Are Crypto-Accelerators Really Inevitable?

20 bit zero-knowledge in less than a second on simple 8-bit microcontrollers

[Published in L.C. Guillou and J.-J. Quisquater, Eds., *Advances in Cryptology – EUROCRYPT '95*, vol. 921 of *Lecture Notes in Computer Science*, pp. 404–409, Springer-Verlag, 1995.]

David Naccache[1], David M'Raïhi[1], William Wolfowicz[2], and Adina di Porto[2]

[1] Gemplus Card International, 1 place de Navarre, F-95208, Sarcelles, France
{100142.3240, 100145.2261}@compuserve.com
[2] Fondazione Ugo Bordoni, via Baldassare Castiglione 59, I-00142, Rome, Italy
cripto@itcaspur.bitnet

Abstract. This paper describes in detail a recent smart-card prototype that performs a 20-bit zero-knowledge identification in less than one second on a simple 8-bit microcontroller without any dedicated crypto-engine aboard.
A curious property of our implementation is its inherent linear complexity: unlike all the other protocols brought to our knowledge, the overall performance of our prover (computation and transmission) is simply proportional to the size of the modulus (and not to its square).
Therefore (as paradoxical as this may seem...) there will always exist a modulus size ℓ above which our software-coded prover will be faster than any general-purpose hardware accelerator.
The choice of a very unusual number representation technique (*particularly* adapted to Fischer-Micali-Rackoff's protocol) combined with a recent modulo delegation scheme, allows to achieve a *complete 20-bit zero-knowledge interaction* in 964 ms (with a 4 MHz clock). The microcontroller (ST16623, the prover), which communicates with a PC via an ISO 7816-3 (115,200 baud) interface, uses only 400 EEPROM bytes for storing its 64-byte keys.
An overhead video-projected demonstration will be done at the end of our talk.

1 Introduction, Context and Basic Bricks

Although crypto-dedicated smart-cards are an industrial reality since several years, the price of these components is still too high for their massive generalization. As a result, the coding of public-key primitives in simple 8-bit microcontrollers is an important commercial issue with a wide gamut of practical applications.

In the past, several software-only implementations were proposed: the first (and probably the best known) is the Fiat-Shamir implementation in the pay-TV system *Videocrypt*. In Eurocrypt'94, Naccache, M'raïhi, Vaudenay and Raphaeli [7] described a DSA variant based on the pre-computation of ready-to-use *signature coupons*. The NIST has an implementation of the DSA on an 8-bit microcontroller and other remarkable developments in this domain were achieved by Quisquater, Chaum and Fiat's company *Algorithmic Research Limited.*

1.1 Fischer-Micali-Rackoff's Protocol

In Eurocrypt'84, Fischer, Micali and Rackoff [4], presented a factoring-based zero-knowledge protocol for proving the knowledge of a secret s (which modular square v is published by the prover).

In itself, this protocol (actually a Fiat-Shamir [3] with $k = 1$) is very simple:

1. The prover picks a random r, computes and sends to the verifier $x = r^2 \bmod n$.
2. The verifier replies with a random challenge bit b.
3. – If $b = 0$, the prover replies with $y = r$
   – If $b = 1$, the prover replies with $y = rs \bmod n$
4. and the verifier makes sure that $y^2 \equiv xv^b \bmod n$

The security of this *perfect zero-knowledge* protocol is formally established under the sole assumption that factoring n is impossible (we incite the reader to consult [2], [3] and [4] for more details about this method and its numerous generalizations).

1.2 The Brugia-di Porto-Filipponi (BPF) Number Representation System

Denoting by $\{p_i\}$ a set of c co-prime integers, any positive integer $x < g = \prod_{i=1}^{c} p_i$ can be uniquely represented by the list $\{x \bmod p_1, x \bmod p_2, \ldots, x \bmod p_c\}$.

This representation [1] has the distinct advantage of allowing to perform a multiplication in linear (instead of **square**) complexity: if x and y are represented by the lists $\{x_1, x_2, \ldots, x_c\}$ and $\{y_1, y_2, \ldots, y_c\}$ then their product $z = xy$ will correspond to:

$$\{z_1 = x_1 y_1 \bmod p_1, z_2 = x_2 y_2 \bmod p_2, \ldots, z_c = x_c y_c \bmod p_c\}$$

Note that addition (or subtraction) is still linear in this notation (replace the elementwise multiplication by additions or subtractions modulo p_i) but modulo reduction is unfortunately far from being easy (the simplest method seems to be a Chinese remaindering followed by a conventional division). However, a particular property of Fischer-Micali-Rackoff's protocol allows the prover to skip the modular reduction as will be seen later.

Note that if the co-primes are very different from the maximum capacity of the machine words (for instance the first c primes), a considerable redundancy is introduced in each number, a more subtle coding allows to limit this redundancy to it's strict minimum (only 2 bytes are " lost " in each 128-byte value represented on 130 bytes). By choosing $p_1 = 64811$ and $p_2 = 65521$ (the biggest possible two-byte prime), most of the most (and this is not a typographic error) significant bits of p_i are ones.[1]

After a BPF list-multiplication (scalar product), the terms of the resulting list are reduced modulo p_i with Montgomery's algorithm [5] (this operation, which consists in reducing each 4-byte coordinate modulo a 2-byte p_i requires eight byte-by-byte multiplications per co-ordinate and does not affect the overall time linearity).

In our particular implementation, the Montgomery parasite factors ($2^{-16} \bmod p_i$) are not eliminated by the card as this operation can be trivially subcontracted to the verifier.

1.3 Randomized Modular Multiplication

In the European patent application EP 91402958.2, Naccache [6] describes how to delegate the modular reduction of the product of two integers to a powerful verifier. In this procedure (the exact parameter sizes and a complete security proof can be found in Shamir's Eurocrypt'94 paper [8]), the sender simply adds to the product a random multiple of the modulus which is eliminated by the receiver. In other words, instead of sending $z = xy \bmod n$, the prover sends $z' = xy + rn$ (where r's role is to mask the non-reduced product xy)[2] and the receiver computes $z = z' \bmod n$.

This technique is applied by our prover in the following way: after the scalar product, n (pre-recorded in the card's EEPROM in BPF format) is multiplied by a random r and added on-the-fly (as there is no carry propagation in BPF format) to the product (r^2 or sr) which is sent to the verifier.[3]

2 The Protocol

Given these building bricks, implementation is straightforward:

1. The card picks a random r, computes and sends to the PC $x = r^2 + r'n$. r' is an on-the-fly randomizer and x is represented in BPF format.

[1] with this setting, $g \cong 8.1 \times 10^{312}$ is the hexadecimal number:
`b0306dfa10d2bac63339d5fe274fc9ea61d4938dd4c706ea747307fc4ef1465ea49214e`
`30352470531fde44942461730afeca91c365bc9d867be7e06a46ceeb01ef910a4716759`
`2b6b3c8b837f690cc0affcb706ac22de64bc8d3f78fc90a3505d10ea547c63e86983f98`
`68d78084b5533044d865c1cd6f40053396ec2f7783f4d61.`

[2] r should normally be bigger than n by about ten bytes.

[3] The generation of a pseudo-random r, both in BPF and such that: $\lceil \sqrt{g} \rceil \leq r \leq 2^{80} + \lceil \sqrt{g} \rceil$ is done, in $O(\log(n))$, by a proprietary algorithm.

2. The PC:
   (a) eliminates the Montgomery constants in each coordinate of x
   (b) re-codes x in the conventional format (Chinese remaindering)
   (c) replies with the bit b.
3. – If $b = 0$, the card replies with $y = r$
   – If $b = 1$, the card replies with $y = rs + r''n$ (r'' is an on-the-fly randomizer).
4. The PC:
   (a) eliminates the Montgomery constants in each coordinate of y
   (b) re-codes y as a conventional number and checks that $y^2 = xv^b \bmod n$

Note that the verifier can easily check the prover's answer in linear time when $b = 0$ (the prover can then reveal r' which would allow to the verifier to check the responsive in BPF). The linear-time verification of the prover's answer if $b = 1$ is still an open problem (a positive answer will mean that factoring-based zero-knowledge identification and verification are both feasible in linear time).

Although very low, the complexity attained by our prover is not a provable minimum: it is tempting to imagine that a sub-linear complexity would simply mean that some of the modulus bits are not read by the prover but (surprisingly) such may be the case if the modulus is voluntarily chosen to be redundant (for instance, n can be generated to be compression-suitable). Also, and as paradoxical as this may seem, there exists a modulus size ℓ above which our software-coded verifier will be faster than any general-purpose hardware accelerator.

3 Implementation and Performances

The communication amount requested by our prototype is 285 bytes per round (5700 for jumping over " mystic " 1/1000,000 security level barrier). When transmitted according to the ISO 7816-3 standard at 115,200 bauds this transmission takes 0.6 seconds).

An EEPROM option allows to halve the communication by allocating 128 additional bytes and using 3 different secrets (without multiplying them by each other as done in the Fiat-Shamir: after the commitment phase, the verifier simply " points " the secret key he wants to be used in the round).

The code size (approximately 900 bytes), breaks down as follows:

ROM constants:

```
compressed prime table  102 bytes
Montgomery constants    256 bytes
```

ROM code:

```
communication routines          200 bytes
BPF multiplication              120 bytes
Montgomery 4-byte reduction     30 bytes
randomized multiplication core  200 bytes
```

EEPROM data

n	130 bytes
s	130 bytes
card ID	64 bytes
other ‘‘ bookkeeping ’’ data	64 bytes

The following table illustrates four of the possible trade-offs of our mask:

number of secrets	*transmission*	*protocol time*
1	5700 bytes	964 ms
3	2850 bytes	658 ms
7	1420 bytes	432 ms
15	710 bytes	314 ms

Table 1. Possible trade-offs for a security level of 2^{-20} and 4 MHz clock

4 Further Extensions: Two-Way Authentication

As explained in [6], the randomized multiplication can be applied to Rabin's scheme as well.

This can be applied to implement a very quick two-way authentication (not implemented in our prototype) where the card authenticates the PC as well:

1. The card picks a random r, and sends $x = r^2 + r'n$ (in BPF) and $y = CRC(r)$.
2. The PC:
   - remainders x, reduces it modulo n and computes its roots modulo n
   - discards the roots which CRCs do not correspond to y
   - encodes the remaining root w in BPF and sends it to the card
3. The card compares w and r.

$\Rightarrow$ Different ns should be used for sections 2 and 4 and w should not be randomized before being sent to the card...

5 Acknowledgments

The first author would like to acknowledge Jacques Stern's suggestions and improvements regarding the randomized multiplication.

6 Note

This paper does not represent a complete product description, nor does it reflect or represent, in any manner, Gemplus' intention to mass-produce cards using

the features herein described. Availability of such final products may depend on commercial agreements between Gemplus and third parties. Gemplus offers no guarantee that such cards will be free of licensing rights belonging to third parties or related to Gemplus' patented technologies.

References

1. O. Brugia, A. di Porto & P. Filiponi, Un metodo per migliorare l'efficienza degli algoritmi di generazione delle chiavi crittografiche basati sull'impiego di grandi numeri primi, Note Recesioni e Notizie, Ministero Poste e Telecommunicazioni, vol. 33, no. 1-2, 1984, pp. 15-22.
2. U. Feige, A. Fiat & A. Shamir, Zero-knowledge proofs of identity, Proc. 19th. ACM Symp. Theory of Computing, 210-217, (1987) and J. Cryptology, 1 (1988), 77-95.
3. A. Fiat & A. Shamir, How to prove yourself: Practical solutions to identification and signature problems, Proc. of Crypto'86, Lecture notes in computer science 263, 181-187.
4. M. Fischer, S. Micali & C. Rackoff, A secure protocol for oblivious transfer, presented at Eurocrypt'84 but missing in the proceedings.
5. P. Montgomery, Modular multiplication without trial division, Mathematics of computation, vol. 44, 1985, pp. 519-521.
6. D. Naccache, Method, sender apparatus and receiver apparatus for modulo operation, European patent application no. 91402958.2, November 5, 1991.
7. D. Naccache, D. M'raïhi, S. Vaudenay & D. Raphaeli, Can DSA be Improved ?, Proceedings of Eurocrypt'94, to appear.
8. A. Shamir, How to implement public-key schemes with 16,000 bit moduli on a smart-card with 36 bytes of RAM, presented at the rump session of Eurocrypt'94 (05-10-1994 at 20h11).