Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.N.E. Bos, D. Chaum

SmartCash: a practical electronic payment system

Computer Science/Department of Algorithmics & Architecture Report CS-R9035 August

SmartCash: a Practical Electronic Payment System

Jurjen Bos
David Chaum
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, the Netherlands

SmartCash is an electronic privacy-protecting payment system that is realizable using current smart card technology. It is currently being implemented.
Until now, there were two different electronic payment systems: systems using *private keys* and the ones which use *public keys*. Private key smart card payment systems provide either privacy protection (using one key for the whole system) or security (using one key per card), but never both. Public key payment systems are not feasible, because today's smart cards are not capable of doing such calculations.
SmartCash uses a combination of private and public key methods to achieve both privacy and security. It only requires simple computations (DES) on the smart card, while it uses RSA to provide good privacy protection. A payment in the SmartCash system takes only a few milliseconds, so it can be used in high-speed applications.

Key Words & Phrases: Electronic cash, Smart Card.
1985 AMS Mathematics Subject Classification: 65V05, 68M20;
Computing Reviews code: E34, Public Key Cryptosystems.

Features

The user's smart card can be inserted in a *transponder* (TR), that is used for communication and for doing the computations needed in the protocol. The protocol is designed so that this transponder does all the privacy-relevant parts, while the smart card does the security-relevant parts. The protocol also does not require the user to trust the smart card. The protocol uses very little of the smart card's memory, so that simple smart cards can be used.

The money is carried by *checks*. Checks are numbers of a special structure that allows parts to be revealed while other parts remain secret. Those checks have no intrinsic value of their own; the actual money is represented by a *balance* in the smart card. The smart card increases and decreases this balance during the protocols.

For simplicity, we will talk about DES as block cipher and RSA as public key cipher in this article. Of course, also other systems can be used.

The life of a check

Figure 1 shows an overview of the life cycle of a check.

A check starts its life as a **check commit**. A check commit is the result of a protocol in which the smart card gives the transponder a commit (one-way function value) of the check secret key (see next section). As commit function we use a one-way function based on DES.

After getting the check commit, the transponder has one of two choices:

- Ask the smart card to open the commit, so that the proper functioning of the smart card can be verified. An **opened check** cannot be used anymore.
- Producing a **precheck** that can be used for the next step.

The smart card makes sure that no transponder will be allowed to make both choices for one check. (This requires only one bit of memory on the card.)

The precheck is a blinded check that can be sent to the *money central* for signing. The blinding factor for this check is computed by the smart card and the transponder together. The precheck is constructed by the transponder, but the smart card monitors the computation using the so-called *woop protocol*, that we will describe later on.

This precheck is sent to the money central for signing with the private RSA key of the money central (the third root). The transponder can then divide out the blinding factor to get a signature on the actual check number, the **signed check**.

The signed check is only usable together with a set of secret numbers provided by the smart card. For speed, we let the smart card precompute those numbers. We call this *expansion*, and we call a check in this state an **expanded check**.

The expanded check can be used for a high-speed payment taking no more time than the transmission of about 1K bits of information.

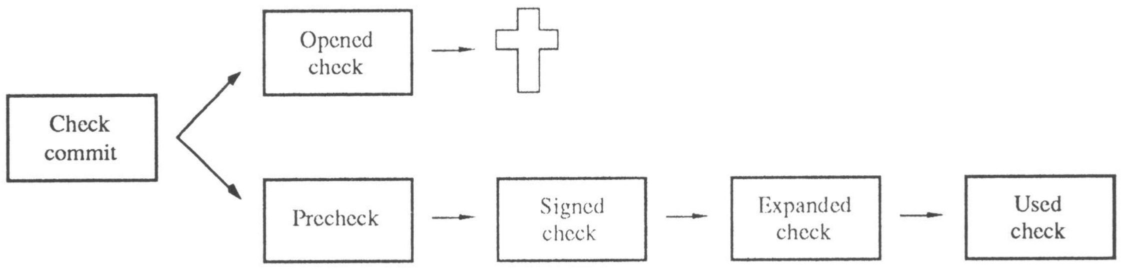


Figure 1: checks from cradle to grave.

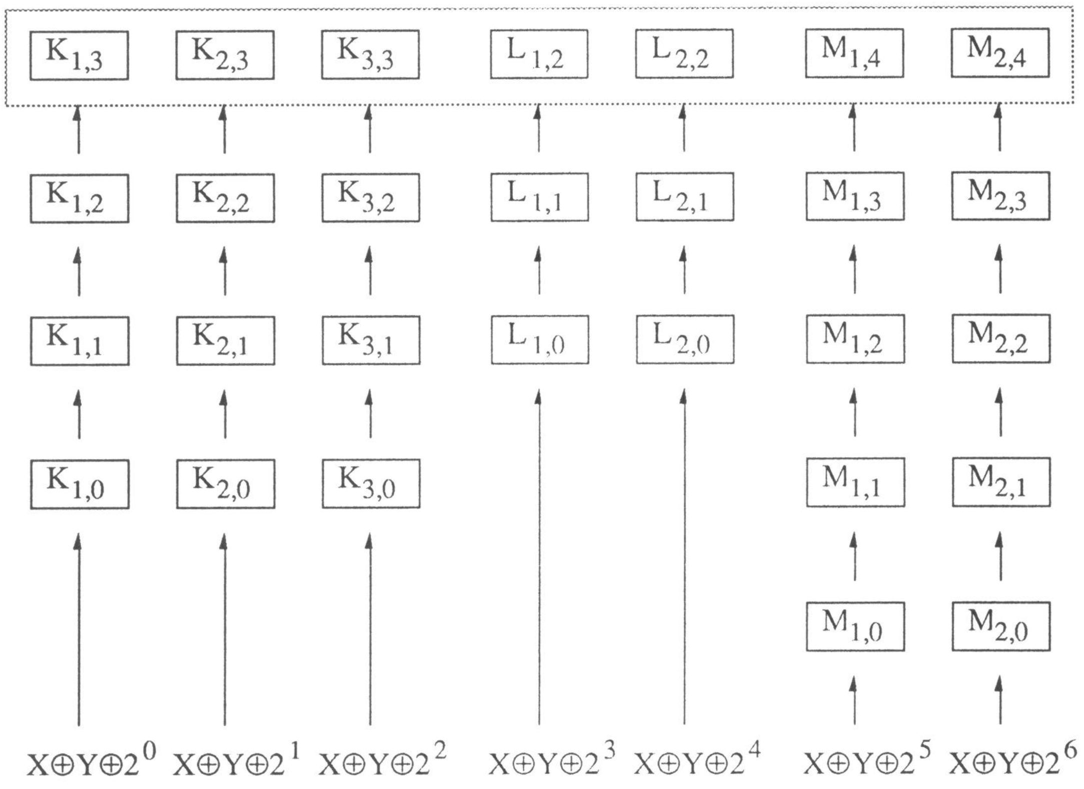
Structure of checks

A check is constructed from a *check key*, which is the exclusive or (⊕) of a key *X* provided by the smart card and a key *Y* provided by the transponder. The key *X* is computed from the smart card secret key, so it can be computed again as needed.

Via this check key matrices *K*, *L*, and *M* are created using a DES based one-way function, denoted in figure 2 as arrows. (Note that the key to each column is the check key except for one bit.) The precheck is a one-way function of the top row elements of the matrices *K*, *L*, and *M*.

During the check committal, the smart card sends the top row of each matrix to the transponder, together with a commit on *X*. The transponder can then compute the entire check if the smart card reveals *X*: this is the opening of a check. If the transponder decides to use this check, it creates together with the smart card a blinding factor for this check. The smart card also gives the transponder a signature *H*, which is constructed using its secret key *S*.

The blinded check can be sent to the money central for signing. The money central returns the sum of *H* with the signed check, so that sending illegal checks to the money central yields nothing to the sender.



$K_{1,3}$ $K_{2,3}$ $K_{3,3}$ $L_{1,2}$ $L_{2,2}$ $M_{1,4}$ $M_{2,4}$
$K_{1,2}$ $K_{2,2}$ $K_{3,2}$ $L_{1,1}$ $L_{2,1}$ $M_{1,3}$ $M_{2,3}$
$K_{1,1}$ $K_{2,1}$ $K_{3,1}$ $L_{1,0}$ $L_{2,0}$ $M_{1,2}$ $M_{2,2}$
$K_{1,0}$ $K_{2,0}$ $K_{3,0}$ $M_{1,1}$ $M_{2,1}$
$M_{1,0}$ $M_{2,0}$
$X \oplus Y \oplus 2^0$ $X \oplus Y \oplus 2^1$ $X \oplus Y \oplus 2^2$ $X \oplus Y \oplus 2^3$ $X \oplus Y \oplus 2^4$ $X \oplus Y \oplus 2^5$ $X \oplus Y \oplus 2^6$

Figure 2: Example of a check structure

Paying with a check

Paying with a check is performed with help of the smart card. The transponder cannot pay alone, because the entries of K, L, and M are needed. First, the payee determines the *payment vector*. This payment vector determines the numbers that the payer has to send. The smart card of the payer reveals the corresponding numbers from the matrix while decreasing the balance. The three parts of the payment vector have the following functions:

- The K part is the *challenge*, a random number that makes the payment vector unpredictable to the payer.
- The L part is the *amount of payment*, an encoded form of the actual amount.
- The M part makes sure that no payment vector can be computed from an earlier one. The value of the M part is computed from the K and L part by a publically known function. The reason for this is explained further on.

The only thing that the payer has to do is sending the repective matrix entries to the payee. Those numbers are made by the smart card during the expansion, and sent to the payee via the transponder.

Tricks used in the protocols

Like all cryptographic protocols, the payment protocol consists of a big amount of tricks, that are combined to yield all the needed features. We will only show the more interesting tricks used in the protocol. In practice, we need several extra tricks to prevent all different kinds of attacks that are possible in the used setting.

Computing numbers together

The protocol uses random numbers that are computed by two parties: the check key, for example, is computed by the smart card and the transponder together.

Figure 3 shows this is done. The protocol consists of three steps: first the smart card sends the commit of its key to the transponder, then the transponder sends its key to the smart card, and finally the smart card sends its actual key, so that the transponder can check the commit. The two parties then perform an agreed combination of the two keys (exclusive or in our case).

$$\begin{tikzcd}
\text{SC} & & & \text{TR} \\
\text{choose } X & {} \arrow[r, "\text{commit}(X)"] & {} & \\
& {} & {} \arrow[l, "Y"'] & \text{choose } Y \\
& {} \arrow[r, "X"] & {} & \text{verify } X
\end{tikzcd}$$

Figure 3: Computing a key $X \oplus Y$ together by two parties.

The "woop" protocol

Probably the most interesting trick that we use is the protocol in which the smart card verifies whether the transponder's computation of the blinded check is performed correctly. The smart card has to verify a computation that it can not perform itself, so it shadows the computation by computing the modulo reduction of all the numbers involved.

This reduction, which is a reduction modulo a secret modulus, is called the *woop function*. For this modulus we chose the smart card's check key X. The smart card can only shadow computations consisting of additions, subtractions and multiplications this way. Because of this, we have to reduce the modulo reduction that is used in the RSA computation to those steps.

$$\begin{tikzcd}
\text{SC} & & & \text{TR} \\
\text{compute } a & {} \arrow[r, "\text{commit}(a)"] & {} & \text{random } b \\
& {} & {} \arrow[l, "{b, n}"'] & \\
\text{compute } a & {} \arrow[r, "a"] & {} & \text{verify commit }(a) \\
& {} & {} \arrow[l, "d = xr^3 \text{ div } n"'] & r = a + b \\
\text{verify } x(a+b)^3 \equiv dn + m & {} & {} \arrow[l, "m = xr^3 \bmod n"'] &
\end{tikzcd}$$

Figure 4: Computing a blinded check under smart card control.

In figure 4 we see the computation of blinding the check x to the blinded check m. The RSA modulus used is n. The blinding factor is $a+b$, where a is computed by the smart card and b is chosen by the transponder (this is explained in the previous trick). The smart card cannot remember the number a, as it is 512 bits, so it computes it twice. The woop of a is computed while a is transmitted to the transponder. All other numbers are reduced as they are received.

The computation of m is verified by the smart card using the equality

$y = n(y \operatorname{div} n) + (y \bmod n)$, where $y = x(a+b)^3$.

This equality is verified modulo the secret woop modulus. All subcomputations are also reduced, so that no intermediate result exceeds the size of the woop modulus. The woop modulus is 64 bits, which makes it very difficult for a cheating transponder to give a wrong answer.

It is hard to compute a good value for the size of the woop modulus (it involves a lot of game theory and combinatorics), but this size is certainly high enough.

Preventing reuse of a check

We will explain this with the check of figure 2. Suppose also that the payee requires the entries of *K* and *L* as given in figure 5, so that the challenge vector is (2,0,1) and the amount vector is (2,1).

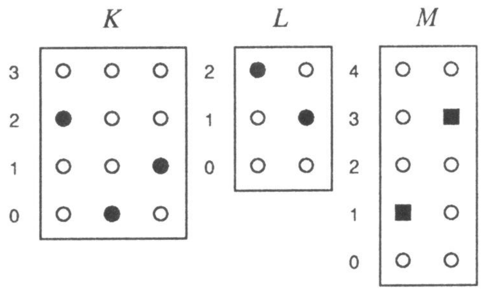


Figure 5.

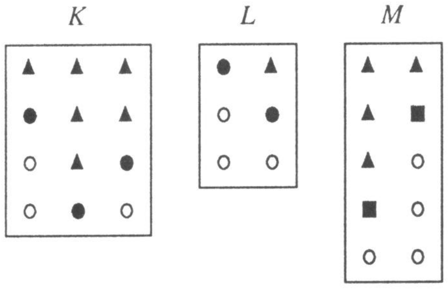


Figure 6.

The construction of the *M*-vector is as follows: the sum of the entries of the challenge and amount vector together is 6, and this number is bounded by 13 (=3·3+2·2). The difference of these two numbers is 7, which is 13 in base 4, so the *M*-vector becomes (1,3).

The smart card gives the transponder the indicated entries of the matrices. Hence the transponder is able to compute all the entries in the matrices *above* each revealed number (indicated in figure 6 with triangles).

This special structure makes it impossible to spend a previously used check again. Suppose that a transponder tries to use the recorded response from a payment transaction for paying, or one of the check that he payed with before. Suppose furthermore that this transponder is so lucky to receive a challenge and amount vector he can answer, using one or more triangles in figure 6 for the *K* and *L* parts. Now the construction of the matrix *M* implies that the number indicated in the *M*-matrix must be lower than that of the old check. Then the *M*-matrix contains at least one entry that is less than the recorded response, so this entry cannot be computed. In other words: if one entry of the matrix is shifted up, another one will have to be shifted down.

Only if both the challenge and amount vector are exactly the same as before, the reuse of a recorded check will be successful. As an extra precaution, the payee requires part of the check (the first 64 bits) to be sent before the challenge is given.

Upgrading the smart card balance

For privacy protection, we invented a new protocol that upgrades the balance of the smart card. It inhibits information transmission from the money central to the smart card. The protocol can be pictured as follows:

$$\begin{tikzcd}
\text{TR} & \text{SC} \\
{} \arrow[r, "{e,\ v}"] & \text{compute } G \text{ from } e,\ v \\
\text{verify } G & {} \arrow[l, "\text{commit}(G)"'] \\
{} \arrow[r, "G"] & \text{verify } G
\end{tikzcd}$$

Figure 7: Upgrading the smart card balance.

In figure 7, *e* is a sequence number (known by the transponder and the money central) and *v* is the upgrade amount. The transponder just got the number *G* from the money central, but doesn't want to just sent this number to the smart card, because there could be privacy-related information in it. Therefore, the smart card shows that it already can compute *G* from *e* and *v* so that the number cannot tell the smart card anything that it didn't know in the first place.

Improvements upon the described protocol

There are several improvements on the protocol. We will describe some of them here. They are already incorporated in the version we are implementing.

Combination of checks

Several checks can be combined to one number. This saves transmission time when the checks have to be transmitted for signing by the money central. The combination is done by multiplying powers of the checks together. For example, checks a, b, c, d, and e can be combined to the number $a^5b^7c^{11}d^{17}e^{19}$.
This whole combination can be sent to the money central for signing, and can be taken apart by the transponder. A complicated version of the woop protocol is necessary to enable the smart card to check the computation.

Use without transponder

Both the payment and the upgrading of card and check can be done without the transponder. The user will loose some of his privacy (because that was what the transponder is for) but it will be very handy in an environment where the transponder and non-transponder systems coexists. The user can then choose himself whether the transponder is used for a certain protocol, having the choice between convenience and more privacy.

Security improvements

The protocol as described here can be disturbed by turning the smart card off during certain points in the protocol. An extra trick makes sure that this will never yield anything to the user. This is necessary because writing in smart card EEPROM is usually rather slow (about 10 ms).

Receipts

The smart card can give the user a receipt of payment by signing a description of the payment with its own secret key (using DES). This receipt can later be used if the payee claims not to have received the money.

Conclusion

We showed it is possible to make a payment system with good security and unconditional privacy with affordable hardware (a prototype of this system has actually been built). However, if the future gives us smart cards that can do RSA, a still more powerful payment system can be developed, yielding a system that needs no tamper proof device for the user at all, giving better philosophical privacy protection.