SCALPS

(Smart Card Applied to Low Payment Systems)

J.-F. Dhem J.-J. Quisquater D. Veithen

Abstract

This paper deals with the conception of a smart card chip dedicated to low payment system for electronic replacement of the coin.

In order to be connected to the existing terminals, it complies with the T=0 smart card protocol described in ISO7816-3.

The chip is based on a modified version of the GQ zero-knowledge scheme. This protocol is well adapted to a variable security level requiring less computations than other authentication protocols.

An original random number generator is proposed to be used with our implementation.

The Montgomery modular multiplication algorithm is used to perform 512 bit multiplication on 8 bit words at a speed of 33 multiplications by second at the standard frequency of 3.57 MHz.

A chip was fabricated. The size is 11.8 mm^2 (halved if the layout is done by hand) in CMOS 1.5 μm with two metal layers. The maximum working frequency is 6 MHz.

A small demonstration board, enabling the connection of the chip to a computer using the RS232 port, was realized.

Introduction

The aim of this research was to realize a system for electronic money as secure as possible where a smart card will be the purse.

At the present time two main types of cards are coexisting: The magnetic ones and the smart card. The former arrived to their limits of possibilities and there are static identification systems very insecure. The latter uses a chip which may content a wide range of dynamic applications.

Nowadays the chips used in smart cards are of three kinds: the first ones are memory chips, like these used in french PTT cards. There are of very low cost but insecure. The second ones are secret key microprocessor-based smart cards. The third are microprocessor based with public-key dedicated coprocessor (like Thomson ST16C853, Siemens SLE44C200 or Philips 83C852). These

are normally safer than the second and requires cheaper terminal and system management but are more expensive.

In the present paper, we have tried to demonstrate the feasibility of a public-key dedicated chip as if the coprocessor of the third type of smart card chips was used alone. This could, for example, be used for dispenser, parking cards,... Other protocols, than the one describe hereafter, are conceivable for this use; the one here is perhaps not the best one. A lot of works in the domain has been done by many researchers (See for instance [Cha92]).

Hereunder the cryptographic algorithm we used is firstly described, thereafter an original number generator, the communication protocol used in smart cards, the Montgomery modular multiplication algorithm, the architecture of our chip and finally the characteristics of the realized chip and the tests to which it was submitted.

Choice of a cryptographic algorithm

Our aim was to conceive a smart card chip able to replace a small amount of money. To reach this goal it must be:

- *impersonal*: not linked to a particular person;
- *exhaustible*: to be impossible to build a card for the life with some information on it;
- *universal*: the possibility to use it everywhere and in all the situations. In this case an on-line transaction is impossible.
- *authenticable*: to certify to the receiver that it is a truly authorized card.

A choice between an on-line system and an off-line system was possible. The first solution requires a link between the terminal and the bank. Since we focus our attention on a low cost card-terminal system, the second solution seems to be the most attractive.

In our device, the card holds a certain amount of money, which will be refered as *prepaid tokens*. In a more general system, the card may hold several kind of tokens differentiated by their use and equivalent in money. Whenever a user wants to refill his card, he goes to a bank and simply indicates the kind and amount of tokens he needs. The bank then transfer this money to the account of the appropriate service provider related to the token (parking, soda dispenser, ...). When the user wants to pay something, he initiate an interaction with the terminal, which simply withdraw the appropriate numbers and kind of token from the card's memory. The terminal does not have to interact with a bank afterwards because the payment already took place before.

In our work, we limited ourselves to a monopolistic case where there is only one service provider gathering all the tokens.

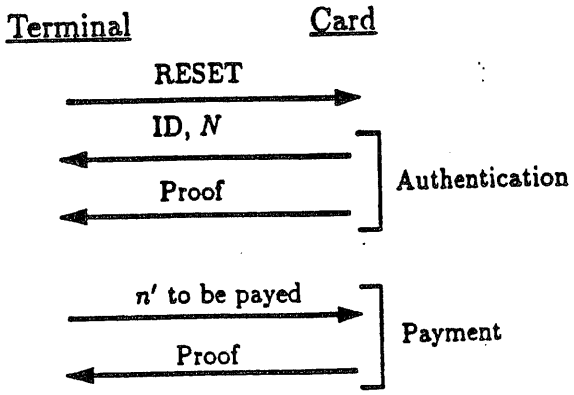


Figure 1: Payment protocol

The ideal protocol (Figure 1) is divided into two phases :

- Authentication;
- Payment.

The payment phase may be repeated a certain amount of time until the completion of the transaction. During the authentication, the card issues its identity and number of tokens remaining and then proves it, by showing for example a secret related to the identity. A step where the terminal indicates the application could be added and hence the type of tokens to use in the transaction. Similarly, in the payment, the terminal sends n' the number of tokens to pay and the card then proves that it effectively did withdraw this number from its memory.

The actions that an opponent may undertake are the following :

- To modify the identity;
- To modify N;
- To modify n';
- To alter a proof;
- To take the place of the terminal.

In this last item, the opponent tries to gather informations about the card and to break the included secret. The practical protocol must thus have the ability to avoid these attacks.

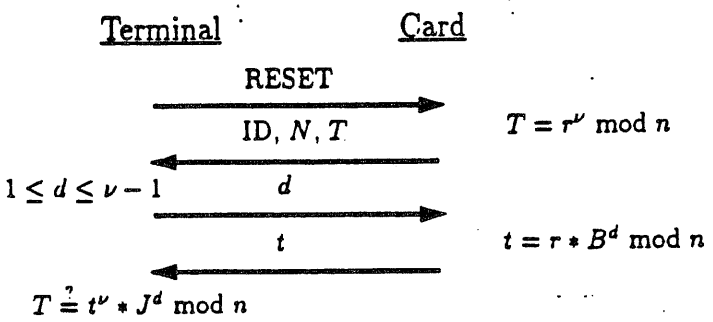
Zero-Knowledge scheme are well adapted in this situation mainly because the computations are reduced when comparing to an RSA signature scheme for example and because there is no leak of the information contained in the card. Among them the Guillou-Quisquater [GQ88], [GQ90] scheme requires more multiplications than the Fiat-Shamir [FS87] scheme but its security level may be modulated.

The problem of exhaustibility may be resolved with a Lamport scheme [Lam81] but it needs on-line communications for updating the system.

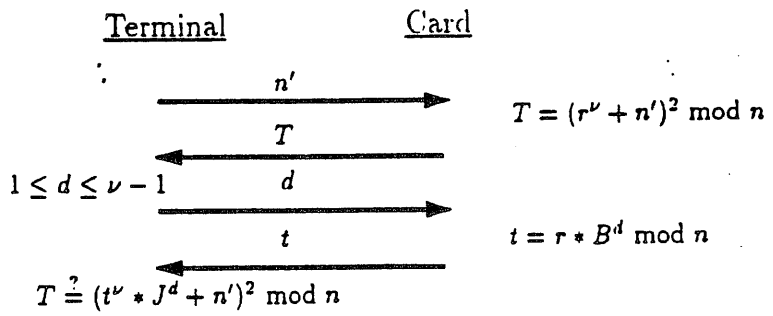
We finally adopted a scheme based on the GQ Zero-Knowledge protocol and two ways to implement the protocol were investigated.

Separated authentication and payment phases

1. Authentication phase



2. Payment phase



Where:

ID: identity of the card;

N: money remaining in the card;

r: 512 bit random number;

n': coin to be take off the card (1 to 8 at one time);

ν: the exponent such that $B^\nu * J = 1 \bmod n$;

B: the 512 bit secret of the card;

J: red(ID) where red() is for example the hash function described in the ISO9796 standard;

$n = p * q$, the 512 bit modulus where p and q are two large primes.

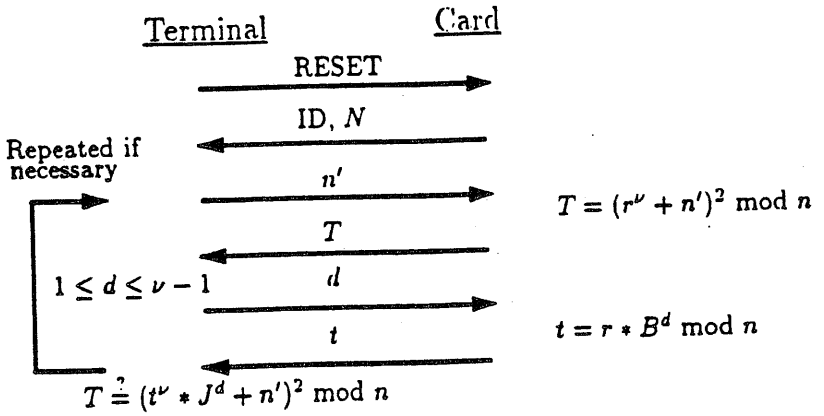
At each iteration, an opponent has exactly one chance over $\nu - 1$ to guess the question d of the terminal so that he can choose every t and then calculate a $T = t^\nu * J^d \bmod n$ that will agree with the verification of the terminal: is T equal to $t^\nu * J^d \bmod n$?

The security level may be modulated, for example, with a high one for the initial phase (authentication) and a lower one for the second. For telephony, the second phase may be repeated each period of time. In our case the shortest period is ≈ 1 s.

The two phases are linked with the ID. If someone wants to authenticate with some mean and to pay with another, this would be impossible because the terminal will do the verification with B linked with the first ID. In this case, a low security level is well adapted. With one chance over ν to deceive the terminal it is also at the second pass so the probability to deceive the terminal is now $1/\nu^2$ and so on.

Mixed authentication and payment

Here it is not possible to modulate the security level as in the first solution but it simplifies the implementation of the circuit as wanted.



The square in the computation of T is there to avoid a separation between the random number r and the number of coin n' to take off the card. If not, an opponent may take less coin off the card than he says to the terminal (he sends $T + n''$ to the terminal).

It is not possible to use something like $T = (r + n')^\nu \bmod n$ because no verification is possible by the terminal.

Some attacks must be taken into account. If someone send $t = 0$ to the terminal and $T = n'^2 \bmod n$ he may do it without any secret. The solution is then very simple; the terminal must refuse a card sending t=0.

Random generator

As shown above, a random number generator is needed to implement the GQ algorithm.

As we already need a modular multiplication function for the GQ algorithm the use of the exponential modulo as random generator as shown by Micali and Schnorr [MS90] was a good way to investigate. We use $r = g^3 \bmod n$ as random generator but the 512 bit seed g [Figure 2] has been modify somewhat.

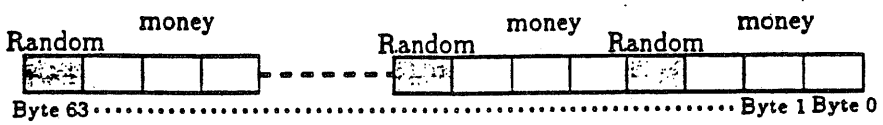


Figure 2: seed of the random generator

Only 16 words on 64 are initially randomly chosen. The other words are at 0XFF at the beginning. Each time, one coin is taken off the card, one bit of the non random part of the seed is put to zero. With that solution the card contains a maximum of 384 coins. The card refuse to work without paying a minimum of one coin so that the seed of the random generator is always different. With this implementation we only need 512 bits of EPROM to memorize the seed and the money. Normally, with a Micali-Schnorr random generator, only a small part of the generated number is random. With a modification of the seed like in Figure 2 and the fact that the random number r is never revealed in itself (only $(r^\nu + n')^2 \bmod n$ and $r * B^d \bmod n$ are transmitted), we may conclude that to generate 512 bit random at a time is sufficiently strong for our use. Some other works must be performed to prove its randomness in other uses.

With an EEPROM, it is possible to imagine to set the value of a true counter in the seed so that the maximum possible amount of money will be 2^{384}. We definitively choose to use an EPROM to minimise the costs of the card and the possibly security problems linked to EEPROM.

A rechargeable card was truly very attractive but it needs an EEPROM and especially a cryptographic protocol to ensure a secure refill in the card. This is too much for our aim of a small not expensive card-terminal system.

Communication protocol

Low cost smart cards use a standard communication protocol named T=0 (part of ISO 7816).
We use this protocol (Figure 3) in its simplest form :

- Answer To Reset (ATR) reduced to TS, T0 (3B, 00);
- Vpp always set active (ACK = INS $\oplus$ 01);
- bit duration set to ETU = $\frac{1}{9600}$ s = 372 clock ticks with a 3,579545 Mhz clock;
- even parity;
- guard time set to 2 ETU;
- time out set to 1 s.

The chip has thus the ability to communicate with a standard smart card reader.

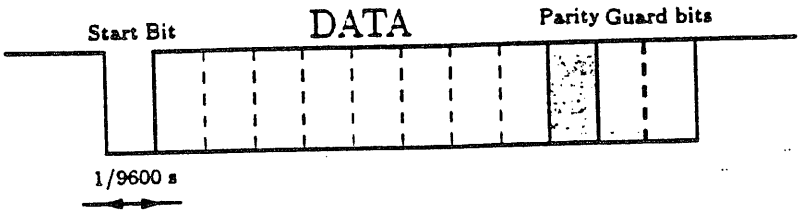


Figure 3: Communication protocol T=0

Montgomery algorithm

Montgomery algorithm [Ara92], [DK91] has been used to perform modular multiplications. To compute $A \times B \bmod n$ where A, B and n are 512 bits numbers, the algorithm is the following one:

$J_0 = n_0^{-1} \bmod n$
$R = 0$
for $i = 0$ **to** 63 {

$R = R + A_i \times B$
$q = R_0 \times J_0$
$R = R + q \times n$

}

To simplify the computations and the implementation, we decided to choose n with its eight last significant bits (n_0) to one so that $J_0 = 1$. To construct such a n, one has to choose a 256 bits number p and then another q such that $q_0 = -p_0^{-1} \bmod 2^{|n_0|}$ then $n = p \times q$ ($n_0 = p_0 \times q_0 \bmod 2^{|n_0|}$). This does not

significantly reduce the security level based upon the factorization of n. Since A is always a number in the EPROM memory and B a number in RAM, only 128 bytes of RAM is needed to perform all the modular multiplications in the protocol.

The complete algorithm is now:

```
R = 0
for i = 0 to 63 {
    for j = 0 to 63 {
        M = 0
        T = A_i
        for t = 0 to 8 {
            M = 2 × M
            if (LSB(T)==1) M = M + B
            T = T ROR 2
        }
        R_j = R_j + M
    }
    q = R_0
    for j = 0 to 63 {
        M = 0
        for t = 0 to 8 {
            M = 2 × M
            if (LSB(q)==1) M = M + n
            q = q ROR 2
        }
        R_j = R_j + M
    }
}
if (R > n) R = R - n
if (R > n) R = R - n
```

The result is $A \times B \times I \bmod n$ where $I = 2^{-512} \bmod n$. There are several ways to take this I into account :

- Multiply every operand by $2^{512} \bmod n$ (with a standard method) before the multiplication and by 1 after the multiplication.
- Precompute $H = (2^{|n|})^2 \bmod n$ and $L = (2^{|n|})^{2^{|n|}} \bmod n$ and multiply A by H and the result R by L.

- The terminal uses the Montgomery algorithm. The secret B must be recomputed as follow : $B' = B \times 2^{512} \bmod n$ with a standard method.
- Compute the number of I introduced and take them into account in the verification.

The last option has been selected because it does not involve any precomputation at the card side. The terminal must verify that :

$$(t^{\nu} \times J^{d} \times (2^{|n|})^{\nu \times (d-1)+1} + n')^2 = T$$

which implies a little computation overhead.

The time complexity for one modular multiplications is $2 \times k^2 \times t$ additions where $k = 64$ and $t = 8$ and yields about 65536 operations. It is possible to perform 33 modular multiplication in about 1 second (security level = 1/66).

Circuit architecture

To compute $A_i \times B_j$ registers A (rotate register), B, ACCU1 and ACCU2 have been used around a ten bit adder (Figure 4). Each operand is affected to a different register so that a load in memory, an addition and shifting a register can take place simultaneously. Register S64 hold the 65th byte of R and DB is a 1 bit register to store the 521th bit of R after the multiplication step of the algorithm. Register D holds the exponents (ν and d).

To reduce the final result, n is simply added in its 2 complementary form using the inverter before register B. The counters C1 ... C7 take control of the loops (i, j, t), the shifting of the registers and the number and duration of the bits transmitted through the I/O line.

The control unit is made of PLA (holding the entire protocol including the Guillou-Quisquater Zero-Knowledge scheme, modular multiplication. I/O and EPROM programming) because of their size in comparison with ROM.

128 × 28	PLA	ROM
Size (mm^2)	0.5	2.3
Consumption (mA)	28.5	1.2

Their advantage is reduced after the placement and routing of the chip because of the bad contact localization.

All the personalisation information is in the 256 byte EPROM: the secret B (512 bits), the identity ID (512 bits maximum) the random seed and the money (512 bits) and finally some work parameters like the security level and T=0 configuration words.

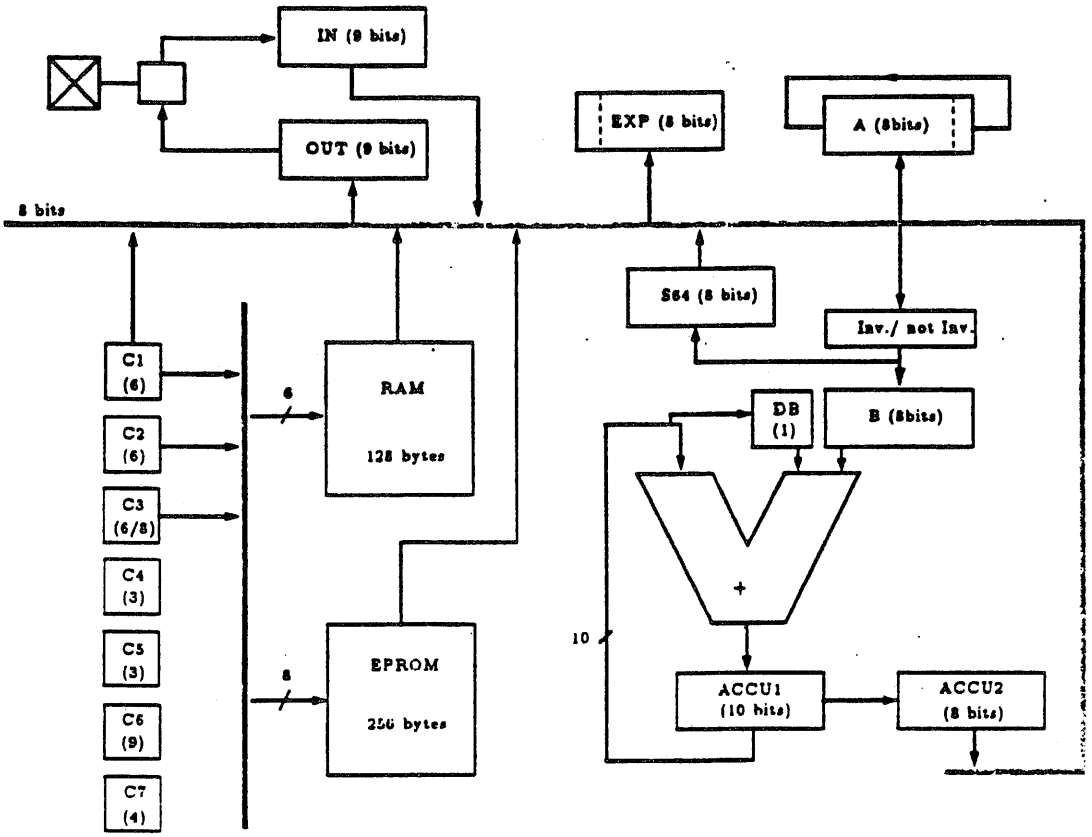


Figure 4: Data path.

Characteristics of the chip

The chip has been designed using the ES2 ecpd15 technology: CMOS 1.5 μm with two metal layers. With that foundry access to EPROM technology was impossible so that we use an external EPROM. For this first test chip we do not have optimised the design a lot: the routing was performed automatically by the CADENCE SOLO2030 design tools. The die size is $3.38 * 3.49\ mm^2$. With an optimisation of the control unit and a routing by hand, we think that we may divide the die size by nearly two. The use of an external EPROM implies a $12V$ power supply with an analog switch to control it. To drive the EPROM, the chip needs a lot of supplementary pins compare to a real smart card one.The pins CLK, GND, VCC, Vpp, I/O and Reset are the same as on the real smart cards. Some other pins were also added to facilitate the tests of the chip like access to internal buses. The chip does not contain the necessary physical or electrical protections. It was not the aim of this work.

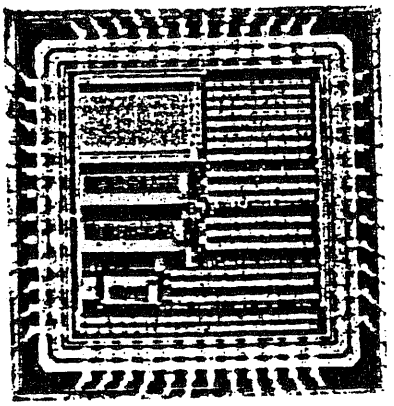


Figure 5: $3.38 * 3.49\ mm^2$ 1.5 μm chip

Tests

The chip was initially designed to work at 3.579545 MHz because of the standard low price quartz frequency for smart card used in many countries. We test it for a variety of chosen data's. Nine of the ten received chips work well until 6 MHz without any problems (the last one was broken). The protocol T=0 was implemented without any timer so that, for example, at 5.34 MHz the communications works automatically at a baud rate of 14400 bps. At a higher frequency than 6 MHz, the chip seems to work correctly except the programming of the EPROM. This is normal because the timing of the programming is also dependent of the clock frequency. This does not implies a security problem because the card detects a mis-function of the EPROM and refuses to work. The power consumption is about 34 mA at 6MHz/5V with the EPROM. The main part of it is due to PLA consumption. A redesign with more ROM than PLA may dramatically reduce the power consumption but increases the size.

A demonstration board composed of a small printed circuit with our chip and the EPROM (it represent the smart card) and a small interface between an RS232 Mac/PC serial port and the card coupler (this interface works exactly like commercial interface product between a PC and a T=0 smart card) were realized. Only a small program in C is necessary to drive the card to simulate the T=0 interaction between the PC and the card. With a C modular multiplication implementation, the PC also does in real time the verification made by a truly terminal.

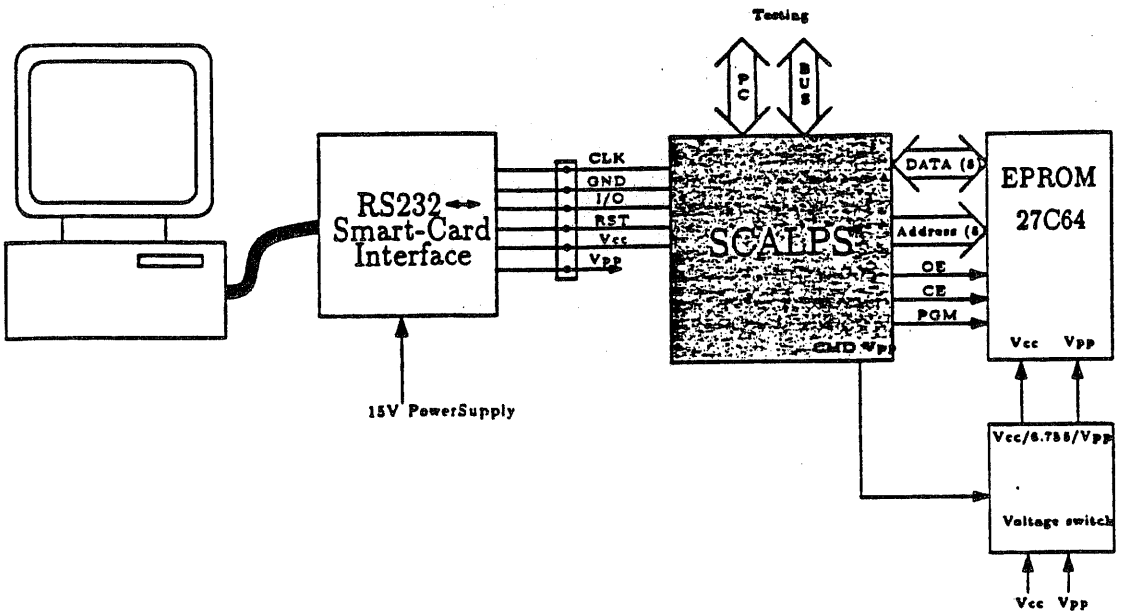


Figure 6: Demonstration circuit

Conclusions

After a comparison between the different crypto algorithm usable in the case of electronic money we got to the conclusion that a zero knowledge GQ protocol is well suited for it. The only disadvantage is that we do not have, as ideally wanted, a truly exhaustable money because the chip has a protected secret who may generate new money if it is uncovered. The primordial off-line characteristics of our chip make it impossible at the present time.

A chip with a really adaptive security level during the personalisation process (not in the foundry) was realized. It is truly more secure than the very used memory cards but more cheaper than the microprocessor based chips because of its smaller complexity and die size.

The interface with existent terminals is the same as today. The terminal needs a low level computation power (nearly the same as in the smart card).

At the moment this chip may only be used in a monopolist system (the card seller must be linked to the dispenser) but this is not very restricting because it is nearly always the case in reality for low payment.

We think that this way is very promising.

References

[Ara92] B. Arazi. Variations on the Montgomery theme. manuscript, 1992.

[Cha92] D. Chaum. Achieving electronic privacy. *Scientific American*, pp. 96–101, August 1992.

[DK91] S.R. Dussé and B.S. Kaliski Jr. A cryptographic library for the Motorola DSP56000. In I. Damgård, editor, - *Proc. Advances in Cryptology - EUROCRYPT '90*, vol. 473 of *Lecture Notes in Computer Sciences*, pp. 230–244, New York, 1991. Springer Verlag.

[FS87] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A.M. Odlyzko, editor, - *Proc. Advances in Cryptology - CRYPTO '86*, vol. 263 of *Lecture Notes in Computer Sciences*, pp. 186–194, New York, 1987. Springer Verlag.

[GQ88] 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, - *Proc. Advances in Cryptology - EUROCRYPT '88*, vol. 330 of *Lecture Notes in Computer Sciences*, pp. 123–128, New York, 1988. Springer Verlag.

[GQ90] L.C. Guillou and J.-J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, - *Proc. Advances in Cryptology - CRYPTO '88*, vol. 403 of *Lecture Notes in Computer Sciences*, pp. 216–231. New York, 1990. Springer Verlag.

[Lam81] L. Lamport. Password authentication with insecure communication. In *Proc. Communications of the ACM*, vol. 24, pp. 770–774. ACM. November 1981.

[MS90] S. Micali and C.P. Schnorr. Efficient, perfect random number generators. In S. Goldwasser, editor, - *Proc. Advances in Cryptology - CRYPTO '88*, vol. 403 of *Lecture Notes in Computer Sciences*, pp. 173–198, New York, 1990. Springer Verlag.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Proc. Communications of the ACM*, vol. 21, pp. 120–126. ACM, February 1978.