The Formal Verification of a Payment System

Ross J Anderson

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

Abstract. We describe what we believe was the first use of formal methods to verify a bank payment system. This was an electronic purse for offline small-to-medium value payments, and has since developed into the VISA COPAC product. We verified it using a variant of the BAN logic.

1 Introduction

UEPS, the Universal Electronic Payment System, was an electronic funds transfer product designed for use in developing countries, where poor telecommunications make offline operation necessary. It was built around smartcard based electronic wallet 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 [1].

This architecture is very demanding from the point of view of security, and its protection involves a combination of factors such as the tamper-resistance of the smartcards used; a back-end accounting system that settles transactions reported by customers and merchants within a few days and can thus detect imbalances resulting from card alteration or forgery; and a fall-back processing mode in which, even if the electronic security systems are penetrated, the cards can still be used under the existing system of bank card controls (such as merchant floor limits and hot card lists).

Recently, tampering has been much in the news. In a recent paper [2], we showed how many common smartcards and security processors could be broken using a variety of attacks. However, many of the most devastating attacks did not involve any physical penetration of the device, but rather a failure of the protocol that the device implemented. This is by no means a new phenomenon; protocol failures have been known for many years, and yet they still regularly cause failures in banking systems and elsewhere.

For example, one British bank adopted a system whereby customers' Personal Identification Numbers (PINs) were encrypted and written on the magnetic strip of their debit cards. The idea was to support offline operation and thus improve customer service and cut costs. However, criminals found that they could replace the account number on their own card (whose PIN they knew) with a number found, for example, on a discarded cash machine slip; they could then draw money from that account [3]. The failure of course was that the account number should have been encrypted along with the PIN, and VISA protocols now support this [16].

In another case, a prepayment electricity meter system used tokens to convey an enciphered value of electricity from a shop to a meter in the customer's home; however, the cryptography did not protect the electricity tariff, and tokens could be produced with a tariff of a fraction of a cent per kilowatt hour [4]. Very recently, a cinema ticket vending system was broken by organised criminals; the system designers had arranged for the smartcard issued to the customer to check that the vending machine was authentic, but not vice versa. Many other systems have fallen to protocol attacks, and an introductory article on the subject can be found in [5].

As a result, researchers began to appreciate that unaided human intuition was not adequate when designing such protocols, and began to look for a better way. The seminal work of Burrows, Abadi and Needham in 1989 provided a 'logic of authentication'; this enables one to reason formally about the beliefs of the participants in a security protocol, and follow the chain of reasoning that establishes the genuineness and the freshness of authentication messages (or fails to) [6]. We will describe this logic in more detail below.

As far as we are aware, UEPS is the first live financial system whose underlying protocol suite was designed and verified using such formal techniques. We used an extension of the BAN logic, and our work raised some interesting scientific questions. Firstly, logics like BAN had been thought limited in scope to verifying mutual authentication or key sharing (e.g., by [9]); we showed that this was not so. Secondly, we found a bug in another extension of BAN [8]. Thirdly, we highlighted the need for a formalism to deal with cryptographic chaining. Fourthly, this type of formal analysis turns out to be so useful and indeed straightforward that we argued it should be routine 'due diligence' for financial and security critical systems. Our results were first published in [1]; in this chapter, we have updated them and presented them in a manner aimed at the general formal methods community rather than at security and banking system specialists.

2 The Application

Developed countries have many sophisticated networks which cater for cash 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.

The opening up of the formerly centrally planned economies of Eastern Europe, India, Latin America and Africa created a sudden demand for modern banks and their associated payment systems. However, telecommunications are a serious problem: decades of neglect had 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 modem communications: manual exchanges are still widespread. 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 [16] cannot be used.

However, the flip side of this problem was an 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 might not only integrate retail banking and shopping functions but also provide the payment side of utility and government networks. Slashing transaction costs by an order of magnitude could promote economic growth and eliminate a major bottleneck in economic development.

This was the opportunity perceived by our client, the Net 1 group, which secured a contract from a building society in South Africa (now part of Nedcor Bank) to design and build a national eftpos system. This institution had the largest card base in the country with some 22% of the market, and most of its accounts were simple savings accounts. It had the largest base of black customers of any financial institution in the country, and with political change, plans were made to provide a full range of banking services.

The poor telecommunications in homelands and townships, plus the requirement to keep costs down, led naturally to an electronic wallet approach. Money is transferred between bank cards, customer cards and merchant cards using offline terminals, which can be made portable if necessary.

3 Design Philosophy

The security of UEPS is based on two levels of authentication. The basic payment instrument 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 authentication codes: 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 to the card before it is supplied to the bank. The latter code is checked before funds are credited to the retailer presenting the cheque, while the former is only checked in the event of a dispute. Both these codes are calculated using the standard banking encryption algorithm (ANSI DES MAC [15]) on amount, payee and date.

Had public key technology been available in low cost smartcards, it would have been possible for the merchant to check a digital signature generated on the cheque by the customer's smartcard. This was not an option at the time, and so we had to design a transaction protocol to minimise the likelihood that bad cheques would get into the system. This uses a challenge-response protocol by which both cards in any transaction verify each other and carry on a secure session. In effect, the merchant trusts an application that runs in the customer smartcard (but is hopefully beyond most customers' ability to tamper with); this vouches for the authenticity of the electronic cheque. (The merchant is not trusted with the clearing house's key or the bank's key, and so cannot verify the cheque's authentication code directly.)

Similar transaction protocols are employed between the bank and the customer, and between the customer and the clearing house, and the use of independent security mechanisms (authentication codes and challenge response protocols) enables us to meet resilience requirements: the whole system should not be exposed to fraud through the compromise of any one key or device.

That is the theory, However, in reality there is always the possibility that a design blunder might leave open a hole that an attacker could discover by chance and exploit opportunistically. Most actual banking system failures have happened in this way [3]. A decision was therefore taken to use formal methods to verify the most security critical parts of the system. The choice of the BAN logic was a matter of circumstance; we had just become aware of it, and thought that it might be suitable for the job.

4 Authentication Protocols

In order to make the verification of the UEPS protocol easy to understand, we will digress to describe basic authentication protocols and the BAN logic.

A typical application for an authentication protocol is when two principals, whom we will conventionally call Alice and Bob, wish to communicate privately for the first time. They not share a key with each other, but each of them shares a key with some third party, conventionally called Sam, who will act as an introducer. They might proceed as follows:

1. Alice first calls Sam and asks for a key for communicating with Bob.
2. Sam responds by sending Alice a pair of certificates. Each contains a copy of a key, the first encrypted so only Alice can read it, and the second encrypted so only Bob can read it.
3. Alice then calls Bob and presents the second certificate as her introduction. Each of them decrypts the appropriate certificate under the key they share with Sam and thereby gets access to the new key.

We will simplify the discussion by ignoring the encryption algorithm details and using the following notation: if P is the plaintext and K the key, then the ciphertext is given by:

$$C = \{P\}_K$$

We will assume that this gives integrity as well as secrecy — cut-and-paste attacks on encrypted blocks do not work. (One may think of the encryption operation as a variable-length block cipher with the property that any change to the plaintext will cause a change to the ciphertext, and vice versa, that is completely unpredictable to anyone who does not know the key.)

We will also use the notation N_A to mean a 'nonce' (number used once) generated by Alice. Such nonces may be random numbers or serial numbers, depending on the application; they are used in cryptographic protocols to help ensure that messages are fresh.

With this notation, we can describe one of the first authentication protocols to be published, by Needham and Schroder in 1978. It formalises the protocol that we described in general terms above.

$$
\begin{aligned}
&A \longrightarrow S : A, B, N_A \\
&S \longrightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
&A \longrightarrow B : \{K_{AB}, A\}_{K_{BS}} \\
&B \longrightarrow A : \{N_B\}_{K_{AB}} \\
&A \longrightarrow B : \{N_B - 1\}_{K_{AB}}
\end{aligned}
$$

Here Alice takes the initiative, and tells Sam that she wants to talk to Bob. Sam provides her with a session key, and with a certificate to convey this key to Bob. She passes it to him, and he then does a challenge-response to check that she is present and active.

There is one flaw in this protocol — Bob has no way of knowing that the key K_{AB} from Sam is fresh: Alice could have waited a year between steps 2 and 3. In many applications this may not be important; but if Charlie ever got hold of Alice's key, he could use this to set up session keys with many other principals, and could continue to use these even after Alice had set up a new K_{AS} with Sam. In other words, revocation is problematic.

The majority of crypto protocols published in the almost twenty years since then have turned out to have some such subtle flaw. This is quite remarkable when one considers that an authentication protocol is just a program of typically 3–5 lines and that one would expect to get such a small program right after starting at it for a while. Anyway, as mentioned above, the persistence of such protocol flaws led to the development of the BAN logic, which we will now describe briefly.

5 The BAN Logic

The BAN logic provides a formal method for reasoning about the beliefs of principals in cryptographic protocols. Its underlying idea is that we will believe that a message is authentic if it is encrypted with a relevant key and it is also fresh (that is, generated during the current run of the protocol). This is formalised using a notation which includes:

$A \mid\equiv X$ *A believes X*, or more accurately that A is entitled to believe X;

$A \mid\sim X$ *A once said X* (without implying that this utterance was recent or not);

$A \mid\Rightarrow X$ *A has jurisdiction over X*, in other words A is the authority on X and is to be trusted on it;

$A \triangleleft X$ *A sees X*, i.e. someone sent a message to A containing X in such a way that he can read and repeat it;

$\sharp X$ *X is fresh*, that is, contains a current timestamp or some information showing that it was uttered by the relevant principal during the current run of the protocol;

$\{X\}_K$ *X encrypted under the key K* — a notation we explained above;

$A \leftrightarrow^K B$ *A and B share the key K*, in other words it is an appropriate key for them to communicate.

There are further symbols dealing, for example, with public key operations and passwords, that need not concern us here; the reader should consult [6] for the full description.

These symbols are manipulated using a set of postulates which include:

the message meaning rule states that if A sees a message encrypted under K, and K is a good key for communicating with B, then he will believe that the message was once said by B (we assume that each principle can recognise and ignore his own messages). Formally, $\dfrac{A \mid\equiv A \leftrightarrow^K B, A \triangleleft \{X\}_K}{A \mid\equiv B \mid\sim X}$

the nonce-verification rule states that if a principal once said a message, and the message is fresh, then that principal still believes it. Formally, $\dfrac{A \mid\equiv \sharp X, A \mid\equiv B \mid\sim X}{B \mid\equiv X}$

the jurisdiction rule states that if a principal believes something, and is an authority on the matter, then he should be believed. Formally, we write that $\dfrac{A \mid\equiv B \mid\Rightarrow X, A \mid\equiv B \mid\equiv X}{A \mid\equiv X}$

There are a number of further manipulation rules for dealing with concatenated statements. These are as one would expect; For example, if A sees a statement then he sees its components provided he knows the necessary keys, and if part of a formula is known to be fresh, then the whole formula must be.

Given this machinery, we can now set about attempting to verify the soundness of cryptographic protocols. Very often, we get stuck; and given the sparsity of the BAN postulates, it is usually pretty clear what is missing when this happens. For example, in the Needham-Schroder protocol referred to above, it is straightforward for A and B to deduce that K_{AB} is a key that they share (S being an authority on this) but it is necessary to add the assumption that B accepts the key as new in order to show that it is a good key. This 'hole' in the chain of trust makes the replay attack on Needham Schroder obvious; again, the interested reader is referred to [6] for full details.

6 The UEPS 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.

Our first task, in view of our requirement for double encryption, was to implement a way of doing this which satisfied the security requirements of the client within the technical constraints of the card. In view of growing bank concern about the strength of DES [11], we had to use double encryption; we decided to combine this with key chaining in order to get a compact implementation. 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:

$A \longrightarrow B : \{\{A1\}_{K2}\}_{K1}$
$B \longrightarrow A : \{\{B1\}_{K3}\}_{K1}$ where $K3 = \{A1\}_{K2}$
$A \longrightarrow B : \{\{A2\}_{K4}\}_{K1}$ where $K4 = \{B1\}_{K3}$

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 efficiently.

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 our security requirements dictated four bytes of redundancy, and our communications protocol (dictated by the limitations of the available terminal equipment) 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 development process we had been using the BAN logic to check the correctness of the authentication structure, and it proved its value by highlighting several subtle errors and redundancies in the first draft of our protocol specification (the proof was a constantly evolving multicolour document covering several whiteboards in the Net 1 boardroom). We found, however, that while the BAN logic supports conventional block chaining operations, it cannot deal with key chaining.

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 to and retailer to, are essentially similar, but each transaction type uses different keys to prevent splicing attacks (e.g., in which a customer card is manipulated in order to appear to be a merchant card).

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:

$$
\begin{array}{ll}
C \longrightarrow R : \{C, N_C\}_K & (= L) \\
R \longrightarrow C : \{R, N_R\}_L & (= M) \\
C \longrightarrow R : \{X\}_M &
\end{array}
$$

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 information is accumulated without chaining.

$$
\begin{array}{l}
C \longrightarrow R : \{C, N_C\}_K \\
R \longrightarrow C : \{R, N_R, C, N_C, \}_K \\
C \longrightarrow R : \{C, N_C, R, N_R, X\}_K
\end{array}
$$

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 $R \mid\equiv X$ (the syntax of cheques and cryptographic keys is similar for our purposes here; a cheque is good if and only if it is genuine and fresh).

Now $R \mid\equiv X$ 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 $\{C, ...\}_K$. 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). $\sharp X$ 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. 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 ($P \mid\equiv Q \mid\sim K$).

It has since been suggested that we might rather use the existing extension of the BAN logic by Gong and others [8, 10], which formalises recognisability in a different way. However, there always remains the nagging doubt that logics which reason about belief and implication in the same calculus may fall foul of the transitivity paradoxes [12]; and in the specific case of that particular BAN extension, we found a significant bug. It turns out that axioms F4 and F14 (in [8]: F2 and F7 in [10]) 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. This is clearly undesirable and we prefer our original single-postulate extension of BAN.

Quite apart from the soundness aspects, there are pragmatic reasons to prefer a small set of rules: we found it more tedious to work with the large extension of BAN defined in [8, 10], as there are many more rules which have to be considered at each stage. It is much preferable to use a lightweight, specialised tool where possible; a more complex, general purpose tool would probably have to be supported by proof checking software. While some useful work in this direction has been done (e.g., [14]), a system that can be learned in an hour or less and deployed effectively without mechanical help will remain more attractive to the working engineer.

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 rare 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 was reported during the pilot.

7 Conclusions

The UEPS system was a commercial success. During the pilot stage, the average float of about ten days turned out to be sufficient to cover the capital costs. It offered the bank the same level of information and control as on a cheque account, but cleared two days sooner and cost about a tenth as much to run (these savings were largely passed on to the customer). No fraud losses were recorded in the first year of the pilot period, and the pilot was extended to the other three major banking groups in South Africa. The system was then sold in Namibia and Russia; and finally in 1996 it was adopted by VISA as the COPAC electronic purse. It is expected to be rolled out in Latin American and other parts of the world as a fully supported VISA product during 1998 [7].

It may owe this success in some small part to the fact that we used formal methods to verify its prototype. This may have helped prevent its running into the same kind of opposition and reservations on security grounds as, for example, Mondex [13] (it may be of note that Mondex have just hired one of our recent formal methods PhDs).

Quite apart from the commercial effects that our exercise may have had on the product and on the repute of formal methods within the banking industry, there were scientific benefits. We have showed that the BAN family of logics is not restricted to verifying mutual authentication and key exchange, but are a practical and helpful tool in the design of working crypto protocols, even for substantial and complex systems. Once a modicum of familiarity has been obtained with their use, this is straightforward; so much so that a failure to verify a cryptographic protocol using BAN or one of its competitors could well be construed as a failure of due diligence.

The BAN logic was also useful as a general design discipline. In addition to helping us tighten up the security aspects, 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

A number of people contributed to this work in various ways, including Roger Needham, who introduced me to the BAN logic; Serge Belamant, André Mansvelt and Gavin Shenker of Net1, who supplied the problem; and a number of protocol researchers including Cathy Meadows, Ralf Hauser and the members of the Cambridge University Computer Laboratory who provided useful feedback following the first publication of this work in [1].

References

1. RJ Anderson, "UEPS - a Second Generation Electronic Wallet" in *Computer Security — ESORICS 92*, Springer LNCS vol 648 pp 411–418
2. RJ Anderson, MG Kuhn, "Tamper Resistance — a Cautionary Note", in *Proceedings of the Second Usenix Workshop on Electronic Commerce* (Nov 96) pp 1–11
3. RJ Anderson, "Why Cryptosystems Fail" in *Communications of the ACM* vol 37 no 11 (November 1994) pp 32–40

4. RJ Anderson, SJ Bezuidenhoudt, "On the Reliability of Electronic Payment Systems", in *IEEE Transactions on Software Engineering* v 22 no 5 (May 1996) pp 294–301
5. RJ Anderson, RM Needham, "Programming Satan's Computer", in *'Computer Science Today'*, commemorative issue of Springer Lecture Notes in Computer Science (vol 1000) pp 426–441
6. M Burrows, M Abadi, RM Needham, "A Logic of Authentication", in *Proceedings of the Royal Society of London A* v 426 (1989) pp 233–271
7. "Pre-authorised, off-line debit card launched", in *Card World Independent* (Sep 96) pp 1–2
8. L Gong, *Cryptographic Protocols for Distributed Systems* (PhD Thesis), University of Cambridge 1990
9. VD Gligor, R Kailar, S Stubblebine, L Gong, "Logics for Cryptographic Protocols — Virtues and Limitations", in *Proceedings, Computer Security Foundations Workshop* no IV, IEEE 1991, pp 219–226
10. L Gong, RM Needham, 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
11. G Garon, 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
12. M Hesse, *Structure of Scientific Inference*, Macmillan 1974, pp 142–146
13. R Martin, "Mondex: The way forward?", in *Cards International* no 104 (24/2/94) p 9
14. LC Paulson, "Proving Properties of Security Protocols by Induction", in *10th Computer Security Foundations Workshop* (in press); also Report 409, Cambridge University Computer Laboratory (1996)
15. *Applied Cryptography*, B Schneier, John Wiley and Sons, 1996
16. *'VISA Security Module Operations Manual'*, VISA, 1986