NetCard — A Practical Electronic Cash System

Ross Anderson, Charalampos Manifavas and Chris Sutherland

Computer Laboratory, Pembroke Street, Cambridge CB2 3QG

1 Introduction

Over the last ten years or so, there have been a number of proposals for electronic cash systems, which let a customer make a payment to a merchant over a computer network by sending messages called 'digital coins' which the merchant can redeem for value (see, e.g. [Cha] [CAFE]). These coins may be managed using a tamper resistant 'electronic wallet': such systems have already been introduced in a number of countries to meet specific local requirements [And1].

Recently, a likely international standard has emerged from VISA and MasterCard; their Secure Electronic Transaction [SET] protocol will facilitate credit card transactions on the Internet. However, an electronic credit card system, such as SET, will suffer from relatively high transaction costs. Not only is the system overlaid on the existing credit card legacy infrastructure, but it will also function in the same commercial framework. Banks worldwide have developed an intricate system of contractual relationships that determine how much money will be paid for processing each transaction, and these costs are geared to transactions in the tens to hundreds of dollars. Thus although SET will be economic for payments in this range, it will be less so for the smaller payments needed for many network services. The digital analogues of stamps and phone tokens — 'micropayments' — will need different treatment, and we discuss some options here.

The application that first motivated us was electronic publishing. When we wish to read an article from a journal that is not in the local libraries, we currently have to buy an annual subscription. This is usually so expensive that we explore alternatives, such as writing to the author for a preprint. It would probably be much better for all concerned if we could just buy the pages of interest. So we set out to provide a mechanism whereby journals could be sold electronically, with a page charge as an alternative to an annual subscription.

Other examples are video-on-demand, other fast network services, computer facilities, road tolls and utility meters [AB]. In all of these applications, a customer makes a series of small payments to the same vendor.

As time passes, more complex applications will emerge. For example, we may wish to rent a video from a shop in Los Angeles to watch in Cambridge. This could involve simultaneous payments to the video store for the content, and to a network service provider to pay for the bandwidth. However the principle is the same; we are still making a series of low value payments to a merchant. We are simply running two such processes at once.

We therefore set out to design a system for repeated small payments. We did not want to force either the customer or the merchant to buy expensive hardware, and we did not want to add significant extra load to the existing legacy banking systems, as computing and network costs typically make up the largest part of a bank's non-interest expenditure. We believed, when the project started in June 1995, that we would have to support the same levels of offline operation as in existing cheque and credit card systems.

2 Concept

In many previously proposed electronic cash systems, the customer purchases a number of electronic coins from a bank and spends them with one or more merchants. The merchants can then redeem the coins with the bank. The coins typically involve some kind of digital signature, which means that at least one modular exponentiation is required to process each coin.

Our key innovation is that, instead of having to do a digital signature each time she spends a coin, the customer can sign a whole stick of coins at the same time. She can then pull the coins off the stick one by one and present them to the merchant, who is able to check that each coin was one of the stick signed by the customer. He only has to perform one signature verification per customer, even when she spends a whole lot of coins.

In the first version of our system, which we shall describe next, the coins are still generated by the bank. However this requirement can be relaxed, and some or all of the coins generated by the customer, which gives additional benefits — for example, she can create coins in any denomination, and indeed in any currency, that she requires. These protocols, and their security properties, will be described in succeeding sections. Finally, we shall discuss the lessons learned about the relationships between local and global trust.

3 Initial Design Assumptions

The first protocol was designed on the assumption that in order to control network and processing costs, small transactions would not be authorised online to the bank. One of the lessons learned from current online payment systems is that maintaining the 99.99% availability needed by retail customers is very expensive. It means not just multiple redundant backup of computer and communications facilities, but also sizing systems for peak traffic — typically 1pm on the Saturday before Christmas. But if stand-in processing is possible, then outages and traffic peaks can be dealt with by placing some traffic offline at a fairly predictable cost in fraudulent and over-limit transactions.

In this scenario, however, there needs to be a mechanism to prevent double spending by customers. The approach adopted in the UEPS system [And1] was to issue each customer with a smartcard based electronic wallet, and in our first protocol we assumed that this would be favoured by the industry.

The problem with smartcards is that one can either have a universal secret that is present at least in merchants' cards, or use public key cryptography. In the former case, criminals might reverse engineer the card [BFLAR] and extract the keys using chip testing techniques [Wie]. Systems such as UEPS control this risk by having centralised reconciliation and fraud detection systems, together with a fall-back processing mode similar to current credit card operations. The effect is that a capable motivated opponent would be better off spending his resources attacking the legacy credit card system.

We set out to follow the second route and use public key crypto. An important reason for this was that the second and third authors are funded by the NetCard project, a DTI/EPSRC initiative to develop payment mechanisms for high speed networks, and one of our industrial partners in the project is developing a public-key capable smartcard. However we still want the smartcard to do as little public key crypto as possible — at most one signature for each series of transactions between a customer and a merchant.

4 First Protocol

In our first protocol, the customer first chooses a pseudonym P and generates a pair of keys with KP^{-1} being her private key and KP her public key. The bank then creates a certificate with its name B, an expiry date e, the customer's credit limit \$, her pseudonym P and her public key KP, all signed using the bank's key KB^{-1}. Using the symbolism of [AN]:

$$CP = \{B, e, \$, P, KP\}_{KB^{-1}} \tag{1}$$

For simplicity's sake we will first consider that the digital coins are all for the same amount. They are 64 bit strings constructed by encrypting a serial number and some redundancy under a secret key selected by the bank. Different keys are chosen for each customer pseudonym P, and these keys are kept in a secure hardware device such as a VISA security module [VSM] at the bank. They are not disclosed to any third party (except possibly to a court in the case of later dispute).

The customer may now purchase digital coins in sticks which in our prototype system contain 100 coins each. The j-th stick of coins will have

$$C_i = \{i, j, B\}_{K_{BP}} \tag{2}$$

where the bank's name B is used to provide redundancy.

Each stick of coins is digitally signed by the bank. Again, the bank's name is inserted to provide redundancy, as is the issue date t, and the coin stick is then encrypted and supplied to the customer as

$$CS = \{C_1, C_2, \ldots, C_{100}, \{B, t, C_1, C_2, \ldots, C_{100}\}_{KB^{-1}}\}_{KP} \quad (3)$$

The signature protects the customer against fraud by the bank or its employees; if the bank later fails to honour a coin or even wrongfully accuses her of having passed an invalid coin, she can produce the stick signature in her defence.

When the customer wishes to make a purchase from a merchant, she first certifies the genuineness of the coins which she is about to spend. She may not be able to tell in advance how many coins will be used, as she may not know how many pages of a learned journal she will have time to read, or how much of an online video she will watch. So she may vouch for a whole stick of them. She does this by hashing the coins $C_1, \ldots, C_{100}$ recursively as follows. Firstly, she creates a set of common data CD that is shared between her and the merchant and that specifies the transaction completely (this will include both their names, the date, sequence numbers and whatever else is relevant; the details are not so important as the uniqueness of the value CD). Then the last coin value is hashed with CD to give $h_{100} = h(CD, C_{100})$, and each of the coins is hashed into this value in turn:

$$\begin{aligned}&\text{for } i = 99 \text{ to } 1 \text{ step -1}\\ &\qquad h_i = h(CD, C_i, h_{i+1});\end{aligned} \quad (4)$$

Finally the customer signs h_1 together with the common data CD and sends this to the merchant, together with her public key certificate CP:

$$S = \{h_1, CD\}_{KP^{-1}} \quad (5)$$

The effect of her signature S, which will be made explicit in her service contract, is that she vouches for the coins: she claims they were properly issued by a bank and not spent elsewhere in the meantime. The fact that the coins are kept, and her signature issued, by a tamper-resistant device protects honest customers against accidental double spending (in a software implementation, accidental double spending could occur if a hard disk were restored from a backup).

The merchant checks the signature against the certificate, and if these verify he signals her to begin the transaction. To spend the first coin, she sends the merchant the values C_1 and h_2. If $h_1 = h(C_1, h_2)$ he accepts the coin as valid. To spend the second coin, she sends C_2 and h_3, and so on.

In order to redeem the coins, the merchant presents to the bank the customer's signature S, the coins $C_1, \ldots, C_m$ actually spent by the customer and the corresponding hash values $h_1, \ldots, h_m$. The bank checks the coins, verifies that none of them has been spent before, and credits the merchant's account. The bank may guarantee the transaction so long as the signatures and hashes are consistent, and the total value spent does not exceed the customer's credit limit (otherwise the validity of the coins could be checked online as with credit card transactions over the merchant's floor limit).

If there are multiple coin issuers in a system, then the merchant may use a different bank from that which issued the coins. In this case, an interbank settlement mechanism is constructed in the traditional way; the information presented by the merchant $\{CD, S, C_1, \ldots, C_m, h_1, \ldots, h_{m+1}\}$ is treated as a cheque and sent to the bank B through a clearing house.

During the third and fourth quarters of 1995, we implemented a version of this protocol. We found that handling many coins at the bank was tiresome, and set out to streamline the system. We were inspired by two comments and a business development.

The business development was the issue by VISA and MasterCard of the SET specification, which specifies (inter alia) that customer electronic banking applications may be software-only and that all transactions would be authorised online: SET is a zero floor limit protocol. We had not anticipated this, because of the increased processing costs for banks. However, Spanish banks' move to online-only eftpos in 1988 had reduced card fraud from 0.21% of turnover in 1988 to 0.008% in 1991 [BABE], and this may have persuaded US banks, with their much higher fraud rates, that online operation was worth adopting.

The first comment, from Bruce Christianson, was that only one coin in each stick need be supplied by the bank. The others coins could simply be counters so long as both bank and customer signatures were present and the hash function was strong. The second comment, from Jan Camenisch, was that one might as well have the customer generate the coins.

5 Second Protocol

Before the reader concludes that this arrangement is impossibly favourable for the customer, let us outline the basic transaction flow by means of an example.

1. The customer, Alice, decides to purchase some service from an online provider. To take a concrete example, a customer in San Francisco decides to purchase a number of pages from an academic journal published by Bernd whose firm is in Germany. The page price is 37 pfennigs.
2. Alice does not know how many pages she will want to purchase, but thinks she is unlikely to need more than 100. So she creates 100 coins for 37 pfennigs each and sends a signature on them to Bernd together with her credit card details.
3. Bernd goes to his bank and gets a pre-authorisation for 37 Deutschmarks. This validates the coins and he sends a message to Alice saying that she can now download pages at one coin each.
4. Alice now downloads 22 pages; for each of them she sends Bernd one of the coins, whose validity he can easily check.
5. She signals the end of the transaction, whereupon Bernd sends a capture message to his bank which files a debit of DM 8,14 to Alice's account. The capture message contains a short string that proves that he received the 22

coins that he claims to have received; he is not able to defraud Alice by claiming for coins that he did not in fact receive.

6. The system is also secure in that all disputes can be resolved. All parties to the transaction — Alice, Bernd and their bankers — have authenticated evidence of the actions of the other parties.

The formal details are as follows. When a customer wishes to use m digital coins for denomination δ in a transaction with merchant B, she first generates the common data CD that are unique to the transaction. She then generates a random nonce N_A and calculates the last coin, C_m, as

$$C_m = h(CD, N_A) \tag{6}$$

Next, the rest of the coins in the stick are generated in reverse order by hashing the previous coin with the common data and the index of the coin in the stick:

$$\begin{aligned}&\text{for } i = \text{m-1 to 0 step -1}\\ &\qquad C_i = h(CD, i, C_{i+1});\end{aligned} \tag{7}$$

Finally she signs h_0, the coin denomination δ, the number of coins m and the common data CD, and sends these to the merchant.

This can be thought of as a prepayment transaction of the type used to provide deposits for rental cars and hotel reservations; the merchant makes an authorisation request for the maximum amount, namely $m\delta$, with the SET `SaleInd` flag set to *false* so that the credit card network treats it as a preauthorisation rather than as an immediate sale. Given a positive authorisation response, he sends a signed acknowledgement to the customer.

She can now spend the coins C_1, C_2 and so on, and the merchant can verify them by a simple hashing operation. Once the transaction is complete, and the merchant has received coins C_1 ... C_k, the merchant sends the bank C_k plus the total amount claimed. The bank can check the transaction by dividing the amount claimed by the coin denomination δ to get the number of coins spent, and work back through that many rounds of hashing to check that the result is the same C_0 that appeared in the original authorisation request.

6 Third Protocol

The second, more lightweight, protocol has a number of advantages over the first version. It lets customers to make small transactions in any currency they choose, with electronic coins of whatever denomination is convenient. It is made possible by the fact that all SET transactions are authorised online against the customer's current available balance. So long as this is an effective control on double spending, the second protocol appears safe.

A variant will be needed in the interim to work with the first version of SET. This draft does not explicitly support micropayments; there is no room in the message formats for the coins, and the customer does not sign anything for the merchant. Nonetheless, we can still bootstrap micropayments on top of SET compliant systems. The customer can generate another signature to sign the coins for the merchant, and he in turn can simply keep this together with the coins as evidence that she did indeed spend $k\delta$ with him. This is directly comparable to the way in which current terminal draft capture systems print out a draft that is signed by the customer, and retained by him against the possibility of a disputed transaction.

However it would be simpler if the micropayment system were supported in SET or its successors. Assuming that this comes to pass, what would the ideal protocol look like?

7 Fourth Protocol

Neither the first version of SET nor the second version of our protocol has much room for error. Online authorisation could fail because of attacks on the underlying legacy systems; alternatively, there might be a systematic failure of the public-key front end that SET bolts on to them (e.g., a leak or false revocation of a root certification key). Regardless of the cryptographic sophistication, real world systems inevitably have bugs in their design, implementation and operation. Such bugs account for the majority of actual frauds on banking systems in the past, and recovering security has in some cases been very expensive because it was not a design requirement [And2].

So it is prudent to minimise the number of single points of failure, and enable security to be recovered after unexpected failures (insofar as this is possible). How can such resilience be built into a micropayment system?

Our suggestion is that instead of moving all the way from the first protocol to the second by letting the customer manufacture the coins, we retain the requirement that the bank supply at least one coin in each stick, which we will call the root coin. The customer will be issued with a limited supply of root coins, and may have a smartcard or other tamper-resistant device that performs part of the protocol and thus makes double spending difficult. The resulting protocol is similar to the second protocol but with the nonce N_A replaced by a root coin C_j supplied by the bank. In addition, instead of signing h_0 and sending it to the merchant, the customer will sign and reveal $h(h_0)$.

This improves substantially on the second protocol since, in the event of a failure of other security mechanisms, banks can move to a fully online system in which transactions are checked online by the card issuing bank wherever fraud rises above the level at which this becomes economic. As the issuing bank knows C_j, it can calculate and return h_0 as an authorisation response through VISA and the acquiring bank to the acquirer gateway (which provides the guarantee to the merchant).

Of course, precautions may have to be taken to prevent a false host attack in which customers collude in real time with someone doing an active wiretap on the legacy systems between the acquirer gateway and the card issuer. There are a number of options:

- the issuing bank could sign h_0. However, this would not protect against a catastrophic failure of the certification authority structure;
- the issuing bank could return h_0 encrypted under the issuer working key used in the current legacy systems. However this would not protect against a catastrophic failure of the legacy infrastructure, such as if organised criminals build a DES keysearch machine to break zone control master keys;
- one might even ignore the problem, arguing that legacy systems have been vulnerable to message manipulation for decades; that there have been no significant attacks; and that if any materialised, they could be dealt with piecemeal by installing line encryptors and other ad hoc controls.

Whatever the engineering details, this fourth protocol can clearly give us a low cost security recovery mechanism. Its independence of most of the SET and legacy system processing suggests that it could help us to recover after many of the unpredictable failures that experience shows to be inevitable.

8 Conclusion

Our recursive hashing technique greatly reduces the computational complexity in applications where a series of low value payments are made to the same merchant. We have shown how it can be used in simple payment schemes based on both the smartcard and the online processing models of electronic commerce, and can also provide some novel and valuable features, such as a security recovery facility that does not depend on either the legacy systems or the SET protocols. It is an open problem whether hashing techniques can be combined with the more complex anonymous cash schemes.

In December 1995, we learned that three other groups had independently developed micropayment systems that are rather similar to our second protocol. These are the 'Tick Payments' of Torben Pedersen of the CAFE project, the 'PayWords' of Ron Rivest and Adi Shamir [RS], and a scheme from the iKP team at IBM Zürich [HSW].

From the scientific point of view, one of the more interesting lessons learned from implementing our first protocol and developing the others from it has been that local and global trust interact in interesting and often unexpected ways. The details of this will be the subject of a future paper; the high order bit appears to be that the global trust has to go somewhere. In a payment system, the global mechanism to prevent double spending can be a centralised system of online authorisation, authorisation using end-to-end authentication, tamper resistant objects or (more realistically) some combination of these. Moving the primary locus of trust, even slightly, can have profound effects; and very small design changes can greatly improve the system's resilience and robustness.

Acknowledgement: The first author is grateful to the Isaac Newton Institute for hospitality while this paper was being written. The second and third authors were supported by the DTI funded NetCard project. All three authors acknowledge the help of Mike Roe and other colleagues at the security group at Cambridge University in tweaking bugs in early versions of this protocol.

References

[And1] "UEPS — A Second Generation Electronic Wallet", RJ Anderson, in *Computer Security — ESORICS 92*, Springer LNCS v 648 pp 411–418

[And2] RJ Anderson, "Why Cryptosystems Fail", in *Communications of the ACM* v 37 no 11 (November 1994) pp 32 - 40

[AB] "Cryptographic Credit Control in Pre-payment Metering Systems", RJ Anderson, SJ Bezuidenhout, *Proceedings, 1995 IEEE Symposium on Security and Privacy* pp 15–23

[AN] "Programming Satan's Computer", RJ Anderson and RM Needham, in *Springer Lecture Notes in Computer Science volume 1000*

[Beg] "Fast Server-Aided RSA Signatures Secure Against Active Attacks", P Béguin, JJ Quisquater, *Advances in Cryptology - CRYPTO 95*, Springer LNCS 963 pp 57–69

[BABE] "Card Fraud: Banking's Boom Sector", in *Banking Automation Bulletin for Europe* (Mar 92) pp 1–5

[BFLAR] S Blythe, B Fraboni, S Lall, H Ahmed, U de Riu, "Layout Reconstruction of Complex Silicon Chips", in *IEEE J. of Solid-State Circuits* v 28 no 2 (Feb 93) pp 138–145

[Cha] "Achieving Electronic Privacy", D Chaum, *Scientific American* (August 92) pp 96–101

[CAFE] "The ESPRIT Project CAFE — High Security Digital Payment Systems", JP Boly, A Bosselaers, R Cramer, R Michelsen, S Mjølsnes, F Muller, T Pedersen, B Pfitzmann, P de Rooij, B Schoenmakers, M Schunter, L Vallée, M Waidner, in *Computer Security — ESORICS 94*, Springer Lecture Notes on Computer Science volume 875 pp 217–230

[HSW] "Micro-Payments based on *i*KP", R Hauser, M Steiner, M Waidner, *preprint*, IBM Zürich, January 16th 1996

[Ped] "Electronic Payments of Small Amounts", TP Pedersen, Aarhus University Technical Report DAIMI PB–495, August 1995

[RS] "PayWord and MicroMint–Two Simple Micropayment Schemes", RL Rivest, A Shamir, *preprint*, MIT, January 26, 1996

[SET] *Secure Electronic Transactions*, VISA and MasterCard 1996

[VSM] *VISA Security Module Operations Manual*, VISA, 1986

[Wie] "Electro-optic sampling of high-speed devices and integrated circuits", JM Wiesenfeld, *IBM Journal of Research and Development* v 34 no 2/3 (Mar/May 90) pp 141–161