LOW-SIZE COUPONS FOR LOW-COST IC CARDS

Marc Girault
France Télécom R&D
42 rue des Coutures, B.P. 6243
14066 Caen CEDEX 4, France
marc.girault@francetelecom.fr

Abstract

We address the challenging issue of authenticating a (very) low-cost IC card in a (very) short period of time with public key techniques. Such a need is growing, e.g. in public transport area, where the requirements are very tight, due to a contactless interface and a very short transaction time. Zero-knowledge discrete-logarithm-based authentication schemes with "use and throw coupons" (i.e. commitments which are precomputed, stored in the card and used only once) are well suited to this problem, but state of the art provides coupons which are still too large in many applications (namely 85-90 bits, by Girault and Stern [GS94]).

In this paper, we first observe that the [GS94] paradigm allows to derive a better bound than the one above (about 64 bits), but turns out to be too restrictive, as it only considers off-line attackers (those who perform exhaustive trials prior to the authentication process). This leads us to propose a more realistic environment, in which both off-line and on-line attacks can take place. Then we show that the length of the coupons can nonetheless be still significantly decreased (down to 32 bits!), just by assuming that both time and computational power of the enemy are limited during authentication process. Combined with self-certification or elliptic curve techniques, this allows us to store more than 200 coupons in the IC card, if (only) 1 Kbyte of the E^2PROM memory is occupied by the whole cryptographic material.

Keywords: Identification, authentication, low-cost IC cards, zero-knowledge, hash-functions, off-line/on-line, coupons, transport.

1. INTRODUCTION

We address in this paper the issue of authenticating a (very) low-cost IC card in a (very) short period of time with public key techniques

(sometimes called "on the fly" authentication [PS98, Ste00]). These features are usually viewed as conflicting with each other, as the current standard public key algorithm (namely RSA [RSA78]) requires high-cost microprocessor cards (those equipped with a dedicated cryptoprocessor), so that signatures can be computed within one second or a little bit less — which is still too slow in some applications. Moreover, RSA cannot be implemented at all in case additional constraints apply, such as a contactless interface.

Nonetheless, there are many situations in which all these requirements must hold together. As a first example, let us consider the "electronic purse". Since payments are generally local and off-line, the verification device stands in a public place (at a grocer's, in a parking area etc ...) and it is therefore highly desirable that public key techniques be used, so that no secret master key lies in such a place (even if it is strongly protected). We also expect the transaction to be fast, and the card to be cheap. All that can hardly be achieved with the current technology.

Another still more restrictive environment is public transport. An electronic subway ticket or a toll card should be contactless, so that its holder is not required to stop, which implies both a less powerful chip and a much shorter transaction (in the order of 100 milliseconds)! Again, public key techniques should preferably be used and the challenge is high.

Other factorization-based protocols exist (e.g. Guillou-Quisquater [GQ88]), which are much faster than RSA. When optimized in several ways, they can meet the requirements of some applications, and so they form a first type of solutions to the problem we have raised. But they still cannot meet the tighter requirements as e.g. the transport ones. To deal with them, zero-knowledge(-like) schemes based on the discrete logarithm are best-suited. This is because the heavy part of the computation (often called the "commitment" phase) can be done off-line, i.e. before the identification actually takes place. What remains to be computed on-line is very few: e.g. in Girault's variant [Gir91] of Schnorr's scheme ([Sch89, Sch91]), the card only has one small multiplication and one addition to make — just some milliseconds. This is a practical illustration of the concept of "off-line/on-line signature" from Even, Goldreich and Micali [EGM89]. The drawback is the necessity to store all the precomputed commitments, also called "coupons" in some literature [MN94]. Of course, the shorter they are, the more we can store.

This paper is more specifically about how to make the coupons as short as possible. We show that we can make them at least twice as short as the current state of the art allows (which means: four times as short as done in a current practice), simply by taking advantage of the facts that authentication time is limited and that the enemy computational power

during the authentication process is also limited. More precisely, it turns out that we can use 50-bit coupons in most cases and even 32-bit ones in specific cases.

This result is of high practical significance, in that it allows to store up to 256 commitments in a 1Kbyte memory, instead of 90 with the current state of the art. Combined with self-certification techniques or implemented in their "elliptic curve" version, this leads to schemes allowing to store the whole key material and up to 225 coupons in a 1Kbyte memory only. Since many cheap smart cards have (or will soon have) a 8 or 16 Kbyte memory, the occupancy due to cryptography appears to be very reasonable, and the overall scheme as somewhat "optimal".

2. CRYPTOGRAPHIC STATE OF THE ART

>From a cryptographic point of view, a "coupon" is a commitment, i.e. the result of applying a one-way function to some parameter (usually a random number) or, more generally, a hash-value of such a result (and possibly of data to be authenticated). The commitment step is the first step of most zero-knowledge identification protocols.

The (bit-)length of cryptographic hash-values in identification schemes was for the first time investigated by Girault and Stern at Crypto'94 [GS94]. Beforehand, it was commonly believed that one-way hash-functions were convenient in this context. In [GS94], the authors first showed that collision-resistance (a stronger property than one-wayness) was required, by cryptanalysing various well-known identification schemes used in conjunction with one-way hash-functions. Then they proved that collision-resistance was sufficient to ensure soundness property of these schemes. Finally, they showed how to relax the collision-resistance requirement in the case of arithmetic-based schemes, just by very slightly increasing the length of the challenges sent by the verifier.

On the whole, this paper was somewhat paradoxical in that the first two results stated that a hash-value length of 80 bits (if 2^{80} computations are deemed to be computationally infeasible — but not less) was greatly insufficient, while the last result almost stated the opposite (namely that about 85-90 bits are enough). Still today, this "bound" (even if not presented as such) is the best one publicly known, and is often referred to (e.g. [PS98, Poi00, PS99]).

More precisely, [GS94] claims the following. Let h be a r-collision pseudo-random hash-function (i.e. a hash-function for which it is computationally infeasible to find r pairwise distinct inputs which hash to the same output). Assume that such a function is used to compute commitments in a zero-knowledge discrete-logarithm-based scheme (such as

Schnorr's scheme). Let k be the bit-length of the challenge sent by the verifier. Define the level of security of the scheme as the integer l (expressed in "bits") such that the probability of success of a masquerade is equal to 2^{-l}. Then this level is equal to $k - \log_2(r-1)$, i.e. the (masquerade) success probability is equal to $(r-1)2^{-k}$.

For example, if $r = 9$ (which is achieved nowadays by choosing hash-values of length $t = 85$ or 90 bits, as shown in the same paper through an analysis related to the birthday paradox), the success probability is 2^{-k+3}. If we want $l = 32$, then we have to choose $k = 35$, i.e. only three bits more than if we used "normal" (2-)collision-resistant hash-functions, which output hash-values of typically 160 bits.

3. THREE OBSERVATIONS

We now make three observations about the above stated "result" (or "theorem").

The first observation is that the authors did not draw all the advantages from their own theorem, since the latter remains true even when applied to much large values of r (and consequently smaller values of t). As an example, let us choose: $r = 2^{16}$, $t = 64$ and $k = 48$. Even for so large values of r, the definition of r-collision-resistance still holds: only it has no longer to do with the birthday paradox. And, although the resulting hash-function is no longer one-way, we can still claim that it is r-collision-resistant, since it would require about 2^{80} trials to get 2^{16} distinct inputs which hash to the same 64-bit output. As a consequence, the success probability is approximatively equal to $2^{16-48} = 2^{-32}$ and the security level to 32 bits.

Our second observation is that this theorem becomes wrong if applied to excessively large values of r. As an illustrative example, let us choose: $r = 2^{72}$, $t = 8$ (8-bit hash-values!), and $k = 104$. Reasoning as above, the level of security should still be equal to $104 - 72 = 32$ bits. But it is more than obvious than this level is actually ... null! Indeed, with such a small value of t, a successful exhaustive search can be performed after the value of the challenge is known to the fake prover, since there will be only $2^8 = 256$ trials to make in average, which is feasible by almost every computing capability within a very short time. We will refer to such an attack as an on-line attack, as opposed to an off-line attack, in which the enemy makes all his computations prior to the authentication process.

The reason why the theorem is wrong for large values of r can be stated this way: this theorem implicitly considers off-line attacks only. It becomes almost explicit in the proof, since the set of possible challenges

has (from the attacker's viewpoint) the uniform distribution. This is obviously false for an attacker who knows the value of the challenge.

Now, our final observation is that we can extend the area in which the theorem is true by taking into account the limited resources (essentially time and computational power) the enemy has during the authentication process. For example, let us choose: $r = 2^{32}$, $t = 48$ and $k = 64$. Since it would still require about 2^{80} trials to get 2^{32} distinct inputs which hash to the same 48-bit output, the off-line attack remains infeasible. But, depending on the number of computations the attacker is able to perform once he received the challenge, the level of security may or may not reach 32 bits. For example, in an environment where the attacker can be server-aided and is not required to reply immediately to the challenge, he may perform (let us say) 2^{32} trials and therefore have one chance over 2^{16} (much larger than one over 2^{32}) to masquerade successfully. On the contrary, if the environment is constraining and the enemy cannot make more than 2^{16} computations, then the level of security is still equal to about the desirable one, i.e. 32 bits.

We now make more precise the latter observation by applying it to a specific zero-knowledge scheme.

4. APPLICATION TO THE GPS SCHEME

As an example, we apply the new technique to the so-called GPS variant of the Schnorr scheme (proposed in 1991 by Girault [Gir91] and later proven secure by Poupard and Stern [PS98]), in which the computation to be performed at the time of authentication is as fast as possible: $y = R + sc$ (without any modular reduction). We directly define its version with hash-values:

Let h be a (pseudo-random) t-bit hash-function, n be a large (say 1024-bit) composite number, g be an integer of very large order modulo n, s be a (say 160-bit) integer and $v = g^{-s} (\mathrm{mod} n)$. Alice's public key is v and her secret key is s. In order to prove her identity to Bob, Alice's card picks R (say 256 bits) at random, computes the "commitment": $x = h(g^R (\mathrm{mod} n))$ and sends x to Bob. Bob sends a k-bit "challenge" c to Alice. Alice's card computes the "response": $y = R + sc$ and sends it to Bob. Bob checks that: $h(g^y v^c (\mathrm{mod} n)) = x$.

In the "off-line/on-line" version of this protocol, the commitment is precomputed (by a trusted authority or possibly by the card itself) and stored in the card — it is now called a "coupon" [MN94]. In this way, only has the card to compute y at the time of authentication. Note also that this version spares Alice to store n in her card, if it is a "system parameter" (i.e. a parameter generated by a trusted authority and shared by

all the users), since she no longer has to make any computation modulo n.

In practice, not only Alice or her card must be authenticated, but also some specific data (typically the decrease of a counter or of a balance, in a payment transaction). This is easy to achieve by e.g. including these data in the input parameters of the hash-function h. We omit it here, for simplicity of the description.

What is the security level of this protocol ? To answer this question, we need the knowledge of two bounds. The first one is the number 2^M of "operations" deemed to be computationally infeasible (in any environment). Nowadays, a current choice is: $M = 80$. The second one is the number 2^m of "operations" deemed to be computationally infeasible during the authentication process by the (possibly fake) authenticated device. Clearly, the latter bound is very context-dependent. If Bob checks that the response-time is less than one second, and if the environment makes unrealistic that the enemy's device be equipped with high computational resources (let us say: at most a single PC), then $m = 16$ is a very plausible value.

More generally, since there are about 2^{25} seconds in one year, and even 1 million computers are unable to make 2^M operations in one year, we can choose $m \leq M - (25 + 20) = M - 45 = 35$, under the above assumptions.

When no hash-function is used, the enemy has essentially one only strategy (let us call it the "basic strategy"): "guess" c, pick y at random, compute $x = h(g^y v^c (\mathrm{mod} n))$ and send x to Bob. This will succeed if and only if the guess is right. But when a (non collision-resistant) hash-function is used, he has two additional strategies, the "off-line" one and the "on-line" one.

The off-line strategy consists to collect as many triplets $T_i = (c, y, x)$ as possible (computed in the same way as in the basic strategy), and choose the value of x for which $n_x = \sharp\{c : \exists i, \exists y, T_i = (c, y, x)\}$ is equal to its maximum $n_{\max}$ (where $\sharp E$ denotes the cardinality of set E). Since the enemy can make at most 2^M computations, the average value $\bar{n}$ of n_x is less than 2^{M-t} and, in case 2^M is much larger than 2^t, probability theory shows that each n_x is close to $\bar{n}$, and therefore $n_{\max} < 2\bar{n} < 2 \cdot 2^{M-t} = 2^{M-t+1}$. At this stage, the enemy is able to answer less than 2^{M-t+1} possible challenges, and his probability of success is less than $2^{M-t+1-k}$.

The on-line strategy starts when the attacker has received the challenge c. If he is unlucky, i.e. if $\{\forall i, \forall x, \forall y, (c, y, x) \neq T_i\}$, he still can perform new exhaustive trials, but restricted to the value of c he now knows. To be more explicit, he picks y at random and hopes that

$h(g^y v^c (\mathrm{mod} n)) = x$, where c is equal to the challenge sent by the verifier. Since he can make at most 2^m computations in the limited time and with the limited resources he has, the probability of success will be less than 2^{m-t}.

As a result, the probability of success of the overall strategy will be less than $2^{M-t+1-k} + 2^{m-t} \leq 2 \cdot 2^{\max(M-t+1-k, m-t)} = 2^{\max(M+1-k, m)-t+1}$ and the level of security l greater than $\min(k-1-M, -m) + t - 1$. In order to maximize l, we choose: $k = M + 1 - m$, which leads to:

$$l \geq t - m - 1.$$

With $M = 80$ and $m = 16$, this gives: $k = 65$ and $l \geq t - 17$. If we want a security level of 32 bits, we can choose: $t = 49$ or 50, but if $l = 15$ is enough, then t can be chosen as small as 32 bits! This analysis confirms the results which were claimed above.

5. SOME REMARKS

Even though the new approach was above applied to GPS scheme, any discrete-logarithm-based scheme with a fast response operation, such as Schnorr's one, can a priori be used. This is because the enemy strategies described above only have to do with the length of the hash-values and not with the underlying authentication scheme.

Normally, the parameter R should also be stored in the device. This is avoided by computing it with a pseudo-random generator implemented in the card, so that it is recomputed instead of being stored. The card only has to store the current state of the pseudo-random number generator, which may be as short as 80 bits.

There is a 4-pass variant of these schemes, the goal of which is to "spread" the randomness generated by the verifier by separating his challenge into two ones, respectively sent in the pass 0 (the new one) and pass 2 (as before). This allows to reduce at the minimum the length of the random parameter involved in Alice's answer and consequently is more attractive in factorization-based schemes, such as Guillou-Quisquater [GQ88] or Ong-Schnorr [OS90], for which the computation time is very dependent on this length.

Another approach for solving the problem of storage was initiated by Schnorr himself in [Sch89] and consists to store a few commitments in full (i.e. without applying a hash-function) and "regenerating" new ones starting from "old" ones ([Sch89, Sch91, Sch99]). But the two first regenerating algorithms have been broken ([Roo91, Roo97]), and the third one is too recent and needs further study. A slightly different approach [BPV98] was proposed at Eurocrypt'98, but was broken at Crypto'99 [NS99].

6. IC CARD IMPLEMENTATION FEATURES

The two main characteristics of the resulting scheme are the following:

1 the card only has to perform the operation: $y = R + sc$, where c is small and R, s, y are reasonably large (around 200 bits). This can be easily implemented in a microprocessor card, and very quickly computed: just some milliseconds. We can even envisage to implement it in a simple IC card (without microprocessor).

2 the memory occupancy is very reasonable, since coupons can be made as short as 50 bits (resp. 32 bits). Let us summarize all the cryptographic parameters to be stored in the card (in addition to its identity Id):

- the secret key s (160 bits)
- the current state of the pseudo-random number generator (80 bits)
- the public key v (1024 bits)
- the certificate of the public key v (1024 bits)
- the $C_{\max}$ coupons ($C_{\max} \times 50$ bits; resp. $C_{\max} \times 32$ bits).

For example, in a 1 Kbyte memory, $C_{\max}$ can be close to 115 (resp. 185). But this can still be increased in at least two ways.

The first way consists to use self-certification techniques [Gir91], allowing to save the 1024-bit certificate, so that $C_{\max}$ is now close to 140 (resp. 215). For example, let S_{CA} be the RSA signature function of a Certification Authority CA. Then the "self-certified public key" can be defined as: $w = S_{CA}(v - Id)$, and it is sufficient to store w instead of the pair (v, certificate), since v can be securely extracted from w, Id and the CA verification public function (see details in [Gir91]). Note that it would work with any signature scheme giving message recovery.

The second way consists to use the elliptic curve version of the GPS scheme (the adaptation being quite straightforward). Now v can be only 320 bits and its certificate about 400 bits, by using the (ordinary) GPS signature scheme. This leads to values of $C_{\max}$ as large as 145 (resp. 225).

Finally, note that this scheme can also generate digital signatures, but in this case coupons shall be at least 160-bit large, since h must be collision-resistant, and $C_{\max}$ is equal to about 45.

7. CONCLUSION

We have shown that authentication schemes using coupons must have a specific treatment, taking into account the possibility of performing fraudulent computations before and during the authentication process.

We have given evidence that very short coupons (50 bits and even 32 bits) may be appropriate, under very realistic assumptions.

So short coupons make (more particularly) discrete-logarithm-based authentication schemes extremely attractive for very fast (or "on the fly") authentication. For example, in the elliptic curve version of GPS authentication scheme, the number of coupons which can be stored when the whole cryptographic material must stand in a 1 Kbyte memory, is in some environments as large as 225.

Acknowledgments

We acknowledge Fabrice Boudot for useful comments on our proposal and valuable help in editing the paper.

References

[BPV98] V. Boyko, M. Peinado and R. Venkatesan. Speeding up Discrete Log and Factoring Based Schemes via Precomputations. In Advances in Cryptology – EUROCRYPT'98, LNCS 1403, Springer-Verlag, pages 221–235, 1998.

[EGM89] S. Even, O. Goldreich and S. Micali. On-line/Off-line Digital Signatures. In Advances in Cryptology – CRYPTO'89, LNCS 435, Springer-Verlag, pages 263–277, 1991.

[Gir91] M. Girault. Self-Certified Public Keys. In Advances in Cryptology – EUROCRYPT'91, LNCS 547, Springer-Verlag, pages 490–497, 1991.

[GS94] M. Girault and J. Stern. On the Length of Cryptographic Hash-Values Used in Identification Schemes. In Advances in Cryptology – CRYPTO'94, LNCS 839, Springer-Verlag, pages 202–215, 1994.

[GQ88] L.C. Guillou and J.J. Quisquater. A Practical Zero-Knowledge Protocol Fitted to Security Microprocessors Minimizing both Transmission and Memory. In Advances in Cryptology – EUROCRYPT'88, LNCS 330, Springer-Verlag, pages 123–128, 1988.

[MN94] D. M'Raihi and D. Naccache. Couponing Scheme Reduces Computational Power Requirements. In Proceedings of

Cardtech'94, pages 99–104, 1994.

[NS99] P. Nguyen and J. Stern. The Hardness of the Hidden Subset Sum Problem and its Cryptographic Applications. In Advances in Cryptology – CRYPTO'99, LNCS 1666, Springer-Verlag, pages 31–46, 1999.

[OS90] H. Ong and C.P. Schnorr. Fast Signature Generation with a Fiat-Shamir-like Scheme. In Advances in Cryptology – EUROCRYPT'90, LNCS 473, Springer-Verlag, pages 432–440, 1991.

[Poi00] D. Pointcheval. The Composite Discrete Logarithm and Secure Authentication. In Proceedings of the 2000 International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000), LNCS 1751, Springer-Verlag, pages 113–128, 2000.

[PS98] G. Poupard and J. Stern. A practical and Provably Secure Design for "On the Fly" Authentication and Signature Generation. In Advances in Cryptology – EUROCRYPT'98, LNCS 1403, Springer-Verlag, pages 422–436, 1998.

[PS99] G. Poupard and J. Stern. On the Fly Signatures Based on Factoring. In Proceedings of the 6th ACM Conference on Computer and Communications Security, pages 37–45, ACM Press, 1999.

[RSA78] R.L. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signature Schemes. In CACM, Vol. 21, n° 2, pages 120–126, Feb. 1978.

[Roo91] P.J.N. de Rooij. On the Security of the Schnorr Scheme Using Preprocessing. In Advances in Cryptology – EUROCRYPT'91, LNCS 547, Springer-Verlag, pages 71–80, 1991.

[Roo97] P.J.N. de Rooij. On Schnorr's Preprocessing for Signatures Schemes. In Journal of Cryptology, Vol. 10, n° 1, pages 1–16, 1997.

[Sch89] C.P. Schnorr. Efficient Identification and Signature by Smart Cards. In Advances in Cryptology – CRYPTO'89, LNCS 435, Springer-Verlag, pages 239–251, 1990.

[Sch91] C.P. Schnorr. Efficient Signature Generation by Smart Cards. In Journal of Cryptology, Vol. 4, n° 3, pages 161–174, 1991.

[Sch99] C.P. Schnorr. Fast Precomputation for Discrete Logarithm Cryptosystems. In CRYPTO'99, rump session, 1999.

[Ste00] J. Stern. On the Fly Signatures. In RSA Conference 2000 Proceedings, 2000.