Public Key Authentication with one (on-line) Single Addition

Marc Girault and David Lefranc

France Télécom R&D
42 rue des Coutures
F-14066 Caen, France
{marc.girault,david.lefranc}@francetelecom.com

Abstract. We focus on the GPS identification scheme implementation in low cost chips, i.e not equipped with a microprocessor (such as those embedded in some prepaid telephone cards or RFID tags). We present three solutions to decrease the overall number of manipulated bits during the computation of the answer by a factor two or three. All the solutions stand in the use of low Hamming weight parameters. The first one consists in building the private key as the product of low Hamming weight sub-keys. The second one suggests the choice of full size low Hamming weight private keys. Finally, the third solution corresponds to a variant of the basic GPS scheme in which large challenges with low Hamming weight are used. Whereas the first solution does not withdraw the need for a multiplier in the chip, the two other ones are ideally suited to low cost chips as they can be implemented with only one serial addition. Therefore, as a surprising result, one entity can be public key authenticated by doing one on-line addition only at the time of authentication!

Keywords: Low cost chips, GPS identification scheme, RFID tags, zero-knowledge.

1 Introduction

In 1989, C.P. Schnorr [11] presented an asymmetric identification scheme, based on *the discrete logarithm modulo a prime integer* problem, which contains three passes: the prover first sends a *commitment*, then receives a *challenge* from the verifier and finally sends an *answer* depending on both the challenge and private parameters.

One year before, J.J. Quisquater and L.C Guillou had presented the algorithm GQ [6], based on *the e-th root modulo a composite integer* problem, which also contains three passes. In both schemes, the first step consists in computing a commitment with one modular exponentiation; but, on average, the exponents used in the Schnorr scheme have a larger binary size than (the constant one) in GQ: as a consequence, computing commitments in the Schnorr scheme requires the manipulation of more bits than in GQ. Whereas the challenge steps are identical, a second difference between the two schemes stands in the answer: the

Schnorr one only requires one modular multiplication and one modular addition while GQ requires another (small) modular exponentiation. Thus, in this third step, the GQ answer computation manipulates more bits than the Schnorr one.

Moreover, in his article, C.P. Schnorr presented a *preprocessing* algorithm to efficiently calculate the commitments. A few years later, P. de Rooij [2] proved this solution to be insecure. However, as in most discrete-logarithm-based identification schemes, the commitment can be computed in advance, so that it does not require the exponentiation to be efficiently computed. Moreover, one can even envisage that the commitment be computed by another entity, namely a trusted third party, subsequently stored in the non-volatile memory of the chip. Thus, the Schnorr scheme, as claimed by its author, is well-designed for identification by smart cards as the computation power can be limited to the one needed to perform one modular addition and one modular multiplication.

With this scheme, C.P. Schnorr was able to efficiently use (ordinary) smart cards for authentication. But, the price of such devices still limits their wide development. Thus, chips with small computation power, typically between 500 and 2000 logical gates, called low cost chips in the following, represent a good alternative: their low price makes it usable everywhere. But, for such devices, the Schnorr scheme is no longer well-designed. Indeed, performing modular reductions and even multiplications (see subsection 2.3) are quite difficult for these devices. Then, new schemes appear, trying to modify the answer structure to decrease the computation cost.

In 1991, the GPS scheme was introduced by M. Girault in [3] and proved secure by J. Stern and G. Poupard in 1998 [10]. This scheme is quite similar to the Schnorr one except it is based on *the discrete logarithm modulo a composite integer* problem and the answer computation is easier than the Schnorr one, as it only contains one multiplication and one addition without any modular reduction.

At Financial Cryptography 2002, T. Okamoto, M. Tada and A. Miyaji presented the OTM scheme [8] based on the discrete logarithm problem; the new fact was the absence of multiplication in the answer which only contains one addition and one modular reduction. But one year later, in the same conference, J. Stern and J.P. Stern presented an efficient attack against the OTM scheme [12]. The authors presented at the same time a new scheme also based on the discrete logarithm problem; once again the main fact stood in the answer which contains no multiplication; its computation is based on a new easy operation called dovetailing (this scheme is described in appendix B).

At Information Security Conference 2002, T. Okamoto, H. Katsuno and E. Okamoto [14] presented another variant of GPS, but which does not substantially decrease the number of operations of the basic GPS scheme.

In this paper, we continue the saga of implementing cryptographic schemes in low-cost devices. We focus on the basic GPS scheme which seems to be the best designed for such a goal: the absence of modular reduction makes the optimization of the multiplication very important as most of the manipulated bits come from this operation (whereas in the Schnorr scheme, it also comes from the modular reduction). So, we first recall the GPS scheme with its computation and security properties. After recalling the baby-step giant-step algorithm in part 3, we introduce two new types of private keys: in part 4, the private key is the product of low Hamming weight numbers such that it improves the answer computation cost in comparison with the use of GPS with classical private keys, or the Stern-Stern scheme. In part 5, we present full size private keys with low Hamming weight and such that the non-zero bits are far enough from each other: a direct application is for low cost chips, as the answer computation only requires one addition so that it can be done quickly and sequentially. In part 6, we focus on the type of challenges sent to the prover. Thus, we present a variant of the classical GPS scheme in which a new set of challenges is used; we also give security proofs of such a variant. Once again, such sets make this variant of GPS ideally designed for low cost chips. With reasonable sizes of parameters, we can achieve a level of security around 32, an adequate value in many environments, by only computing one on-line addition. In a final part, we compare these three solutions with the existing schemes: the basic GPS one and the Stern-Stern one.

2 The Basic GPS Identification Scheme

2.1 The GPS scheme

The GPS identification scheme from [3, 10], such as labellized by the NESSIE project [4] in 2003, is an interactive protocol between a prover and a verifier which contains one or several rounds of three passes. It is based on the discrete logarithm modulo a composite integer problem: during a round of authentication, a user proves his knowledge of a private value s related to the public value v by the equation: $v = g^{-s} \bmod n$. More precisely, a prover holds a private key s and a public key (n,g,v) such that:

- $n = pq$ is the product of two prime integers such that factoring n is difficult (thus, different sizes of n should be used depending on the fact that n is a universal or individual public key),
- g is an element of $\mathbb{Z}_n^*$ ($\mathbb{Z}_n$ denotes the residue class ring modulo n and $\mathbb{Z}_n^*$ the multiplicative group of invertible elements in $\mathbb{Z}_n$); preferably g is of maximal order modulo n,
- $v = g^{-s} \bmod n$.

There are also four security parameters S, k, R and l defined as follows:

- $S \geq 160$ is the binary size of the private key s,
- k is the binary size of the challenges sent to the prover and determines the level of security of the scheme.
- R is the binary size of the exponents used in the commitment computation. It is typically equal to $R = S + k + 80$.
- l is the number of rounds the scheme is iterated. Theoretically, l is polynomial in the size of the security parameters; l is often chosen equal to 1.

Prover		Verifier
choose $r \in \llbracket 0, 2^R \llbracket$ compute $W = g^r \mod n$		
	$\xrightarrow{W}$	
		choose $c \in \llbracket 0, 2^k \llbracket$
	$\xleftarrow{c}$	
check $c \in \llbracket 0, 2^k \llbracket$ compute $y = r + s \times c$		
	$\xrightarrow{y}$	
		check $y \in \llbracket 0, 2^R - 1 + 2^{k+S} \llbracket$ verify $g^y v^c = W \mod n$

Fig. 1. The basic GPS identification scheme

2.2 Security of the scheme

We briefly recall the security properties of the GPS scheme (more details are given in [10]).

- An honest prover is always accepted.
- It can be shown that if a dishonest prover is able to be authenticated with a probability substantially greater than 2^{-k} (the probability of guessing the value of the challenge), then this prover can be used to recover the private key. Thus, k is called the level of security of the scheme.
- Finally, it can be proved that a passive eavesdropper cannot learn information about the private key even from "many" executions of the protocol.

2.3 Answer computation cost

The computational cost of the answer is expressed in bit additions (assuming adding a t-bit number is equivalent to adding t times one bit). The answer y is equal to the addition of r with the result of the product $s \times c$. As the final goal is to implement such a scheme in low-cost devices, the *shift and add paradigm* seems to be an adequate implementation for the multiplication. On average, c has a Hamming weight equal to $k/2$ so that adding $s \times c$ to r with this algorithm leads on average to $k/2$ additions of a S-bit number so that the computation cost is equal to $kS/2$ bit additions.

In a low cost chip, decreasing the number of bit additions is important but it is not the essential. The essential is to make the operations as serial as possible, since (costful) RAM (Random Access Memory) is very small, read/write operations in NVM (Non-Volatile Memory) are slow, and addressing is sequential rather than random. This is why, while the first method we propose only decreases the number of bit additions, the two other ones propose to compute the answer with one large number addition, a serial operation in essence.

3 The Baby-Step Giant-Step Algorithm and some Variants

In this section, we recall Shanks' *baby-step giant-step algorithm* [1] and some variants, which will be used in the following sections.

3.1 The classical algorithm

This algorithm was presented by D. Shanks in order to find discrete logarithms modulo n, where n is an integer (prime or not). Thus, given a public value $v = g^s \bmod n$ with s a S-bit secret number (we suppose S is an even integer), it recovers s in $\mathcal{O}(2^{S/2})$ in time and space. J.M. Pollard presented a variant [9] the running time of which is still $\mathcal{O}(2^{S/2})$, but uses very little space.

Shanks' algorithm consists in computing two lists of values: $\{g^i \bmod n \mid 0 \leq i \leq m-1\}$ and $\{v \times g^{-j \times m} \bmod n \mid 0 \leq j \leq m-1\}$ where $m = \lceil 2^{S/2} \rceil$. In the two sets, two values meet for one j_0 and one i_0 such that $s = i_0 + m \times j_0$.

The efficiency of this algorithm stands in the fact that it "cuts" the value s as an addition of two values.

3.2 Case of low Hamming weight secret exponents

In 1999, D.R. Stinson described variants in the case of low Hamming weight secret exponents [13]. Thus, given a public value $v = g^s \bmod n$ with s a S-bit secret number with an Hamming weight equal to t, it recovers s in $\mathcal{O}(S\binom{S/2}{t/2})$ (assuming S and t are even integers).

The algorithm uses *splitting systems*: it splits the set $[\![0, 2^S - 1]\!]$ in $S/2$ sets B_i, $0 \leq i \leq S/2 - 1$, such that any subset of $t/2$ values of $[\![0, 2^S - 1]\!]$ stands in a B_k. Thus, with such systems, it finds a decomposition of the t positions of the non-zero bits in two sets of $t/2$ values that stand respectively in one B_{j_0} and one B_{i_0}. Finally, an exhaustive search over all the possible sets of $t/2$ elements in each set B_j as in the classical algorithm (two lists) recovers the two $t/2$-sets corresponding to the positions of the non-zero bits.

Once again, this algorithm "cuts" the value s as an addition of two values.

3.3 Case of secret exponents equal to a product in a group of known order

Let $G = \langle g \rangle$ be a group of known order N, v be a public value equal to g^s in G where s is the product of two elements respectively picked in X_1 and X_2. This variant takes advantage of the structure of s (we assume than N is larger than all the possible values s) and recovers s in $\mathcal{O}(max(\#(X_1), \#(X_2)))$.

As described by J. Hoffstein and J.H. Silverman in [7], computing the two sets $\{v^{j^{-1} \bmod N} \mid j \in X_1\}$ and $\{g^i \mid i \in X_2\}$ makes two values meet in the two sets for one j_0 and one i_0; the value s is then equal to $j_0 \times i_0 \bmod N$.

The practicality of this variant relies upon the fact that the order of the group is known, which makes possible to compute the values $j^{-1} \bmod N$.

4 Low Hamming Weight Private Sub-Keys

In the second variant of the baby-step giant-step algorithm described in 3.3, note that if the order is unknown, the algorithm cannot be applied. Hence, it becomes possible to build a GPS private key as the product of smaller values, in order to optimize the computation cost of the answer, and more generally to speed up any multiplication involving the private key. A concrete solution consists in selecting private keys as equal to the product of low Hamming weight sub-keys.

The GPS scheme has this particular feature that the group generator g is of unknown order (at least to an enemy). As stated above, this can be used to optimize the computation of the answer. In the following, we make more precise the structure of the private keys that we suggest and the different properties it needs to ensure to protect this particular implementation from existing attacks.

4.1 The structure of the private key

The construction of the S-bit private key s consists in choosing randomly t numbers, s_1, s_2,..., s_t of respective binary sizes l_1, l_2,..., l_t with n_1, n_2,..., n_t non-zero bits randomly located, such that $s = \prod_{j=t}^{1} s_j$.

A first constraint stands on such a decomposition: we impose that the s be exactly (and always) a S-bit number. Thus, we need to take into account the position of the non-zero bits and impose for each s_j, a range of bit positions, denoted by $[\![0, b_j]\!]$ ($b_j < l_j$), where the n_j random bits are located such that even if all the random bits in each s_j correspond to the n_j possible most significant bits, we still obtain a S-bit value.

Finally, as in all schemes based on the discrete logarithm problem, the binary size of the private key must be at least 160 so that the classical baby-step giant-step algorithm has a complexity of around $\mathcal{O}(2^{80})$ or more.

As explained below, the goal of this approach is to optimize the computation cost of the answer $y = r + s \times c$. The trick consists in first computing $s_1 \times c$, then $s_2 \times (s_1 c)$, and going on with the other factors s_j. Thus, if c is a k-bit number, computing $s_1 \times c$ implies exactly n_1 additions of a k-bit number and the result value is a $(k + l_1)$-bit number. Then, computing $s_2 \times (s_1 c)$ implies n_2 additions of a $(k + l_1)$-bit number. If we generalize, (assuming adding a k-bit number is equivalent to adding k times one bit) we obtain as a final cost for the multiplication $\sum_{j=1}^{t} n_j \times (k + \sum_{u=1}^{j-1} l_u)$. Finally we need to add the number sc to r which requires $S + k$ bit additions. the answer computation cost is equal to $S + k + \sum_{j=1}^{t} n_j \times (k + \sum_{u=1}^{j-1} l_u)$. To optimize this computation cost equation, we need to minimize this expression considered as a function of t, l_j, n_j with still the other constraints cited above.

4.2 Other security aspects

Since the last variant of baby-step giant-step algorithm cannot be used, performing an exhaustive search seems to be the only way of retrieving s. So, the different factors s_j should be chosen such that the set of possible private keys is large enough: typically 2^{80} elements. It implies that: $\prod_{j=1}^{t} \binom{b_j+1}{n_j} \geq 2^{80}$.

Moreover, to prevent our construction from being vulnerable to the Stinson algorithm recalled above, we need to ensure that the Hamming weight of the private keys is not too low. This can be achieved by choosing numbers n_j such that their product is great enough (depending on the private key binary size).

4.3 Numerical application

The number of factors t should preferably be equal to 2; in that case, the private key s is the product of s_1 of size l_1 with n_1+1 bits equal to 1 and s_2 of size l_2 with n_2+1 bits equal to 1. To avoid the Stinson attack, we also suggest that $n_1 \times n_2$ be great enough (more than 80 for example) in order to obtain a private key with a large enough Hamming weight. Finally, if we look at the computation cost equation, it becomes: $S + k + n_2 \times (k + l_1) + n_1 \times k$ which can be rewritten as $S + k \times (n_1 + n_2 + 1) + n_2 \times l_1$.

For example, a 160-bit private key s can be chosen equal to $s_2 \times s_1$ where s_2 is a 142-bit number with 16 random bits chosen among the 138 least significant ones and s_1 a 19-bit number with 5 random bits chosen among the 16 least significant ones. With such values, we obtain private keys of average Hamming weight equal to 64, which is enough to prevent from the Stinson attack (since the complexity is then around $\mathcal{O}(2^{80})$). The cost equation becomes $22 \times k + 464$.

5 Low Hamming Weight Full Size Private Keys

In this part, we focus on full size (i.e size of n) private keys which can be used in GPS. Generally, using large private keys is not recommended for low cost chips. Here we explain how to take advantage of a full size private key to obtain a very efficient answer computation. Moreover, we also suggest in appendix A a way for efficiently storing such private keys.

5.1 Description of the private key structure

This approach consists in using a full size private key with a few non-zero bits which are distant enough to each other, so that multiplying a number with this key comes to concatening shifted versions of this number.

More precisely, assuming k is the binary size of the challenges sent to the prover, the non-zero bits of the private key must be separated by at least $k-1$ zero bits. Thus, performing $s \times c$ only consists in writing disjoint blocks of bits, each block representing the binary decomposition of the challenge. So, computing the answer $y = r + s \times c$ can be performed sequentially as all the shifted challenges are disjoint. The computation is then very well suited to low cost chips.

Moreover, the computation cost of the answer is small: if we denote by t the Hamming weight of s, it consists of t additions of blocks of k bits so that the computation cost of the answer is equal to $k \times t$ bit additions.

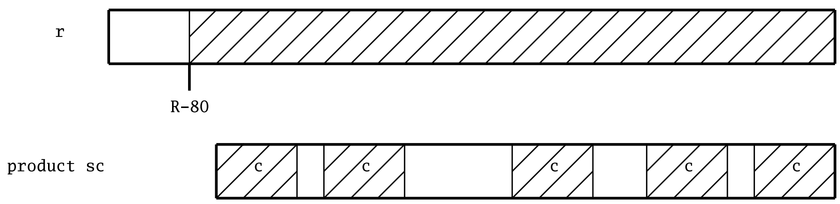

Fig. 2. A private key with a Hamming weight equal to 5

5.2 Security aspects

We first need to obtain a key space large enough; so let us explain how to determine it for a S-bit private key with a Hamming weight equal to t and with non-zero bits separated by at least $k-1$ zero bits.

As the $k-1$ bits following a non-zero bit must be equal to zero, we can reduce the evaluation of such private keys cardinality to the one of private keys of size $S-(t-1)\times(k-1)-1$ with a Hamming weight equal to $t-1$ (subtracting $(t-1)\times(k-1)$ corresponds to the numbers of bits necessarily equal to zero and 1 corresponds to the S^{th} bit, equal to 1). Thus, with this reduction, the cardinality is obviously equal to the number of ways to locate $t-1$ non-zero bits among $S-(t-1)\times(k-1)-1$, which is equal to $\binom{S-(t-1)\times(k-1)-1}{t-1}$.

So let us now explain how to construct efficiently such private keys. The first step consists in finding integer value nb_{free} and t such that $\binom{nb_{free}}{t-1}$ is greater than a required key space size. Then, in a second step, depending on the level of security required denoted by k, we obtain the private key binary size, S, as follows : $S = nb_{free} + (t-1)\times(k-1)+1$.

The second security point focus on the Stinson algorithm, recalled before. The complexity of the algorithm is then around $\binom{nb_{free}/2}{t/2}$. As shown in the below numerical application, this condition is generally ensured.

5.3 Numerical application

If we use 600 free zero bits and a private key with a Hamming weight equal to 29, the key space cardinality is then equal to $\binom{600}{28} > 2^{159}$ and the complexity of the Stinson attack is then around $\binom{300}{14} > 2^{79}$. If we now want to use the private key in environments requiring at most a level of security $k=32$, we finally obtain private keys binary size $S = 600 + 28\times 31 + 1 = 1469$ (for $k=16$, $S=1021$).

6 Low Hamming Weight Challenges

In this section, we present a new way of selecting challenges, which may be of independent interest. Usually, the set of challenges is chosen as an interval of integers $[\![0, 2^k[\![$, where k is the level of security of the scheme. But it can also be defined as a set of values verifying a particular property, in order to achieve a specific goal. For example, in the following, we show how to speed up the computation of the answer in GPS scheme by defining a challenge as a sum of "sufficiently distant" powers of two (so that multiplying by the private key comes to adding several shifted and disjoint versions of the key). This is quite similar to the solution presented in the previous section, except that the roles of the private key and the challenge are swapped. But, contrary to it (and also in section 4) in which specific private keys are suggested but the protocol is not altered, the present one slightly modifies the protocol so that the security proofs must be adapted (which can be done in a straightforward manner).

6.1 Security

In most zero-knowledge interactive protocols [5], the verifier selects a random challenge in an interval of integers $[\![0, C[\![$, and it can be proven (by an "extraction" method) that, at each protocol iteration, a fake prover can answer to at most one value of the challenge. In other words, the probability of a successful masquerade is (substantially) upper bounded by $1/C$. The logarithm in base 2 of C is often reffered to as the "level of security" of the scheme, expressed in bits. For example, if this level is equal to 32 bits, an enemy will remain undetected with probability at most 2^{-32}.

In the Schnorr scheme, the challenges are randomly picked in the interval $[\![0, 2^k[\![$, $2^{2k} < q$, where q is the order of the subgroup which is used. But they actually could be chosen in any subset of the interval $[\![0, q[\![$ of cardinality 2^k, without changing anything else in the protocol: the security proof (the so-called "soundness property") would remain valid. Note however that not any integer subset of cardinality 2^k would be convenient, since two challenges which are equal modulo q call for the same answer by the prover.

In the GPS scheme, the situation is different in two ways. First, the size R of the exponent used in the commitment computation must grow with the maximum size of the challenge, in order to maintain the zero-knowledge property. Second, any integer subset of cardinality 2^k (with a "not too large" greatest element) can take the place of the interval $[\![0, 2^k[\![$, since the order of the subgroup is, in this case, unknown (at least to the enemy). More precisely:

Definition 1 *Let n be a modulus and g an element of $\mathbb{Z}_n^*$. Let S be an integer smaller than the order of g. The short discrete logarithm problem is to find s in $[\![0, S[\![$ given g^s mod n.*

Assumption 1 *The short discrete logarithm problem is polynomially intractable.*

Theorem 1 *(using notations of section 2). Let GPS* the variant of GPS in which the set of challenges is an integer subset of cardinality B bounded by the integer C. Then GPS* is a (statistically) zero-knowledge interactive proof of knowledge of a short discrete logarithm if l, C and B are polynomial in $|n|$, lSC/R is negligible and $log(|n|) = o(l \times logB)$.*

Proof. (Hints) In the classical GPS, the challenges are randomly picked in the set $[\![0, 2^k[\![$ and the value 2^k, used during the security proofs, has two different goals. Indeed, it represents both the cardinality of the set and its upper bound. So, in order to correctly establish the security proofs of GPS*, we rely on the classical GPS proofs given in [10], by separating the two goals of 2^k.

Completeness. Nothing is modified to the classical proof, as it only consists in the verification of the equation $g^y v^c = W \mod n$.

Soundness. We can remark, as in the classical GPS, that the probability of impersonation is equal to $1/B^l$, so that we need $log(|n|) = o(l \times logB)$. Moreover, during the extraction, the running time is linear in l so that l must be polynomial in $|n|$. Finally, an exhaustive search over values smaller than C is required so that C must be polynomial in $|n|$.

Statistical zero-knowledge. First, to generate valid triplets (W,c,y), and assuming that c is not randomly chosen (in the case of a dishonest verifier), we need to try B triplets in average to obtain one with a valid c. In order to obtain l valid triplets, we need on average $l \times B$ tries. So $l \times B$ must be polynomial in $|n|$. In a second part, we need to compare the distribution of a simulation and a real communication. Relying on the proof given in [10], lSC/R must be negligible.

This approach leads to a new secure solution to speed up the multiplication in the GPS scheme, by using specific challenges. Let us now present how they are constructed.

6.2 A new set of challenges: the property

The computation of the answer y in GPS requires an addition and a multiplication ($y = r + s \times c$). Whereas in the previous part we focused on the private key construction to decrease the cost of the multiplication, here we deal with the structure of the challenges. Roughly, our proposal consists in using large and low Hamming weight challenges with the non-zero bits far enough from each other.

More precisely, if r and s are respectively R and S-bit integers, the challenge size k should be such that $R = S + k + 80$. Thus, adding the product $s \times c$ to r mainly transforms the $R - 80$ least significant bits of r.

So, if we denote by B_s the binary representation of the private key s, the present solution consists in using challenges such that adding the product $s \times c$ to r only implies the addition of disjoint blocks B_s on the $R - 80$ bits of lowest Hamming weight of r. To obtain disjoint blocks B_s, the challenges only need to have non-zero bits separated by at least $S - 1$ zero bits. Thus, with this type of challenges, there is no longer any multiplication but only one serial addition of r with a large number composed of disjoint blocks B_s.

Let us now consider the number of such challenges for a given R and S. Let u be the quotient of the Euclidean division of $R - 80$ by S; we can add at most u disjoint blocks representing the private key. The set $\mathcal{C}$ of challenges is:

$$\mathcal{C} = \left\{ \sum_{j=1}^{t} 2^{i_j} \mid \forall j \in [\![1, t-1[\![\ i_{j+1} \geq i_j + S, \ i_0 \geq 0, i_t < R - 80 - S, \ t \leq u \right\}$$

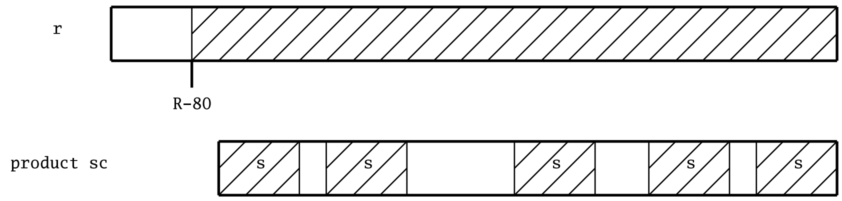

Fig. 3. A large challenge with an Hamming weight equal to 5

6.3 Cardinality of the set

Let us denote by $\mathcal{NB}_{R,S}(h)$ the number of challenges in $\mathcal{C}$ inducing $h \leq u$ additions of blocks B_s for a S-bit private key s and commitment exponents r of size R. First, we consider in the following that the challenges size is at most $R - S - 80$ bits. As in the previous section, for a given h, $\mathcal{NB}_{R,S}(h)$ is also equal to the number of ways to locate h non-zero bits among $R - 80 - S - h \times (S - 1)$ since for each non-zero bit, the $S - 1$ following ones must be equal to zero. So we obtain that $\mathcal{NB}_{R,S}(h) = \binom{R-80-S-h\times(S-1)}{h}$. Then, the cardinality of $\mathcal{C}$ is equal to:

$$\sum_{h=0}^{u} \binom{R - 80 - S - h \times (S - 1)}{h}$$

It is quite obvious that $\mathcal{NB}_{R,S}(h)$ is higher for some h around u. However, in some cases, it can be more interesting to limit h to $u - 1$ or $u - 2$ as the level of security is not really modified and the computation cost can be decreased. For example, if $S = 160$ and $R = 880$, we obtain $u = 5$ and $\mathcal{NB}_{880,160}(5) = 1$! Thus limiting h to 4 decreases the computation cost downto one addition of S bits.

6.4 Numerical application

With this new type of challenges, we can achieve a level of security of around 32 if $R = 1088$ and $S = 160$ (using challenges with an Hamming weight equal to 5 and with a binary size at most equal to 850). This solution can be very efficient in environments using very little power such as low cost chips. Indeed, this solution only increases the regeneration time of the value r.

7 Comparison of the Answer Computation Costs

In this section we compare the three new solutions presented in this article with the existing efficient schemes: the GPS and the Stern-Stern schemes. We recall that these two schemes respectively lead on average to an overall number of $kS/2$ and $kS/3$ bit additions.

In the following table, S is the private key size and HW its Hamming weight.

	Existing solutions		Our solutions		
Level of security	Basic GPS S=160 HW unknown	Stern-Stern S=160 HW unknown	first solution S=160 HW around 64	second solution S=1469 HW=29	third solution S=160 HW unknown
16	1280	853	816	464	480 (380 3)
32	2560	1706	1168	928	800 (850 5)
64	5120	3413	1872		1600 (1800 10)
80	6400	4266	2224		1920 (2270 12)

Fig. 4. Number of bit additions in answer computation

To compare with the solution in which the private key is the product of low Hamming weight sub-keys, we use the numerical application given in section 4: s_2 a 142-bit number with 16 random bits and s_1 a 19-bit number with 5 random bits. The computation cost is equal to $22k + 464$.

For the second solution with full size low Hamming weight private keys, we also use the given example where s is a 1469-bit number with a Hamming weight equal to 29 (which can be used for level of security until 32).

For the last solution, which makes use of large and low Hamming weight challenges, three figures are given: the first one corresponds to the computation cost of the answer and the two last ones into brackets correspond respectively to the challenge binary size needed to obtain the target level of security and the smallest number of disjoint shifts of the private key that can be added to obtain the wanted level.

8 Conclusion

We have presented three new solutions to improve the implementation of the GPS scheme in low cost chips. Two of them use specific private keys the construction of which seems to resist to the current state of the art. The third solution lets the private key unchanged and only implies the use of particular challenges. The latter solution is as secure as the standard GPS scheme and is also the one which best improves the computational cost of the answer. The consequence of this solution is a very efficient implementation in low cost chips, so that one can be public-key authenticated by doing one on-line addition only!

References

1. H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138. Springer-Verlag, 1993.
2. P. de Rooij. On Schnorr's Preprocessing for Digital Signature Schemes. *Journal of Cryptology*, 10(1):1–16, 1997.
3. M. Girault. Self-certified public keys. In D. W. Davies, editor, *Advances in Cryptology - Eurocrypt'91*, volume 547, pages 490–497, Berlin, 1991. Springer-Verlag.
4. M. Girault, Poupard, and J. Stern. Some modes of use of the GPS identification scheme. In *3rd Nessie Conference*. Springer-Verlag, November 2002.

5. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *19th Annual ACM Symposium on the Theory of Computing*, pages 210–217, 1987.
6. L.C. Guillou and J.J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Günther, editor, *Advances in Cryptology - Eurocrypt'88*, volume 330, pages 123–128, Berlin, 1988. Springer-Verlag.
7. J. Hoffstein and J.H. Silverman. Random Small Hamming Weight Products with Applications to Cryptography. Technical report, NTRU Cryptosystems.
8. T. Okamoto, M. Tada, and A. Miyaji. An Improved Fast Signature Scheme without on-line Multiplication. In *Financial Crypto 2002*, 2002.
9. J.M. Pollard. Monte Carlo methods for index computations modulo p. *Mathematics of Computation (1978)*, 32:918–924, 1978.
10. G. Poupard and J. Stern. Security Analysis of a Practical "on the fly" Authentication and Signature Generation. In *Advances in Cryptology - Eurocrypt'98*, volume 1403, pages 422–436. Springer-Verlag, 1998.
11. C.P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology - Crypto'89*, volume 435, pages 239–252, Berlin, 1989. Springer-Verlag.
12. J. Stern and J.P. Stern. Cryptanalysis of the OTM signature scheme from FC'02. In *Financial Cryptography 2003*, 2003.
13. D.R. Stinson. Some baby-step giant-step algorithms for the low Hamming weight discrete logarithm problem. *Mathematics of Computation (2002)*, 71:379–391, 2002.
14. T.Okamoto, H.Katsuno, and E.Okamoto. A Fast Signature Scheme based on new on-line Computation. In *Information Security Conference'02*, number 2851, pages 111–121. Springer-Verlag, 2003.

A Full Size Secret Key Storage

We denote by k the challenge binary size, by S the secret key binary size and by t the private key (low) Hamming weight. Moreover, we also consider that the level of security cannot be greater than k. Once the maximum level of security is determined, the private key storage directly depends on k, so that k cannot be changed after the creation of the card.

This (non unique) method consists in using private keys the non-zero bits of which are quite regularly located.

Indeed, assuming $t-1$ divides $S-1$ (otherwise, we can increase S), we can divide the $S-1$ bits (the S^{th} one is equal to 1) of the private key in $t-1$ intervals of $(S-1)/(t-1)$ bits, the position of the least significant bit of each interval corresponding to a multiple of $(S-1)/(t-1)$. In any interval of $(S-1)/(t-1)$, there is one block of k bits used to write the shifted challenge so that there are $(S-1)/(t-1)-k$ free bits in the interval and as a consequence $(S-1)/(t-1)-k+1$ possible positions to write a non-zero bit in the interval. Thus, the private key can be written as:

$$s = 2^{S-1} + \sum_{i=1}^{t-1} 2^{(i-1)\frac{S-1}{t-1}+j_i}, \forall\ 1 \le i \le t-1,\ 0 \le j_i \le \frac{S-1}{t-1} - k$$

Thus, storing the private key can be reduced to the storage of the $t-1$ values j_i of constant size $log_2((S-1)/(t-1)-k+1)+1$; some values can be written with less than $log_2((S-1)/(t-1)-k+1)+1$ bits, but we would rather add zero bits to obtain the above size, so that we only need to store the value j_i without its binary size: storing s requires $(t-1)(log_2((S-1)/(t-1)-k+1)+1)$ bits.

For example, with this method, we can use a 1485 bit private key with an Hamming weight equal to 29 and in environments requiring a level of security at most equal to 32. Storing this private key leads to the storage of 28 numbers equal at most to 21 $(= 2^4 + 2^2 + 2^0)$, so that $28 \times 5 = 140$ bits are necessary to store such private keys.

B The Stern-Stern Scheme

B.1 The scheme

Like GPS, it is based on the discrete logarithm modulo a composite integer problem. The private key is an odd s such that $1 = g^s \mod n$ where g and n are public parameters. There are also four security parameters R, S, k and l which have the same goal and properties than in the basic GPS.

Prover		**Verifier**
choose $r \in [\![0, 2^R[\![$ compute $W = g^r \mod n$		
	$\xrightarrow{W}$	
		choose $c \in [\![0, 2^k[\![$
	$\xleftarrow{c}$	
check $c \in [\![0, 2^k[\![$ compute $y = \mathcal{D}_s(r, c)$		
	$\xrightarrow{y}$	
		check $0 < y < 2^R - 1 + 2^{k+S}$ verify $c = y \mod 2^k$ verify $g^y = W \mod n$

Fig. 5. The Stern-Stern scheme

B.2 The dovetailing operation

Let us now explain how to compute the answer practically. The operation $\mathcal{D}_s(r, c)$, called dovetailing, consists in making the k-bit number c appear as the k least significant bits of r by adding adequate shifts of s.

Notations. For any integer b, we denote by $|b|$ its binary size and by b_j, the $(j+1)^{th}$ bit of b starting from the least significant bit; thus $b = b_{|b|-1} \dots b_1 b_0$.

First method. In a first way, the authors perform the dovetailing operation by only using s_0 (which is equal to 1 since s is odd). The answer is built progressively; the value y is first initialized with the value r. Then, we look at the bit r_0; if it is equal to c_0, then nothing is done, else we add s to r so that the wanted bit appears. We go on with the following bits. For example, if r_i is different from c_i, then we need to add an adequate shift of s, i.e $2^i \times s$, such that s_0 location coincides with the one of r_i. We follow this algorithm until r_k.

With this method, on average, $k/2$ bits over the k least significant bits of r need the addition of s, a S bit number. Thus, we obtain on average an overall number of $kS/2$ bit additions.

Second method. In a second way, the authors suggest the use of a private key the two least significant bits of which are equal to 01. Thus, they transform the value of r not only by adding, but also by subtracting some shifts of the private key. Indeed, when r_i is not equal to c_i, instead of automatically adding a shift of s, they first compute $t = 2r_{i+1} + r_i - 2c_{i+1} - c_i \bmod 4$. Depending on the value of t, they determine if it is better to add ($t = 3$) or to subtract ($t = 1$) a shift of the private key in order to obtain the wished value for r_{i+1} at the same time.

With this second method, on average, one bit over three requires one addition. The number of bit additions falls down from $k/2$ to $k/3$ so that, finally we obtain $kS/3$ bit additions.