Papers on Smartcard Engineering

Ross Anderson
University Computer Laboratory
Pembroke Street, Cambridge CB2 3QG
Email: `rja14@cl.cam.ac.uk`

This technical report contains two papers on a smartcard based electronic wallet system which I helped to design. The first, 'Making Smartcard Systems Robust', appeared in the proceedings of Cardis 94 (now out of print): the second, 'UEPS - A Second Generation Electronic Wallet', appeared in the proceedings of ESORICS 92 (Springer LNCS v 648 pp 411–418). This system has been fielded in a number of countries, including South Africa, Namibia and Russia.

Its research interest stems a number of factors. Firstly, it was the first (and as far as I am aware is still the only) banking system whose design was verified using formal techniques; secondly, the chaining mechanisms used in the underlying transaction protocol prefigured much of the current work on cryptographic protocol robustness; and thirdly, it shows that even using symmetric cipher technology, it is possible to build a system with a strong degree of accountability, and which can give users a high degree of confidence that they will not be the victim of frauds by bank insiders.

Making Smartcard Systems Robust

Ross Anderson
University Computer Laboratory
Pembroke Street, Cambridge CB2 3QG
Email: rja14@cl.cam.ac.uk

Abstract

Smartcards are often sold as the solution to almost all information security problems. However, placing too much faith in any technology can lead to credibility problems; recent ATM disputes in Norway - and the problems experienced by the pay-TV industry - have shown that some smartcard systems do fail. We discuss their failure modes, and show how a prudent designer can minimise the risk of failure by making his system robust. We explore the nature of robust security: it affects all levels of a system, from security goals through the functionality and protocol levels to the cryptographic algorithms and their interactions. Finally, we discuss a fielded payment system, UEPS, which may provide a paradigm case of how to build a robust smartcard application.

1 Introduction

One of the smartcard vendors' most powerful arguments is the ease with which magnetic strip cards can be forged. Forgery has not only caused direct losses to card issuers and customers, but has led in some countries to serious embarrassment: highly publicised disputes between banks and their customers erode confidence and cast doubt on the integrity of the whole payments system.

For example, phantom withdrawals from automatic teller machines (ATMs) have been the main source of complaints to the UK financial sector ombudsmen in recent years. Yet despite a report from a parliamentary commission of enquiry into banking law which concluded that the system of personal identification numbers was insecure [J], the ombudsmen have followed the banks' line that their systems are infallible and refused to award compensation [A1].

Some of the resulting court cases have been very controversial. In one recent trial, a policeman who had complained about six phantom withdrawals from his account was accused of attempting to obtain money by deception and convicted. The resulting press outcry [E] unnerved the banking industry, and the institution responsible publicly denied that it asked for a prosecution (contrary to the evidence given by their fraud manager in court).

Bad publicity is the banker's nightmare, and preventing it striking at payment systems is a goal of all central banks. However, there is nothing magic about smartcards which prevents application designers making blunders which break their systems' security. This has happened with with satellite TV decoder systems and with prepayment electricity meters [AB]; it now appears to have struck the financial sector too, with disputes in Norway which may be a harbinger of problems to come.

2 Credibility Problems in Norway

In Norway, as in a number of other countries, the banks invested millions in issuing their customers with smartcards, and are now quite adamant (at least in public) that no debit can possibly be made to a customer account without the actual card and PIN issued to the customer. Yet a number of thefts at the University of Trondheim have cast serious doubt on their position.

In these cases, smartcards were stolen from offices on campus and used in ATMs and shops in the town; the victims, who include highly credible witnesses such as senior academic staff and overseas visitors, are quite certain that their PINs could not have been compromised. In any case, the nearest ATM is several kilometres away, so if the victims were observed while using their cards, then the thief must have followed them a considerable distance.

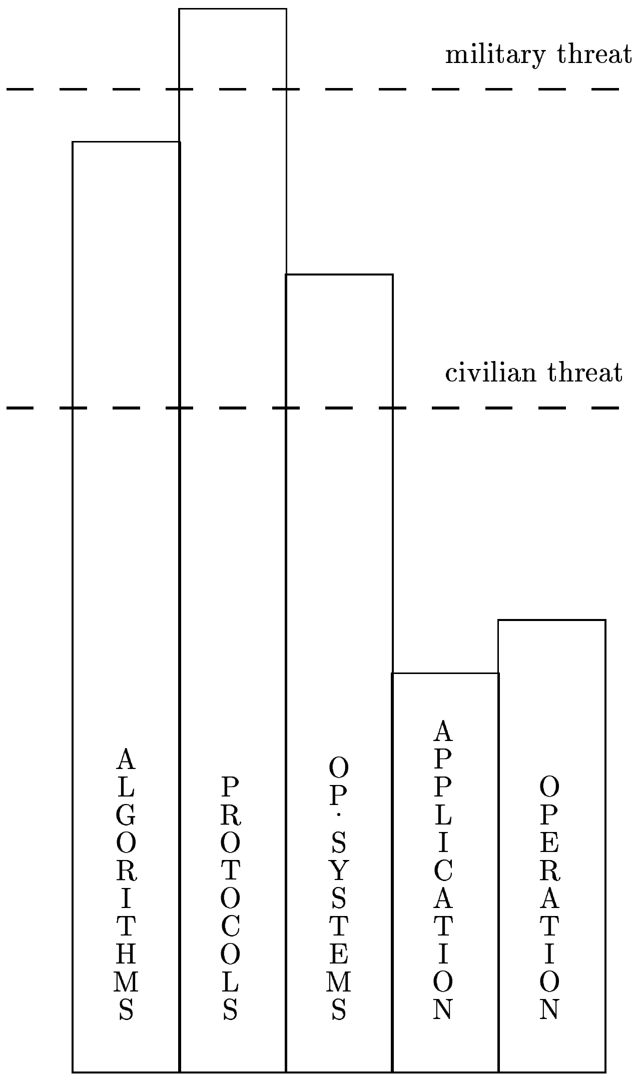
The Norwegian banking ombudsman (Bankklagenemnda), when faced with a complaint from one of the victims, asked the banks for technical details on the system in use. The banking association (Bankforeningen) refused, claiming that *'an important part of the security in the card systems is that the routines are not documented and publicly known. Such an independent investigation will therefore reduce the level of security. Bankforeningen is therefore against letting for instance a consultancy company acquire knowledge of the routines in order to investigate them'* [BN].

The ombudsman then went to Norges Bank, the country's central bank, and asked it to investigate the system. Norges Bank agreed, but even it was not provided with any technical information; Bankforeningen simply gave it 'managerial' assurances that the system was sound, which were passed on to the ombudsman. He then used these assurances an an excuse to deny compensation to the victim.

By now, the reader may be starting to wince, and will wince more when told that the disputed transactions violated the card cycle limits: although only NOK5000 should have been available from ATMs and NOK6000 from eftpos, the thief managed somehow to withdraw NOK18000. The extra NOK7000 was refunded without any explanation.

3 The Nature of Civilian and Military Threats

Our studies of security failure in magnetic strip payment systems [A1] [A2] showed that the vast majority of security failures which led to actual losses occurred at the level of implementation or operation. The following diagram may give a useful conceptual model: the five columns show the amount of confidence available in cryptographic algorithms, protocols, operating systems, applications and operational controls respectively, while the two horizontal lines show the 'military' and 'civilian' threat levels - a capable motivated opponent and an opportunistic opponent respectively.



Algorithms We have a number of encryption algorithms which are thought to be good (DES, IDEA, ...), but since certain government agencies knew about differential cryptanalysis twenty years ago [C], there remains the fear that one of these agencies might find a shortcut attack on our algorithm. But while this may be a serious concern for military system builders, it need not concern most commercial designers overmuch.

Protocols Thanks to formal methods such as the BAN logic [BAN], and to robustness properties which will be described shortly, the protocol security problem can be considered solved. This does not of course mean that there are no weak protocols around; on the contrary, as the successes of the BAN authors [AN] and the recent controversy over ISO 11166 [R] [A3] showed, and recent attacks on satellite TV scrambling systems have confirmed, there are plenty fielded systems which are insecure. However, a diligent designer can now avoid such attacks completely by using publicly available techniques.

Operating systems: Operating systems have also been extensively studied, and (thanks to TCSEC and ITSEC) there are a number of products on the market, ranging from commercial discretionary access control products to military multilevel systems. Of course, military products are usually too expensive and out of date to be used in commerce and industry; and weaknesses are still found in military products from time to time. However, experience shows that if one disregards viruses and worms, the features of the operating system are almost never the main cause of real attacks on commercial systems.

Applications: Here at last we come into the world of real commercial threats. A large number of attacks on electronic payment systems have been due to application programming blunders [A2], and it seems highly likely that some such error is also responsible for the Norwegian problem. In military systems, too, there may be attacks on cryptographic algorithms, but there are many more attacks which result from application blunders [M1].

Operations: Operational blunders also cause many security failures; no matter how good the security technology, sloppy management can render it useless. For example, in August 1993, my wife went into a branch of a major English bank, and told them that she had forgotten her PIN; they helpfully printed a replacement PIN mailer from a PC behind the counter. This was not the branch at which our account is kept; no-one knew her, and the only 'identification' she produced was her bank card and our cheque book. By that time, banks in Britain had endured some eighteen months of bad publicity about poor ATM security (with which my name had been associated), and this particular bank had been a press target since April of that year. With management this dreadful, it makes no difference whether the card itself is magnetic or silicon based.

The moral is that if a customer is to get anything like full value from an investment in smartcard technology - which will usually cost many millions - then the application will have to be designed, implemented and managed in a much more robust way than has often been the case in the past. Now that VISA and MasterCard have agreed to move their enormous customer base from magnetic strip cards to chip cards over the next six years, we are about to see an investment whose sheer size may freeze certain aspects of the technology for many years at the level achieved by about 1996. If the robustness aspects are not got right, and the architectures adopted by banking networks are fragile, then the industry's credibility could be imperilled. Robustness should therefore be an urgent concern of smartcard vendors.

4 The Nature of Robustness

The big question is therefore: how do we build robust applications using smartcards? Without this, the product adds little value. However, there has until now been no generally accepted idea of what robust security consists of.

In the rest of engineering, the exact nature of robustness varies from one discipline to another. Bridge builders know that most possible faults, such as poor steel, contaminated concrete, or uneven weathering, just reduce the structure's breaking strain slightly; so the usual rule is to design a bridge so that its theoretical breaking strain with optimal materials is six times what is required, and to proof test samples of the actual construction materials to half that, or three times the needed strength [P].

In aircraft engineering, on the other hand, many accidents are caused by the failure of critical components, and so designers make extensive use of redundancy. With very critical functions, this may extend to design diversity: for example, a typical modern airliner will determine its flight attitude using two electronic flight instrumentation systems, which use gyros driven by the main power circuits; but if these both fail for some reason, there is a 1950's technology artificial horizon with pneumatic gyros, and a 1920's vintage turn and slip indicator driven by a battery.

But neither overdesign nor redundancy is enough for the secure systems designer. Just using a number of weak encryption algorithms one after another will not necessarily have the same effect as using a strong algorithm; and the unthinking use of redundancy in computer systems can be dangerous, as resilience can mask faults which would otherwise by found and fixed.

However, the fact that blunders are responsible for most real security failures in both civilian and military computer systems shows that robustness is just as much a problem as it is for people building bridges or aircraft. We might expect it to be even harder to pin down to a general principle, since we have to consider what it means at a number of levels. We will look at the algorithm, protocol, application and management levels in turn.

4.1 Robustness of Algorithms and their Interactions

As we noted above, cryptographic algorithms are not the most pressing concern of the commercial secure systems designer. We visit this topic briefly because our work on algorithm interaction gave the insight that algorithm robustness is about explicitness.

Various authors had looked at the problem of how we are to prevent unwanted interaction between cryptographic algorithms, such as between a hash function and a digital signature algorithm. Since the Diffie Hellman paper in 1976 [DH], it had been recognised that hash functions should be one-way, in that it is easy to work out $h(M)$ from M but hard to find a suitable M given only $h(M)$. This was difficult to formalise, because of the implicit temporal ordering, so a second property was proposed, of collision freedom: that it is hard to find two different inputs M and M' such that $h(M) = h(M')$ [D].

For a number of years, cryptographers were content to stipulate that hash function primitives should be one-way and collision free. Then in 1992, Okamoto proposed a third primitive, correlation freedom: a function h is correlation free if it is hard to find two different inputs M and M' such that $h(M)$ and $h(M')$ differ in only a few bits. He conjectured that correlation freedom was a strictly stronger property than collision freedom.

Last year, we showed that this conjecture was true. In fact, we showed that there are many properties which we might want a hash function to have, such as multiplication freedom (we can't find X, Y and Z such that $h(X)h(Y) = h(Z)$) and addition freedom (ditto $h(X) + h(Y) = h(Z)$), and we showed that these properties were independent of each other by constructing functions which had some freedom properties but not others [A4]. This gave us the insight that, at the application level at least, we have to be explicit about the properties which our application requires from any cryptographic primitives we use.

4.2 Robustness of Protocols and Operating Systems

Robustness in cryptographic protocols made its formal début in the academic literature in May 1993: three papers appeared in which it was proposed as a solution to the problem of designing authentication schemes.

1. At the 1993 Cambridge Workshop, Li Gong argued that most protocol errors occur when people try to be smart, and called for a protocol equivalent of structured programming. For example, by putting in the name of the sender and the recipient of each message in a protocol, and by insisting on freshness, most of the cut-and-paste attacks in the literature could be prevented [G].
2. A month later, at Eurocrypt 93, Boyd and Mao suggested that a protocol should be called robust if authenticating any message depends only on information contained in the message itself or already in the possession of the recipient, and that the purpose of a message in a robust protocol should be capable of being deduced without knowledge of the context [BM].
3. The following day, at Oakland 93, Woo and Lam proposed that protocols be made robust by chaining successive steps, and developed a formal semantics for protocols which are immune to cut-and-paste attacks (in their terminology, such protocols possess the 'correspondence' property). They argued that these are absolutely fundamental to proving other security properties [WL1].

Each of these proposals, which appeared logically simultaneously, tackles part of the problem. However, none of them is adequate on its own; protocol failures are known which result from the lack of name, freshness and context information within the security envelope. The interested reader is referred to [AN] for examples.

Our approach to the problem is that explicitness is the proper organising principle for security robustness. This is put forward in [A2], and draws on two sources - firstly, the work on algorithm interaction mentioned in the previous section, and secondly, the experience of a system which used robust protocols and which we implemented in 1991 [A5]. We will discuss this system, UEPS, in section 5 below.

Explicitness subsumes the above three proposals - lack of name, lack of freshness and lack of context are all failures of explicitness. In each case, some information - which both parties know implicitly - is not explicitly expressed within the security envelope, and can thus be manipulated by an opponent.

Explicit protocols have been criticised as expensive [S]; and Woo and Lam have recently experimented with starting from an explicit protocol and looking to see what information can be safely taken out [WL2]. However, the philosophical point here is that optimisation is a process of replacing something which works with something which almost works, but is cheaper; and if the 'optimised' version is not cheaper at all, then the process is undesirable. In fact, the implicit information in a cryptographic protocol (such as 'this is message 3 in a run of Kerberos version 6 subprotocol 17') is already known to both parties, so there is no added communication cost; and as most real protocols either use fast symmetric ciphers, or use fast hash functions to reduce messages prior to calculating a digital signature, it is quite unclear that there is ever a significant computational cost either.

In fact, the protocols used in UEPS [A5] are efficient as well as robust. When designing them, we bundled in everything for two reasons - firstly, to save on code space and to avoid expanding the communications protocol (we had limited ROM and very slow offline devices for inter-card transactions), and secondly, to facilitate the formal verification of the protocol.

4.3 Application Program Robustness

Much of the robustness needed in application programs will be a matter for software engineering. However, even here explicitness plays a significant rôle, with many common methodologies emphasising the need to elicit requirements in the most explicit possible form, to validate these by test cases or prototyping, and to ensure that they are coded and tested properly. However, there are some aspects of the application layer which deserve special attention.

Many older banking systems have the problem that application programmers have discretion over what security features to implement. Consider for example the security modules used in networks of ATMs to encrypt personal identification numbers [VSM]. These devices return a whole series of response codes which may be significant to the security manager: for example, if a programmer starts making unauthorised experiments with live keys, he will probably give rise to a key parity error message.

It would thus clearly be prudent to monitor these response codes, and the obvious solution is to write a device driver which intercepts all abnormal response codes from the security module and brings them to the attention of the audit or security staff. However, no institution we have seen implements proper monitoring; as the system will 'work' without it, and as device drivers are tricky to write, the job never gets done.

One of advantage of a trusted computing base (TCB), whether built from smartcards or from security modules, is that we can design the TCB's transaction set in such a way as that application programmers have to do all the necessary checks. This was done in UEPS by chaining each message to the next; thus a programmer who tries to cut corners by discarding some message information will find that he cannot make the system work at all.

Of course, this strategy can be more productive with a distributed TCB, such as we can implement with smartcards, as the checking can be made much more pervasive. We can manage security state information, such as response codes and certain aspects of transaction history, more intelligently where there is a component of the TCB in every logical node of the network, rather than just having one security processor attached to a mainframe which runs perhaps a hundred security critical processes such as branch accounting, interbank reconciliation and audit.

Suppose for example that we allow bank branch staff to initialise cards for customers. In countries where poor telecommunications force us to work largely offline we will typically have no choice but to do this; UEPS, for example, uses a teller card to load some crypto keys and other initialisation data to the customer card. Suppose we also expect that about 1% of our branch staff will try to embezzle money in an average year [A1]; we might then decide to put the issuing teller's identity, not just on each card, but in every transaction as well. In this way, we can provide a distributed audit system, and open new possibilities for fraud detection.

The key point is that smartcards are objects which can retain security state; and this state need not be limited to the customer's account balance. A well designed system will use it to provide much higher assurance of application functionality than was previously available, and to ensure that partial, insecure implementations do not work at all.

4.4 Robustness at the Management Level: Making Goals Explicit

As noted above, the major management mistake made by banks in the UK and elsewhere was that they did not make their security goals explicit, and in particular they did not recognise the need for a fair arbitration procedure. To quote Tom Watson's 1977 Oxford address:

> 'One of the most important problems we face today, as techniques and systems become more and more pervasive, is the risk of missing that fine, human point that may well make the difference between success and failure, fair and unfair, right and wrong ... no IBM computer has an education in the humanities' [W]

Quite simply, failures are inevitable, and if there is no mechanism to deal with them, then the bank will have to choose between paying all claims and paying none of them. The courts have forced US banks to take the former course [JC], which leaves them reliant on ATM cameras to prevent malicious claims; while banks in the UK and Norway opted for the latter - at the cost of bad publicity, litigation and loss of public confidence.

Furthermore, when banks claim (as the Norwegians did) that their security systems cannot withstand public scrutiny, this also shows that their security goals were not properly thought through. Disclosure of evidence is one of the most ingrained features of the legal systems in free societies - an accused person has the right to see and to challenge every link in the chain of evidence against him, and in most countries the requirements for discovery in civil cases are hardly less strict. Every cryptographic system whose main purpose is to establish liability must be built on the assumption that it will have to withstand scrutiny by hostile expert witnesses in court [A6].

The danger of missing the fine human points seems to be especially acute when designing smartcard based systems, as the technology seems impressive, and the temptation to hubris is correspondingly strong.

4.5 Tying it All Together: Relating Goals to Mechanisms

Finally, we need to be very careful when following a line of reasoning from goals - however well researched, human friendly, legal and thoroughly agreed - to mechanisms. This part of the security design process is well known to be fraught with danger.

Morris reports an interesting explicitness requirement imposed by the NSA in its own internal evaluations [M2]. Each product must have not just an operations manual describing its normal use, but also an attack manual which describes every combination of technical attacks, blunders, and subversion of personnel which could succeed in defeating it. The attack manual may be highly classified and available only to a small number of evaluators, but it must exist, and it must be comprehensive; if the evaluators can find any other significant attacks, the product will be rejected.

This innovation may be valuable in the commercial world as well, and in general, the explicitness principle can be propagated down through the design in a structured way, using techniques developed by the safety critical systems community [A1].

However, where the principal function of a cryptographic system is establishing liability, then the real test must be whether it can stand up in court against hostile expert witnesses. Here, one needs standards of best industry practice, as courts will use these to determine which party to a dispute has been more negligent [A6]. For this reason, we will now discuss some of the robustness features of UEPS.

5 Robust Payment Systems: Best Industry Practice

UEPS, the Universal Electronic Payment System, was designed during 1991 by systems house Net One for its client the Permanent Building Society in Johannesburg, and implemented later that year; its purpose was to extend modern banking services at low cost to South Africa's black population.

The Permanent Building Society, which merged with Nedbank to become NedPerm, had more black customers than any other financial institution in South Africa; but most of these customers only had savings book accounts, and the charges traditionally made for cheque clearance (up to R12 or about \$4) were so high relative to black incomes as to prevent the spread of cheque accounts to this sector of the market.

It was felt that a new product delivery system was needed, which should cater for point-of-sale transactions and keep costs as low as possible. However, the country's poor telecommunications meant that many transactions would have to be carried out offline, and even if the telephone lines had been available, online processing imposes fairly high costs on point-of-sale systems as very high availability is required. Thus offline processing was chosen, and this prevented the use of traditional magnetic stripe cards, as the forgery risk would have been unacceptable. Finally, high levels of illiteracy suggested the use of PINs rather than signatures to authorise transactions.

These requirements drove the design of UEPS, which has been a commercial success. After a trial from 1991 - 1993 under NedPerm's brand name 'Megalink', during which there was no single case of fraud detected, it was adopted by all four major South African banking groups with only minor changes. It is now projected that there will be 2 million cards in issue by early 1995.

The system has been sold to other customers too. It is being introduced by banks in the neighbouring country of Namibia, and a version of it is used by South African Breweries to manage accounts with tavern and liquor store owners. Most recently, after a year's trial, it has been adopted by Sperbank, a large savings bank in Russia, which plans to deploy it during February/March 1995 at 35 branches in 14 regions of the former Soviet Union.

The system relies on value transfer between smartcards. A customer loads her card with money from a card held by a bank teller or installed in an ATM; she then makes purchases by transferring value to a merchant card; and the merchant in turn uploads his takings to his bank via an ATM or terminal.

5.1 Algorithm robustness

We chose DES as the encryption algorithm as the client was comfortable with it and as public key smartcards were not available in 1991. Because keysearch was already considered a threat by the banking community [GO], the only robustness feature incorporated at the algorithm level was double encryption (this is admittedly one feature which is more overdesign than explicitness!).

5.2 Protocol robustness

The payment protocols used in UEPS have the robustness property that all the mutual information between the two parties is made explicit, by being incorporated into the message keys. A run of a protocol between cards A and B, using key pair K1 and K2, and message blocks A1, B1, A2, ... looks like this:

A $\longrightarrow$ B: K1(K2(A, B, A1))

B $\longrightarrow$ A: K1(K3(B, A, B1)) where K3 = K2(A, B, A1)

A $\longrightarrow$ B: K1(K4(A, B, A2)) where K4 = K3(B, A, B1)

The primary design objectives here were to implement both message chaining and double encryption using the minimum possible amount of code space, and to make formal verification of the protocols easier (for details of the verification see [A5]).

Since the Megalink prototype, key diversification has been introduced. The original system had the same keypair K1 and K2 present in every card (albeit one was loaded at the factory and the other at initialisation); in the new system, only the cards embedded in bank and merchant terminals possess a set of universal secrets, and the customer cards have keys derived from their serial numbers using these master keys.

5.3 Goal and Application Robustness - Resolution of Disputes

We did not repeat the ATM designers' mistake of forgetting to provide a means of arbitrating disputed transactions. In fact, a number of features were built into UEPS explicitly for dispute resolution.

Firstly, there are two signatures on each digital payment: one generated with a key known only to the issuing bank and the customer card, and one generated with a key controlled by the clearing house and loaded by them to the card before it is supplied to the bank. The latter signature is checked before funds are credited to the retailer presenting the cheque, while the former would only be checked in the event of a dispute. Having separate signatures was considered important given that the primary legal liability is from the card issuing bank to its customer, and that an unknown number of banks of varying levels of technical sophistication were expected to join the scheme.

A further feature, not mentioned in [A5], was writing encrypted audit trails on the transaction receipt using two separate keys, one known to the customer and his card issuing bank but not the merchant or the clearing house, and the other known to the merchant and the clearing house but not the customer or his bank. In the event of a dispute getting as far as trial, courts are likely to accept transaction logs as evidence, and especially logs which are not written on magnetic media but on paper - and of which copies are kept by both the customer and the retailer. This solves most of the problems with computer evidence discussed in [A6].

Incidentally, although the key used by the customer's card to encrypt a receipt record can be recreated by his bank's security module, and the merchant's key by the central clearing house, this need not have been the case; the keys could exist nowhere else except in the card which uses them. In that case, one would require any customer (or merchant) who disputes a transaction to produce his card to an arbitrator in order to decrypt the record, This shows that, even where we are using a symmetric algorithm, cryptographic keys do not have to be shared to provide a valuable service. This is counter to the general intuition, and relies ultimately on the assumption that the card is tamper evident.

Another benefit of having a number of independent mechanisms is that in the (hopefully rare) event of a technical attack, it is unlikely that all the mechanisms will be broken. After all, some of the mechanisms (such as the encrypted audit records) do not have to be circumvented in order to obtain value from the system, and in fact could not be circumvented unless the attacker had access to the victim's card (or to the bank's security module).

It follows that, with a high degree of probability, the existence of an attack would become evident. This avoids the risk of placing an unmeetable burden of proof on the complainant, which was what decided the US courts against the banking industry in magnetic strip card disputes [JC].

Finally, there is a rule that all merchant transactions must be banked within 14 days. This means that, unlike the similar Mondex system now proposed in the UK [MA], UEPS can refund lost customer cards after a short delay. In any case, settlement functions are a central part of traditional banking systems; abandoning them, as seems implicit in many electronic cash designs, could give rise to unpredictable risks.

6 Conclusion

In the real world, most things break sooner or later. Traditional engineers usually tackle this problem by making systems robust, and this may involve some combination of overdesign and redundancy; however, we have shown that robust computer security depends on explicitness as much as anything else.

This must be used in a structured way, and the starting point is to make the business goals explicit. Next we need to establish an accurate threat model and work though to the functional properties which the system needs. Security is not after all a simple Boolean attribute, but a set of properties which enable a system to fulfil its purpose (of processing banking transactions, helping to win wars, or whatever).

Once we have an explicit set of security properties which our system must have, we can look for ways in which we can use our trusted computing base to enforce them. We can also use application layer features to enforce good operational practice, and thus ensure that all our functional properties interlock strongly. The explicitness principle is also a useful guide to designing the technical features, such as cryptographic protocols, on which the whole structure is built.

Finally, robustness as explicitness is not new; we built it into the design of UEPS, a commercially successful banking system, back in 1991. That exercise showed that there are no significant computational or communications costs associated with robust protocols. The only real investment involved in building a robust security system is some extra effort at the design stage; but doing things right the first time is always cheaper in the long run.

References

[A1] RJ Anderson, "Why Cryptosystems Fail", in *Proceedings of the 1993 ACM Conference on Computer and Communications Security* pp 215 - 227

[A2] RJ Anderson, "Why Cryptosystems Fail", to appear in *Communications of the ACM, November 1994*

[A3] RJ Anderson, "A Note on ISO 11166", presented at the Crypto 94 rump session

[A4] RJ Anderson, "The Classification of Hash Functions", in *Proceedings of the 4th IMA Conference in Cryptography and Coding (1993)* (proceedings to be published by OUP)

[A5] RJ Anderson, "UEPS - A Second Generation Electronic Wallet", in *Computer Security - ESORICS 92*, Springer LNCS v 648 pp 411 - 418

[A6] RJ Anderson, "Liability and Computer Security - Nine Principles", in *Computer Security - ESORICS 94*, Springer LNCS (to appear)

[AB] RJ Anderson, SJ Bezuidenhout, "On the Security of Prepayment Metering Systems", *to appear*

[AN] M Abadí, RM Needham, 'Prudent Engineering Practice for Cryptographic Protocols', in *Proceedings of the 1994 IEEE Symposium on Security and privacy* (to appear)

[BAN] M Burrows, M Abadí and RM Needham, 'A logic of Authentication', *Report 39, Digital Systems Research Center, Palo Alto, Ca.*

[BM] C Boyd, WB Mao, 'Limitations of Logical Analysis of Cryptographic Protocols', in *Pre-Proceedings of Eurocrypt 93* pp T88 - T96

[BN] Behne v Den Norske Bank, Bankklagenemnda, Sak nr: 92457/93111

[C] D Coppersmith, *'The Data Encryption Standard (DES) and its strength against attacks'*, IBM Thomas J Watson Research Center technical report RC 18613 (81421), 22 December 1992

[D] IB Damgård, "Collision free hash functions and public key signature schemes", in *Advances in Cryptology - EUROCRYPT 87*, Springer LNCS v 304, pp 203 - 216

[DH] W Diffie and ME Hellman, "New Directions in Cryptography", in *IEEE Transactions on Information Theory*, v IT-22 no 6 (November 1976) p 650

[E] B Ellis, 'Prosecuted for complaint over cash machine', in *The Sunday Times*, 27th March 1994, section 5 page 1

[ECMA] European Computer Manufacturers' Association, *"Secure Information Processing versus the Concept of Product Evaluation"*, technical report 64 (December 1993)

[G] L Gong, 'Thoughts on Cryptographic Protocols', in *Proceedings of the 1993 Cambridge Protocols Workshop*, Springer LNCS (to appear)

[GO] G Garon and R Outerbridge, 'DES Watch: An Examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's', in *Cryptologia XV no 3* (July 1991) pp 177-193

[J] RB Jack (chairman), *"Banking services: law and practice report by the Review Committee"*, HMSO, London, 1989

[JC] Dorothy Judd v Citibank, *435 NYS, 2d series, pp 210 - 212, 107 Misc.2d 526*

[M1] R Morris, in *Proceedings of the 1993 Cambridge Protocols Workshop, Springer LNCS (to appear)*

[M2] R Morris, in *Proceedings of the 1994 Cambridge Protocols Workshop, Springer LNCS (to appear)*

[MA] H McKenzie, D Austin, 'Banks ready to do business with smart cards', in *Banking Technology v 11 no 1* (Feb 94) p 4

[O] T Okamoto, "Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes", in *Abstracts of Crypto 92*, pp 1-15 to 1-25

bibitem[R]R RA Rueppel, "Criticism of the ISO CD 11166 banking-key management by means of asymmetric algorithms", in *Proc. 3rd Sym. on State and Progress of Research in Cryptography* pp 191 - 198

[S] PF Syverson, 'Adding Time to a Logic of Authentication', in *Proceedings of the 1993 ACM Conference on Computer and Communications Security* pp 97 - 101

[ST] "How £200 can buy account details", in *Sunday Times* 29 November 1992 p 1 and p 3

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

[W] 'Former IBM Chief TJ Watson Jr dies', in *IEEE Computer v 27 no 2* (Feb 94) p 84

[WL1] TYC Woo, SS Lam, 'A Semantic Model for Authentication Protocols', in *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy* pp 178 - 194

[WL2] TYC Woo, SS Lam, "A Lesson on Authentication Protocol Design", in *Operating Systems Review* v 28 no 3 (July 94) pp 24 - 37

UEPS - A Second Generation Electronic Wallet

Ross J. Anderson

University of Cambridge Computer Laboratory,
Pembroke Street, Cambridge CB2 3QG, UK
rja14@cl.cam.ac.uk

UEPS, the Universal Electronic Payment System, is an electronic funds transfer product which is well suited to developing country environments, where poor telecommunications make offline operation necessary. It is designed around smartcard based electronic wallet and chequebook functions: money is loaded from the bank, via bank cards, to customer cards, to merchant cards, and finally back to the bank through a clearing system. This architecture is uniquely demanding from the point of view of security.

As far as we are aware, UEPS is the first live financial system whose authentication protocol was designed and verified using formal analysis techniques. This was achieved using an extension of the Burrows-Abadi-Needham [BAN] logic, and raises some interesting questions: firstly, such formal logics had been thought limited in scope to verifying mutual authentication or key sharing [GKSG]; secondly, our work has found hidden assumptions in BAN, and a problem with the postulates of the Gong-Needham-Yahalom logic [GNY], both concerning freshness; thirdly, we highlight the need for a formalism to deal with cryptographic chaining; and fourthly, this type of formal analysis turns out to be so useful that we believe it should be routine for financial and security critical systems.

1 Introduction

The OECD countries have many sophisticated networks which cater for autoteller machines, the use of credit and debit cards at the point of sale, interbank payments, and various other kinds of transaction. As the telecommunications infrastructure becomes ever faster and more reliable, these systems are increasingly online and centralised, and their existence can weaken the motive for introducing new crypto technologies such as smartcards.

Recent political developments have opened up the formerly centrally planned economies of Eastern Europe, India, Latin America and Africa to modern financial institutions and their associated payment systems. However, telecommunications are often a serious problem: decades of statism and neglect have left many of these countries with abysmal telephone networks, and villages are often without any connection at all. The lines that do exist are often not good enough to support fast modem communications or even the most basic autodial facilities. Transactions must often be carried out offline, and the risk of fraud with forged cards is such that the standard ISO magnetic stripe card with its associated PIN management techniques cannot be used.

This creates a one-off opportunity for these countries to leapfrog two generations in electronic payment technology, and go straight from manual ledger systems to distributed processing based on smartcard electronic wallets. Such systems can not only integrate retail banking and shopping functions but also provide the payment side of utility and government networks. The potential exists to slash transaction costs by more than an order of magnitude, and at the same time eliminate a major bottleneck in national economic development.

2 The Design Requirement

Our client, the Net 1 group, had secured a contract from the Permanent Building Society in South Africa to design and build a national eftpos system. This institution has the largest card base in the country with some 22% of the market, and most of its accounts were simple savings accounts. After building societies in SA were deregulated, plans were made to provide a full range of banking services.

However, poor telecommunications outside the major urban centres made it vital to carry out transactions offline. It was also necessary to be able to transact with utilities such as electricity distributors. Finally, as most customers have low incomes and had not previously used banking payment services, it was felt crucial to keep the transaction costs down, as charges similar to those made on traditional cheque accounts would have been significant compared with the customers' typical income and would have put them off using the system.

These requirements led inevitably to an electronic wallet approach, in which money is transferred between bank cards, customer cards and merchant cards using offline terminals, which can be made portable if necessary.

3 Previous Systems

First generation smartcard systems suffer from a number of drawbacks. One problem is that while the customer cards can be authenticated by a terminal, whether using challenge-response procedures or a public key algorithm, many vendors provide no such mechanism for the customer card to authenticate the terminal in turn. As a result, it may be possible to attack the card using a false terminal. One could, for example, try to record the card's data area at a time when it holds a large credit balance and then rewrite it later. Minimising the exposure to such frauds involves a number of host checks and other ad-hoc measures which are costly in processing terms and generally cumbersome and unsatisfactory.

Another problem is that banks prefer to use the familiar DES algorithm whose vulnerability to keysearch is well known [GO] and increasingly problematic. Several smartcards offer single DES encryption only, while we felt that double key encryption should be mandatory for payment systems.

The third problem is that existing regimes may result in a user carrying multiple cards, such as one for banking/eftpos, one for public telephones and one for the electricity meter. For both cost and strategic reasons we wanted a universal card which could be programmed to cater for multiple accounts on different systems, plus a secure means of transferring money between these accounts.

The final problem is integrity of design. Many existing financial systems have a long history of design evolution, and many frauds result from unforeseen loopholes which appear after seemingly unrelated changes. We felt it crucial to use formal methods to verify the correctness of the crypto protocol which forms the security kernel of the system.

4 Design Philosophy

The security of UEPS is based on two levels of authentication. The core is an electronic cheque which is generated by the client card, passed to the retailer card and then through a central clearing system to the customer's bank. The cheque carries two digital signatures: one generated with a key known only to the issuing bank's security module and the customer card, and one generated with a key which is controlled by the clearing house and loaded by them to the card before it is supplied to the bank. The latter signature is checked before funds are credited to the retailer presenting the cheque, while the former would only be checked in the event of a dispute. Both these signatures are standard message authentication codes, calculated on amount, payee and date.

Had public key technology been available, it would have been possible for a transaction recipient to check a signature. This was not an option at the time, and so we had to design a transaction protocol to prevent any bad cheques getting into the system in the first place. This uses challenge-reponse processes by which both cards in any transaction verify each other and carry on a secure session; it is similar to (but was developed independently of) the 'embedded observer' proposed by Chaum for electronic privacy applications [C]. In both cases, a trusted application in the smartcard vouchsafes for the authenticity of statements which the recipient cannot check directly.

The use of two independent security mechansims not only gives a high level of confidence, but also helps us meet audit and resilience requirements: to keep track of all the cash in the system on a day-by-day basis, and to guarantee that the compromise of any one key will not expose the whole system to fraud.

5 The Transaction Protocol

The transaction protocol is used to ensure the integrity of each step of the cash path, from bank to customer to merchant to clearer to bank. At each step, each transaction is made unique by random challenges, sequence numbers or both.

At the time UEPS was designed (1990-91), the only available programmable smartcard was the GemPlus product. This implements DES rather than a public key algorithm and we therefore had to base the cryptography on a transaction set between secure objects, a concept which is described in [DQ] and is already familiar in the banking industry.

Our first task, in view of our requirement for double encryption, was to implement a way of doing this which was within the technical constraints of the card. This was done by key chaining. Given a run of a protocol between two cards A and B, using a key pair, K1 and K2, and a series of message blocks A1, B1, A2, B2, ... we proceed as follows:

A $\longrightarrow$ B: K1(K2(A1))

B $\longrightarrow$ A: K1(K3(B1)) where K3 = K2(A1)

A $\longrightarrow$ B: K1(K4(A2)) where K4 = K3(B1)

In effect, the intermediate product of each double encryption is used as the second key in the following encryption. In this way each block doubles as an authenticator of all the previous messages in the protocol and information can be exchanged with maximum efficiency.

This is because one normally includes redundancy within each message, in the form of a fixed pattern or an already known variable, in order to check that the encryption has been performed with the right key. As we had a security parameter which dictated four bytes of redundancy, and a communications protocol which exchanged eight byte ciphertext blocks, a naive design would have resulted in half of each message being redundant. However, by key chaining we need only have one redundant data block, namely in the last message (which is the one which causes value to be credited in the recipient card).

During the crypto development process we had been using the Burrows-Abadi-Needham (BAN) logic to check the correctness of the authentication structure, and indeed it proved its value by highlighting several subtle errors and redundant fields in the first draft of our protocol specification. We found, however, that while the BAN logic supports conventional block chaining operations, it cannot deal with key chaining, although this seems to be just as good as data block chaining as a means of accumulating information in a cryptographic variable.

We will illustrate this by the exchange which takes place between a customer card C and a retailer card R when goods are purchased. The other transactions, whether bank - customer or retailer - bank, are essentially similar, but each transaction type uses different keys to ensure that no splicing attack can succeed with more than the probability of a random transaction.

Let C be the customer's name, N_C a nonce generated by him (a random number), R the retailer's name, N_R a nonce generated by him (the transaction sequence number), and X the electronic cheque. Then we can idealise the purchase transaction as:

1. $C \longrightarrow R : \{C, N_C\}_K \quad (= L)$

2. $R \longrightarrow C : \{R, N_R\}_L \quad (= M)$

3. $C \longrightarrow R : \{X\}_M$

In this protocol, the client card debits itself after step 2, while the retailer card only credits itself after step 4. The system is therefore failsafe from the bank's point of view: if anyone tampers with the protocol the only result they are likely to achieve is to increase the bank's float, by debiting the customer card without crediting any retailer.

In order to see how such a protocol can be validated, let us first consider a simplified protocol where the infomation is accumulated without chaining.

1*. $C \longrightarrow R : \{C, N_C\}_K$

2*. $R \longrightarrow C : \{R, N_R, C, N_C,\}_K$

3*. $C \longrightarrow R : \{C, N_C, R, N_R, X\}_K$

This can be analysed in a straightforward way using BAN. The trick is to start from the desired result and work backwards; in this case, we wish to prove that the retailer should trust the cheque, ie $R \mid\equiv X$. This will follow under the jurisdiction rule from $R \mid\equiv C \mid\Rightarrow X$ (R believes C has jurisdiction over X) and $R \mid\equiv C \mid\equiv X$ (R believes C believes X).

The former condition follows from the hardware constraint, that no-one except C could have uttered a text of the form $\{C, ...\}_K$. In effect, we have self-authenticating hardware.

The latter, that $R \mid\equiv C \mid\equiv X$, must be deduced using the nonce verification rule from $\sharp X$ (X is fresh) and $R \mid\equiv C \mid\sim X$ (R believes C uttered X).

Now $\sharp X$ follows from its occurrence in $\{C, N_C, R, N_R, X\}_K$ which contains the sequence number N_R, while $R \mid\equiv C \mid\sim X$ follows from the hardware constraint.

The BAN logic turns out to be easy to use because of the sparsity of its inference rules. When working back from a goal statement, it is rare for there to be more than one possible way to proceed. However, it provides no mechanism for dealing with the key chaining used in the actual protocol. In effect, we have to find a way of unravelling $\{X\}_{\{R,N_R\}_{\{C,N_C\}_K}}$ to $\{C, N_C, R, N_R, X\}_K$.

During the design of UEPS, we solved this problem by adding a further postulate to the BAN schema. The existing message meaning rule says that if P believes that the key K is shared with Q and sees a message X encrypted under K ($P \mid\equiv Q \leftrightarrow^K P, P \triangleleft \{X\}_K$), then he will believe that Q once said X ($P \mid\equiv Q \mid\sim X$).

To this we added a symmetrical rule to the effect that if P tries a key K to decrypt a block, and recognises the result as coming from Q ($P \mid\equiv Q \leftrightarrow^X P, P \triangleleft \{X\}_K$), then he will believe that Q in fact used K ($P \mid\equiv Q \mid\sim K$).

This however conflates recognition of messages with that of keys, and it has since been suggested that we might rather use the existing extension of the BAN logic by Gong, Needham and Yahalom [GNY] and Gong [G], which formalises the concept of recognisability. It turns out that axioms **F2** and **F7** in GNY (**F4** and **F14** in the later work by Gong) together imply a strange result: that someone who possesses a recognisable text X and a fresh key K may conclude that X is fresh by deducing first that $\{X\}_K$ is fresh, and then that X is.

The original BAN logic left the issue of freshness rules rather vague. On page eight it is stated that if (X, Y) is fresh, then X is; in other words, A can 'freshen' a statement X by concatenating it with a nonce just received from B, encrypting it with a key shared with B, and sending it to him.

Two questions follow: firstly, could we not analyse UEPS more easily by adding an extra freshness rule, which allows A to freshen X by encrypting it with a function of a nonce and a key? Secondly, is the word 'fresh' not misleading, in that we should rather concentrate on the action of uttering a statement?

GNY tries to make explicit many things that are left to 'common sense' in BAN, such as spotting redundancy in a decrypted message. It would seem that this refinement, if desirable, has not been thorough enough, and it may have to be extended further to distinguish the utterances of different parties. After all, the utility of 'freshness' is the knowledge it gives us of others' recent utterances.

However, there are good reasons to prefer a small set of rules. We found it more tedious to work with GNY than with BAN, as there are many more rules which have to be considered at each stage; logics which reason about belief and implication in the same calculus may fall foul of the transitivity paradoxes [H]; and finally, as noted above, a complex logic can have unforeseen consequences of a more mundane kind.

Another extension of BAN has been proposed by Kailar and Gligor [KG], who look at a sequence of messages M_i in the context of analysing a multiparty protocol.

This points to another possible approach: linked lists. It is likely that many protocols will prevent splicing attacks by some kind of chaining, so that message M_i contains a hash of message M_{i-1}. Authentication logics should be able to cope with this; a rule that in the linked list $\{A; B; C; D\}$ the freshness of utterance B would imply that of D (but not of A) would seem be right, and this should be a rule for both block and key chaining.

Our work suggests that there are two practical problems for future research in this field. Firstly, what granularity is desirable in our formal calculus; how much should be proved, and what should be left to common sense? Secondly, for a given granularity, what is the minimal effective set of postulates?

To return now to UEPS, we find that the validation, however it is performed, shows that the customer does not receive any confirmation of the transaction, but merely the knowledge that a valid retailer card is present. The value of this knowledge is to rule out a denial-of-service attack by a false terminal; but if the client bank is prepared to tolerate such attacks, then the first message of the protocol could be omitted.

One could also add a confirmation message from the retailer's card, but this would not solve the problem of an interruption after step 2, at which time the customer card has already committed to the transaction and debited itself, while the retailer has still not got the cheque.

As we have seen, no financial benefit can be gained by doing this, and accidents are sufficiently unlikely that they can be dealt with manually. The procedure is to refund to the customer any missing amount which remains unbanked after 21 days; but if the money were to be banked, the dispute would be resolved by comparing the two card records, or inspecting the paper tally roll printed by the merchant terminal (which shows the transaction plus a MAC). No dispute has been reported to date.

6 Practical Aspects

The average float of about ten days has turned out to be sufficient to cover the capital costs, so UEPS has funded itself out of cash flow. It offers the bank the same level of information and control as on a cheque account, as the clearing system tracks the daily balance of every participant. It gives value when the cheque is presented by the merchant, which is often one day after the purchase, and thus about two days faster than with a paper cheque; this does not lead to public complaints as the charges are an order of magnitude less than with the paper cheque system.

No losses were recorded during the first year of operation. The client institution considers it to be a success and the rate of both customer and merchant enrolment is being rapidly increased. From the customers' point of view, it makes fast, secure and low cost payment services available everywhere.

7 Conclusions

The BAN family of logics are not restricted to verifying mutual authentication and key exchange, but are a practical and helpful tool in the design of working crypto protocols, even (and especially) for substantial and complex systems. Although more elaborate systems (like GNY) exist, a first validation should be carried out with BAN, as it is easy to do, and failure to establish a desired result will indicate either a bug in the protocol, or something which BAN cannot express. It will normally be obvious what to do next.

We believe that formal verification should be routine in the design of financial and other security critical systems. It may well not be practical to verify all of a large banking software suite, and some aspects of system security (such as the disaster recovery plan) may remain forever beyond the reach of a formal calculus, but great benefit can be achieved for a small amount of effort by performing a formal analysis of the kernel of the system.

Even within the context of that kernel, we found the BAN logic to be useful as a design discipline. It did much more than help us tighten up the protocol security: it pointed out redundant fields in the messages, allowing the protocol to be made more efficient; it made clear the security dependencies; it provided intellectual stimulation at meetings with designers and programmers who were forced to examine and defend their assumptions; and finally, it greatly strengthened the client's confidence in the system.

8 Acknowledgements

The author is grateful to Roger Needham of the Computer Laboratory, to Serge Belamant of Net1 International, Argèles sur Mer, France, and to André Mansvelt and Gavin Shenker of Net1 Products, Johannesburg, for many stimulating discussions concerning this project.

References

[BAN] M. Burrows, M. Abadi and R. Needham, "A logic of Authentication", Report **39**, Digital Systems Research Center, Palo Alto, Ca.

[C] D. Chaum, "Achieving Electronic Privacy", in *Scientific American*, **267** no 2, August 1992, pp 76 - 81

[DQ] Y. Desmedt and J.-J. Quisquater, "Public-key Systems Based on the Difficulty of Tampering', in *Advances in Cryptology - CRYPTO 86*, Springer Lecture Notes in Computer Science **263** pp 111 - 117

[G] L. Gong, *Cryptographic Protocols for Distributed Systems* (PhD Thesis), University of Cambridge 1990.

[GKSG] V. D. Gligor, R. Kailar, S. Stubblebine and L. Gong, "Logics for Cryptographic Protocols - Virtues and Limitations", in *Proceedings, Computer Security Foundations Workshop* **IV**, IEEE 1991, pp 219 - 226

[GNY] L. Gong, R. M. Needham and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols", in *Proceedings of the 1990 IEEE Computer Security Symposium on Research in Security and Privacy*, pp 234 - 248

[GO] G. Garon and R. Outerbridge, "DES Watch: An Examination of the Sufficiency of the Data Encryption Standard for Financial Institution Information Security in the 1990's', in *Cryptologia* **XV** no 3, July 1991, pp 177-193

[H] M. Hesse, *Structure of Scientific Inference*, Macmillan 1974, pp 142 - 146

[KG] R. Kailar and V. D. Gligor, "On Belief Evolution in Authentication Protocols", in *Proceedings, Computer Security Foundations Workshop* **IV**, IEEE 1991, pp 103 - 116