).

[23] J. N. Pierce. Limit distributions of the minimum distance of random linear codes, *IEEE Trans. Inform. Theory*, (1967), 595–599.

[24] A. Shamir. An efficient identification scheme based on permuted kernels, *Proceedings of Crypto 89*, Lecture Notes in Computer Science 435, 606–609.

[25] R. J. McEliece. A Public-Key System Based on Algebraic Coding Theory, Jet Propulsion Lab, *DSN Progress Report 44*, (1978), 114–116.

[26] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes, *Proceedings of Crypto 92*, Lecture Notes in Computer Science 740, 31–53.

[27] K. Ohta and T. Okamoto. A modification of the Fiat-Shamir scheme, *Proceedings of Crypto 88*, Lecture Notes in Computer Science 403, 232–243.

[28] D. Pointcheval. A new identification scheme based on the perceptrons problem, *Proceedings of Eurocrypt 94*, Lecture Notes in Computer Science 921, 318–328.

[29] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems, *Comm. of the ACM* 21-2 (1978), 120–126.

[30] R. L. Rivest. The MD5 Message Digest Algorithm. *Proceedings of Crypto 90*, Lecture Notes in Computer Science 537, 303–311.

[31] U. S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard. Federal Information Processing Standard Publication 180, 1993.

[32] C.P. Schnorr. Efficient signature generation by smart cards, *J. Cryptology*, 4 (1991), 161–174.

[33] J. Stern. An alternative to the Fiat-Shamir protocol, *Proceedings of Eurocrypt 89*, Lecture Notes in Computer Science 434, 173–180.

[34] J. Stern. A method for finding codewords of small weight, *Coding Theory and Applications*, Lecture Notes in Computer Science 388, 106–113.

[35] J. Stern. A new identification scheme based on syndrome decoding. *Proceedings of Crypto 93*, Lecture Notes in Computer Science 773, 13–21.

[36] J. Stern. Designing identification schemes with keys of short size, *Proceedings of Crypto 94*, Lecture Notes in Computer Science 839, 164–173.