Journal of Automated Reasoning (2006) 36: 5–37
DOI: 10.1007/s10817-005-9018-6

Verifying the SET Purchase Protocols

GIAMPAOLO BELLA[1], FABIO MASSACCI[2] and LAWRENCE C. PAULSON[3]
[1]*Dipartimento di Matematica e Informatica, Università di Catania, Viale A. Doria 6, I-95125 Catania, Italy. e-mail: giamp@dmi.unict.it*
[2]*Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Via Sommarive 14, I-38050 Povo (Trento), Italy. e-mail: fabio.massacci@unitn.it*
[3]*Computer Laboratory, University of Cambridge, J J Thomson Avenue, Cambridge CB3 0FD, UK. e-mail: lcp@cl.cam.ac.uk*

(Published online: 25 February 2006)

Abstract. SET (Secure Electronic Transaction) is a suite of protocols proposed by a consortium of credit card companies and software corporations to secure e-commerce transactions. The Purchase part of the suite is intended to guarantee the integrity and authenticity of the payment transaction while keeping the Cardholder's account details secret from the Merchant and his choice of goods secret from the Bank. This paper details the first verification results for the complete Purchase protocols of SET. Using Isabelle and the inductive method, we show that their primary goal is indeed met. However, a lack of explicitness in the dual signature makes some agreement properties fail: it is impossible to prove that the Cardholder meant to send his credit card details to the very payment gateway that receives them. A major effort in the verification went into digesting the SET documentation to produce a realistic model. The protocol's complexity and size make verification difficult, compared with other protocols. However, our effort has yielded significant insights.

Key words: electronic commerce, security protocols, inductive definitions, deductive verification, Isabelle.

1. Introduction

Recent years have seen substantial progress in the formal verification of security protocols. Detailed analysis of cryptographic primitives, verification of Internet standards, and substantial progress in the automation of model checking and new automatic verification tools have boosted a field that outsiders believe populated by "Yet-Another-Look-at-Needham-Schroeder" papers. The Internet Key Exchange protocol [23], the Cybercash protocol [18], and TLS (the latest version of SSL) [31] have all yielded to automatic or semi-automatic tools. The full analysis of SET – the Secure Electronic Transaction protocol, developed by Visa and Mastercard – has proved to be the toughest challenge.

1.1. THE CHALLENGES OF SET

Why is SET such a challenge for formal verification? The first hurdle is the sheer size of the documentation [19–22], which amounts to over 1,000 pages. The

second, more substantial obstacle is the protocol's complexity. Academic protocols are typically short, almost straight-line programs; they seldom go beyond two levels of encryption and generate few secrets. Internet protocols such as IKE and TLS use cryptography rather sparingly compared to SET. SET has many features that make its verification hard:

- *Multiple-nested encryptions and duplicate message fields* require abbreviations. Most proof tools must expand abbreviations in order to reason about them, but for SET this yields huge expressions.
- *Ubiquitous generation of random numbers and keys* can cause a state-space explosion in finite state methods. The standard model-checking technique of allowing only a handful of nonces and keys would not even allow a single execution to complete, let alone two or more parallel ones.
- *Many other protocol paths* make it hard to single out the few key roles used either by manual analysis (as in the strand space model) or by model checkers to restrict the search space.

SET's use of alternative protocol paths is not bad design but is driven by real requirements. For example, security-aware Cardholders may have preregistered with a financial institution and thus secured their credit cards against the Merchant's eyes. Other Cardholders may decide to trust the Merchant and thus be content with a transaction secured against the outside world. From a Merchant's perspective, all Cardholders should be able to conclude a purchase, whether they bothered to preregister or not.

The complex structure of SET makes it a benchmark for security protocol design and verification, whether or not it is a commercial success. For example, digital envelopes [33] are used in all practical public key protocols such as PGP, and understanding what formal guarantees they offer is vital. If verification techniques are ever to have industrial applicability, then they must be evaluated on major protocols such as SET.

1.2. OUR CONTRIBUTION

We have devoted approximately 6 man-years to a research project on verifying the SET protocol using the inductive method and the Isabelle proof assistant. In previous papers, we have published an overview of our project [7], brief [8] and detailed [6] analyses of SET's Registration protocols, and a brief analysis of SET's Purchase protocols [5]. The present paper completes the presentation of the project with a full, detailed analysis of SET's Purchase protocols.

We have followed the guidelines set out in the previous papers for a careful simplification of SET to make its analysis tractable (the full protocol has hundreds of fields) while retaining the most important mechanisms. Our simplified version

is still one of the most complex protocols ever to be analyzed formally. Particular attention is devoted to the protocol's key construct: the *dual signature*. This mechanism lets the Cardholder agree the order details with the Merchant while hiding those details from the bank. At the same time, it lets the Cardholder share his credit card details with the bank while hiding them from the Merchant. The purpose is simply privacy: there is no reason for the bank to know what sort of goods the Cardholder buys, any more than there is a need for the Merchant to know the Cardholder's bank details. The dual-signature mechanism can be used in other multiparty protocols, as it does not require complicated group-cryptography.

We found that, on the whole, dual signatures work. Credit card details do remain confidential; all parties can be sure that they are dealing with the same transaction, even if they have only partial information. However, SET omits an important field from the dual signature, violating Abadi and Needham's [2] explicitness principle. So, some guarantees are weaker than they should be – particularly for the Payment Gateway, who is supposed to authorize transactions. It is impossible to prove that the Cardholder intended to share his credit card details with the Payment Gateway who is participating in the protocol to authorize the payment. A Merchant, in cooperation with a Payment Gateway, could cause payment to take place through a different Payment Gateway without the Cardholder's knowledge or consent. Although this scenario is not an attack on SET's main objectives of integrity, authenticity, and privacy, it violates the Cardholder's expectations. A simple change to the protocol can fix this problem.

From a verification perspective, our result shows that the inductive method (supported by a powerful prover such as Isabelle) can scale up to protocols as complex as SET. However, we are near the limit of tractability for our approach. Better automation or user interfaces are needed for more complex protocols or more detailed models of SET.

In the next sections we present an overview of SET (Section 2) and of its Purchase protocols (Section 3). We discuss the formal model, presenting the protocol rules in Isabelle syntax (Section 4). Then we discuss successful and failed proofs (Sections 5 and 6). Changing perspective, we consider the difficulty of the proofs purely as automated reasoning problems (Section 7). The paper continues with a discussion of related work (Section 8) and concludes with some suggestions for future research (Section 9).

2. SET Overview

Most Internet Merchants use the SSL protocol to prevent eavesdroppers from learning Cardholders' account details, adopting the classical idea that bad persons are always outsiders. This arrangement has two major limitations:

– Cardholders must trust Merchants to keep these details secure, when Merchants may be dishonest or incompetent [27].

– Merchants must trust Cardholders, when Cardholders can repudiate their purchases or submit stolen credit card numbers.

Visa and Mastercard designed the SET protocol to solve these problems by keeping sensitive information confidential and by authenticating Cardholders and Merchants to financial institutions and to one another [20, page 6]. To achieve these goals, SET comprises five main subprotocols:

– *Cardholder Registration* allows a Cardholder to register a credit card with a Certification Authority. The request includes the Cardholder's public signature key and a secret nonce. The outcome of registration is a public-key certificate that includes the hash of the credit card number (called the *primary account number* or PAN) and of a secret nonce (PANSecret), with the same role of the PIN for physical cards. This phase is expected to run only once during the enrolment of the Cardholder.
– *Merchant Registration* allows a Merchant to register both a signature key and an encryption key. Once again this phase should be run only once for each credit card brand.
– *Purchase Request* allows Cardholders to place orders with Merchants. Normally, this should take place after both Cardholder and Merchant have registered. The specification leaves this option open for Cardholders who wish to Purchase without having registered a public key.
– *Payment Authorization* follows or is combined with *Purchase Request*. It allows a Merchant to verify the Cardholder's details with a Payment Gateway, which authorizes the transactions.
– *Payment Capture* is used by Merchants for the actual funds transfer at the end of the protocol suite.

Here is the protocol in brief. Cardholders and Merchants should register with Certificate Authorities before making purchases. Known fraudsters may be blocked at this stage. Registered principals can then engage in business. During the Purchase protocols, a combination of digital signatures and hashes allows the Cardholder to make purchases without sharing account details with the Merchant and without sharing order information with the Payment Gateway.

3. The SET Purchase Protocols

Before going into details, let us point out some distinctive features in the design of the Purchase protocols.

The first idea is to use *Digital Envelopes*. Asymmetric encryption is too slow to be used for anything other than key distribution. Symmetric encryption is fast and can handle bulk data, but it is bidirectional and offers no proof of origin. So, if Alice wants to send a long message M to Bob, she generates a symmetric key K, encrypts K using Bob's public key, and encrypts the message M using K.

Some types of digital envelope bundle K with an additional short message m and with the hash of M, or the hash of both m and M, and so on. The intuition is that a digital signature binding the symmetric key encrypting M with the hash of M yields a proof of origin also for M. The SET Books [22] and the PKCS Standard [33] discuss the implementation and the cryptographic security of digital envelopes.

The second idea is the *Dual Signature*. This combination of hashes and digital signatures lets several parties agree on a transaction without giving each of them a complete view of the transaction. It avoids resorting to general cryptographic access structures.

Suppose that Alice wants to sign two documents O (for Order) and P (for Payment) but wants to show to Bob only the O part of the transaction and to Charlie only the P part. Then she sends to Bob O, the hash of P, and the *dual signature*, namely, the signature of the concatenation of the hash of P with the hash of O. Clearly Bob can verify the signature because he has the hash of P and can generate the hash of O. Then, she sends to Charlie P and the hash of O, together with the same signature, which he can verify. Although Charlie does not know what is in O and Bob does not know what is in P, they can agree that Alice signed O and P.

Using a further level of encryption, we can use Bob to forward the message to Charlie: Bob receives O, the hash of P, and the dual signature, plus P encrypted with Charlie's public key (so that Bob cannot read it). Then Bob checks the signature as before and forwards to Charlie the hash of O, the dual signature, and the encrypted part P. Charlie can remove the encryption layer and verify the signature. Still Bob does not know P and Charlie does not know O. In summary, the protocol runs as follows:

1. $A \longrightarrow B$: O, $\mathsf{Hash}(P)$, $\mathsf{Sign}_{\mathsf{priSK}\ A}(\mathsf{Hash}(O), \mathsf{Hash}(P))$, $\mathsf{Crypt}_{\mathsf{pubEK}\ C}(P)$
2. $B \longrightarrow C$: $\mathsf{Hash}(O)$, $\mathsf{Sign}_{\mathsf{priSK}\ A}(\mathsf{Hash}(O), \mathsf{Hash}(P))$, $\mathsf{Crypt}_{\mathsf{pubEK}\ C}(P)$

This protocol does not protect Alice from Bob's or Charlie's misbehaving. She must trust Bob and Charlie not to exchange O and P covertly. SET observes the trust assumptions of the credit card domain. In particular, it assumes that merchants and banks will obey privacy regulations, if only because it is in their commercial interest. Because SET authenticates the parties, anybody who does break the law can be positively identified.

REMARK 1. In the real world, digital envelopes and dual signatures speed the underlying cryptography. In the formal model, they complicate symbolic analysis by introducing duplications.

Since the symbolic hash of M is not shorter than M (formally M is a subterm of $\mathsf{Hash}(M)$), sending the symmetric encryption of M with the asymmetric encryption of the hash of M duplicates the symbolic term describing the message M.

To make the presentation readable – both below and in the formal specifications – we introduce many abbreviations. The messages blow up dramatically when abbreviations are unfolded. Yet, without abbreviations the protocol would be unreadable. Hence, as we have noted already, automated support for handling abbreviations and automatically folding or unfolding them on demand is a must for formal analysis of such large scale protocols.

The Purchase protocols are considerably more complicated than the description given above. They involve interaction among three parties and several different protocol paths. *Purchase Requests* may be signed or unsigned, depending on whether the Cardholder has run the Registration protocol. *Payment Authorization* may be invoked during Purchase Request, or authorizations may be batched for processing later. Other complications include split shipments and payment by instalments.

Here, we combine Purchase Request with Payment Authorization, yielding in effect a six-step protocol.

REMARK 2. For sake of readability, the version presented in this section is simpler even than the one that we modeled and verified in Isabelle: certificates are omitted, and the PKCS digital envelopes [33] are replaced by simple public-key encryption. Readers interested in a realistic model of digital envelopes should consider using the Isabelle model detailed in Section 4.

Reducing the SET Purchase protocols to our formal model has not been trivial. A number of tricky issues in the modeling are discussed elsewhere [8]. Our model provides both signed and unsigned versions of Purchase Request and of the main round of Payment Authorization: Authorization Request and Authorization Response.

3.1. INITIAL SHOPPING AGREEMENT

The Cardholder and Merchant agree on the order description (Order-Desc) and the purchase amount (PurchAmt). This agreement step, called the *SET Initiation Process* in the *Programmer's Guide* [22, page 45], is not part of SET and occurs just before it. There are suggestions in the SET *External Interface Guide* [19], but they are not part of the official protocol: the SET Initiation Process is not defined in the *Formal Protocol Definition*, and the *Programmer's Guide* [22, page 45] expects that "standards will be developed to address how this information is exchanged and how the SET protocol is initiated."

SET's system of transaction identifiers is elaborate. The *Programmer's Guide* states that the Merchant originally identifies the transaction from the identifier LID_M (if the Cardholder sends it) or out of band otherwise [22, page 310]. After that, the parties use a different transaction identifier, XID: "XID is a transaction ID that is usually generated by the Merchant system, unless there is

no [Purchase Initialization Response, Section 3.3], in which case it is generated by the Cardholder system" [22, page 267]. In the latter case, the Merchant identifies the order by scanning the order description according to out-of-band agreements.

We decided to resolve SET's complicated system of transaction identifiers by

- requiring presence of the initial message where the Cardholder sends LID_M, so that
- the Merchant is responsible for generating XID, which is used to identify the current transaction.

Should the Merchant fail to choose a globally unique XID, a dishonest Cardholder could collude with a dishonest Merchant and have a transaction authorized by a Payment Gateway. The dishonest Cardholder could purchase the same goods from an honest Merchant by replaying the Gateway's authorization from the other session. The honest Merchant would commit to sending the goods but receive no payment.

3.2. PURCHASE INITIALIZATION REQUEST

The Cardholder sends the Merchant a freshness challenge (Chall_C) and a local transaction identifier (LID_M).

$$C \rightarrow M : \text{LID\_M}, \text{Chall\_C} \tag{1}$$

3.3. PURCHASE INITIALIZATION RESPONSE

The Merchant replies with a signed message that includes a freshness challenge (Chall_M) and generates a nonce that serves as the globally unique transaction identifier★ XID. Also returned (but omitted below) is the public-key certificate of a Payment Gateway, which is determined by the Merchant's bank and the card brand. In our formalization, a certificate is merely a message containing an agent's name and public key, signed by the Root Certification Authority.

$$M \rightarrow C : \mathsf{Sign}_{\mathsf{priSK}\,M}\big(\text{LID\_M},\ \text{XID},\ \text{Chall\_C},\ \text{Chall\_M}\big) \tag{2}$$

3.4. PURCHASE REQUEST

The Purchase Request is the most interesting message in SET. The Merchant and Payment Gateway must agree on the Cardholder's purchase, although each of

★ "a randomly generated 20 byte variable that is globally unique (statistically)" [22, p. 267].

them gets only partial information: the Merchant should not know the card details, and the Payment Gateway should not know what is being bought. To meet this objective, SET uses a dual signature: the Cardholder signs the concatenation of the hashes of the Payment Instructions PIData and the Order Information OIData. He combines this with the card details PANData, including the PAN and other secret numbers, CardSecret and PANSecret, which help to authenticate him. Then he encrypts everything using the Payment Gateway's public key, $\mathsf{pubEK}\ P$. He sends this to the Merchant, along with the Order Information and the hash of the Payment Instructions.

$$C \rightarrow M : \text{PIDualSign}, \text{OIDualSign} \tag{3}$$

Here, C has computed

$$\begin{aligned}
\text{HOD} &= \mathsf{Hash}(\text{OrderDesc}, \text{PurchAmt}) \\
\text{PIHead} &= \text{LID_M}, \text{XID}, \text{HOD}, \text{PurchAmt}, \text{M}, \mathsf{Hash}(\text{XID}, \text{CardSecret}) \\
\text{OIData} &= \text{LID_M}, \text{XID}, \text{Chall_C}, \text{HOD}, \text{Chall_M} \\
\text{PANData} &= \text{PAN}, \text{PANSecret} \\
\text{PIData} &= \text{PIHead}, \text{PANData} \\
\text{PIDualSign} &= \mathsf{Sign}_{\mathsf{priSK}\ C}(\mathsf{Hash}(\text{PIData}), \mathsf{Hash}(\text{OIData})), \\
&\quad\ \mathsf{Crypt}_{\mathsf{pubEK}\ P}(\text{PIHead}, \mathsf{Hash}(\text{OIData}), \text{PANData}) \\
\text{OIDualSign} &= \text{OIData}, \mathsf{Hash}(\text{PIData})
\end{aligned}$$

An *unsigned* Purchase Request – formally modeled but not shown here – obviously lacks this combination of digital signatures and hashing. It authenticates the Cardholder by using the hash of the PANSecret. Though it does not offer the guarantees of a digital signature in terms of proof of origin, it is still better than sending the credit card details to the Merchant.

3.5. AUTHORIZATION REQUEST

After receiving the Purchase Request, the Merchant seeks authorization from a Payment Gateway. First, he verifies the dual signature, using the hash from the Payment Instructions. He also verifies the Order Information. He takes the Payment Instructions (which he cannot read) and combines them with transaction identifiers and the hash of the Order Information. This combination is signed by the Merchant and then encrypted by using the Payment Gateway's public key.

$$\begin{aligned}
M \rightarrow P : \mathsf{Crypt}_{\mathsf{pubEK}\ P}\big(&\mathsf{Sign}_{\mathsf{priSK}\ M} \\
&\big(\text{LID_M}, \text{XID}, \mathsf{Hash}(\text{OIData}), \text{HOD}, \text{PIDualSign}\big)\big)
\end{aligned} \tag{4}$$

3.6. AUTHORIZATION RESPONSE

The Payment Gateway verifies the dual signature using the hash from the Order Information. Then, he checks that the Cardholder and Merchant agree on the Order Description and Purchase Amount by comparing certain hash values.★ Finally, he verifies the Cardholder's secret account information, using the Cardholder's certificate. If satisfied, he confirms authorization to the Merchant by signing a brief message containing the transaction identifier and purchase amount.

$$P \rightarrow M : \mathsf{Crypt}_{\mathsf{pubEK}\,M}\big(\mathsf{Sign}_{\mathsf{priSK}\,P} \big(\mathrm{LID\_M}, \mathrm{XID}, \mathrm{PurchAmt}, \mathrm{AuthCode}\big)\big) \tag{5}$$

What if authorization is denied? In SET, the Payment Gateway always responds to the inquiries of the Merchant, even when authorization is denied. Thus, the actual authCode field may be a "yes,'' a "no,'' a "contact-human-at-800-SET-HELP," and so forth. For simplicity, our model assumes that principals return only 'yes' answers and otherwise abandon the session. Other researchers might analyze the security of the protocol when both "yes'' and "no'' answers may be returned.

3.7. PURCHASE RESPONSE

The Merchant now sends a similar signed message to the Cardholder. It contains the hash of the Purchase Amount, which the Cardholder can verify. Disputes are resolved out of band.

$$M \rightarrow C : \mathsf{Sign}_{\mathsf{priSK}\,M}\big(\mathrm{LID\_M}, \mathrm{XID}, \mathrm{Chall\_C}, \mathsf{Hash}(\mathrm{PurchAmt})\big) \tag{6}$$

4. The Formal Model

We use the Isabelle proof assistant [26] with the inductive method of protocol verification introduced by Paulson [29] and extended by Bella [9]. The operational semantics assumes an infinite population of agents obeying the protocol and a dishonest agent (the Spy) who can steal messages intended for other agents, decrypt them using any keys at his disposal, and send new messages as he pleases. Some agents are compromised, meaning the Spy has full access to their secrets.

Each agent has two asymmetric key pairs, one for signature and one for encryption. Apart from the Spy, agents are of four kinds:

– Certificate Authorities, which sign certificates for other agents, are written `CA i` (for $i \geq 0$).

★ He compares the HOD he receives by message 4 signed by the Merchant with the one he receives in the dual-signature PIDualSign.

- Cardholders are written *Cardholder i*.
- Merchants are written *Merchant i*.
- Payment Gateways are written *PG i*.

The *Root* Certificate Authority is *CA 0*. It is abbreviated as *RCA*, and the model assumes it to be uncompromised. Any other agents may be under the Spy's control. Protocol properties can usually be expected to hold only if the agents involved are uncompromised, though many compromised agents may be present.

A protocol is modeled by the set of all possible traces of events that it can generate. Events are of three forms:

- *Says A B* means *A* sends message *X* to *B*.
- *Gets A X* means *A* receives★ message *X*
- *Notes A X* means *A* stores *X* in its internal state.

Each protocol step consists of many preconditions (typically referring to previous messages being received or fresh keys being generated) and a postcondition (some new messages are sent or stored).

The Purchase protocols are specified in about 230 lines of Isabelle text, including some comments but excluding the general SET public-key model (which totals nearly 1,700 lines). Unsigned purchases add several rules to the specification, namely, the unsigned Purchase Request itself and its handling by the Merchant and Payment Gateway.

4.1. PROTOCOL RULES FOR INITIATING A PURCHASE

We devoted much thought to modeling the SET Initiation Process (see also Section 3.1). This was essential because to prove that all parties agree on the details of a transaction at the end of a run, we must be precise about what transaction is being made at the start of the run. Two issues were particularly delicate.

The first issue is modeling the transaction identifier. Once the identifier is fixed, one can state claims such as "if the Cardholder signed a transaction identified by *X*, then the Merchant record of the transaction identified by *X* matches with the record of the Cardholder."

REMARK 3. The problem of identifying a transaction in e-commerce protocols is similar to that of of identifying a run in authentication protocols. Identifying a run has been considered tricky in the literature, and a number of papers in the mid-1990s have set out contrasting opinions: for example, Lowe

★ Since all messages can be intercepted and redirected by the intruder that manipulate the network, we ignore the sender field in the received message.

[17] and Gollmann [13]. After over a decade of discussion, it is now accepted that nonces or keys created in an authentication protocol are the run identifiers. Our efforts confirm that the most appropriate identifier for a transaction of SET's Purchase protocols is the Merchant's choice of XID.

The second issue is the choice of the agent responsible for labeling the transaction with the identifier. This choice has major implications for the underlying trust assumptions. If Alice is responsible for assigning transaction identifiers, then all proofs about the authenticity of the transactions must assume that Alice is honest, since otherwise she could simply assign the same identifier to two different transactions.

As remarked above (Section 3.1), we had to

- find a way to bootstrap the SET Initiation Phase by out-of-band means,
- find an identifier for the transaction, and
- find the agent responsible for identifying the transaction.

The first step is less trivial than it seems. If we assume that initiation takes place over the network, it becomes impossible to prove anything: the intruder could have changed the details of the transaction from the outset. In practice, the initiation phase might be protected by a weaker protocol, such as SSL.

We have modeled the bootstrapping step with two simultaneous events that are not sent over the network: one from the Cardholder and one from the Merchant. Each stores LID_M together with the transaction details; the Merchant also stores the Payment Gateway's name. We have chosen *XID* for the identifier and mandated the first two messages (optional in the specification), making the Merchant is responsible for generating *XID* appropriately [22, page 267]. Recall the discussion in Section 3.1 and Section 3.3.

Figure 1 presents three rules of the inductive definition, as they are given to Isabelle, showing how we model the SET Initiation Process and the SET system of transaction identifiers.

The first rule, *Start*, shows how we formally model the out-of-band agreement that we have informally described above. The Cardholder *C* and Merchant *M* somehow agree on a *Transaction* specified by *OrderDesc* and *PurchAmt*. No messages are sent, but both parties simultaneously record the transaction using a *Notes* event each. The rule refers to a given trace, here called *evsStart*. The trace *evsStart* is a possible sequence of events that happened so far. The constant *set_pur* denotes the set of traces belonging to the SET purchase. So, by *evsStart* $\in$ *set_pur* we simply denote the fact that this trace must belong to the set of traces of the protocol. Assumptions of the form *LID_M* $\notin$ *range* ... help Isabelle's simplifier and are acceptable because a transaction identifier can be discerned by its length from the credentials, *CardSecret* and *PANSecret*, that the Cardholder obtained from the Registration protocol [6]. Similar assumptions are made about *Chall_C* and *XID* in the next two rules. The second rule, *PInitReq*, formalizes the actual beginning of the protocol. It

```
Start:
 "⟦evsStart ∈ set_pur;
   Number LID_M ∉ used evsStart;
   C = Cardholder k; M = Merchant i; P = PG j;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   LID_M ∉ range CardSecret;  LID_M ∉ range PANSecret⟧
   ⟹ Notes C {|Number LID_M, Transaction|}
     # Notes M {|Number LID_M, Agent P, Transaction|}
     # evsStart ∈ set_pur"

PInitReq:
 "⟦evsPIReq ∈ set_pur;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   Nonce Chall_C ∉ used evsPIReq;
   Chall_C ∉ range CardSecret; Chall_C ∉ range PANSecret;
   Notes C {|Number LID_M, Transaction |} ∈ set evsPIReq⟧
  ⟹ Says C M {|Number LID_M, Nonce Chall_C|} # evsPIReq ∈ set_pur"

PInitRes:
 "⟦evsPIRes ∈ set_pur;
   Gets M {|Number LID_M, Nonce Chall_C|} ∈ set evsPIRes;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   Notes M {|Number LID_M, Agent P, Transaction|} ∈ set evsPIRes;
   Nonce Chall_M ∉ used evsPIRes;
   Chall_M ∉ range CardSecret; Chall_M ∉ range PANSecret;
   Number XID ∉ used evsPIRes;
   XID ∉ range CardSecret; XID ∉ range PANSecret⟧
  ⟹ Says M C (sign (priSK M)
                  {|Number LID_M, Number XID,
                     Nonce Chall_C, Nonce Chall_M,
                     cert P (pubEK P) onlyEnc (priSK RCA)|})
       # evsPIRes ∈ set_pur"
```

Figure 1. Start, initialization request/response in Isabelle syntax.

sees the Cardholder simply send *LID_M* paired with a nonce challenge. The third rule, *PInitRes*, formalizes the Merchant's response, where he chooses and sends *XID*. The signature primitive has the obvious definition.

$$signKX == \{|X, \quad Crypt\,K\,(Hash\,X)|\}$$

We tried various ways of formalizing the initial bootstrapping phase, and other researchers may make different choices. For example, in a previous version of our model, *XID* was generated by using an uninterpreted injective function from the transaction details. This model simplified many proofs, essentially because it was not possible for a bad agent to use the same identifier for different

transaction. Loosely speaking, the same Order and the same Purchase Amount always yielded the same identifier and the unique identification of the transaction was guaranteed by construction. Yet, even in this simplified model, we could not eliminate from many theorems the assumption that the Merchant was honest.

This disturbing fact led us to a more sophisticated model where responsibility for choices of identifiers and transactions details were made explicit. In the refined model, the reason behind the trust assumption is clearer. For example, the need for trusting the Merchant in theorems about agreement between the Cardholder and Payment Gateway appears in step *Start*: the Merchant is responsible for choosing the Payment Gateway for future protocol steps. This admits the possibility of undesirable collusion with the Payment Gateway – see the later discussion in Section 6 – though opinions differ on whether it is an error on the part of SET's designers.

4.2. THE SIGNED PURCHASE REQUEST

Figure 2 presents part of the rule modeling the signed purchase request. Let us go through it in some detail. The rule refers to a given trace, here called *evsPReqS*. Condition *C = Cardholder k* defines a local abbreviation: *C* stands for the

```
PReqS:
 "⟦evsPReqS ∈ set_pur;
   C = Cardholder k;
   CardSecret k ≠ 0;  Key KC2 ∉ used evsPReqS;  KC2 ∈ symKeys;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   HOD = Hash{|Number OrderDesc, Number PurchAmt|};
   OIData = {|Number LID_M, Number XID, Nonce Chall_C, HOD,
              Nonce Chall_M|};
   PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
              Hash{|Number XID, Nonce (CardSecret k)|}|};
   PANData = {|Pan (pan C), Nonce (PANSecret k)|};
   PIData = {|PIHead, PANData|};
   PIDualSigned = {|sign (priSK C) {|Hash PIData, Hash OIData|},
                    t KC2 EKj {|PIHead, Hash OIData|} PANData|};
   OIDualSigned = {|OIData, Hash PIData|};
   Gets C (sign (priSK M)
                {|Number LID_M, Number XID,
                  Nonce Chall_C, Nonce Chall_M,
                  cert P EKj onlyEnc (priSK RCA)|})
     ∈ set evsPReqS;
   Says C M {|Number LID_M, Nonce Chall_C|} ∈ set evsPReqS;
   Notes C {|Number LID_M, Transaction|} ∈ set evsPReqS⟧
  ⟹ Says C M {|PIDualSigned, OIDualSigned|}
       # Notes C {|Key KC2, Agent M|}
       # evsPReqS ∈ set_pur"
```

Figure 2. Signed purchase request in Isabelle syntax.

*k*th Cardholder. He is the active principal in this rule. Recall that we have no limit to the number of Cardholders. The condition *CardSecret k* ≠ 0 checks that this *k*th Cardholder is registered: the *CardSecret* field belongs to the certificates exchanged by the Merchant and the Cardholder. It is fixed to 0 if the Cardholder did not bother to register his public key with a certification authority. The conditions *Key KC2* ∉ *used evsPReqS* and *KC2* ∈ *symKeys* say that *KC2* is a fresh symmetric key.

The next line, *Transaction* = ... refers to the transaction details agreed by the Cardholder and the Merchant in the *Start* rule. The next several lines, starting with *HOD* and ending with *OIDualSigned*, express the construction of the dual signature. To this purpose, the general SET model defines the *EXcrypt* construct as follows:

$$EXcrypt\ K\ EK\ M\ m == \{|Crypt\ K\ \{|M, Hash\ m|\}, Crypt\ EK\ \{|Key\ K, m|\}|\}$$

Here *EK* is a public encryption key, *K* is a symmetric key, and *M* and *m* are fields – note that *m* appears twice. Next come the rule's preconditions: the Cardholder agreed on the transaction details with the Merchant (the *Notes C* event), he sent the Purchase Initialization Request to the Merchant (the *Says C M* event), and finally he received the Purchase Initialization Response (the *Gets C* event). These steps are those described in the actual SET protocol (see Section 3). Skipping to the conclusion, we find the current trace being extended with a *Says C M* event, whereby *C* sends the dual signature, and a *Notes C* event, whereby he notes down the fresh symmetric key and its recipient.

4.3. PROTOCOL RULES FOR COMPLETING A PURCHASE

Figure 3 presents the Isabelle formalization of the next two messages, which belong to the separate Authorization Request protocol. Let us briefly examine *AuthReq*, in which the Merchant contacts the Payment Gateway. By *Key KM* ∉ *used evsPReqs* and *KM* ∈ *symKeys* we pick a fresh symmetric key. Next, through *Transaction* = ..., we refer to the stored transaction details; then we calculate *HOD* to be the hash of the Order Description and Purchase Amount and calculate the Order Information *evsPReqs* from various known quantities. The *Gets M* event refers to the Merchant's reception of the Purchase Request. If all is well, then the trace is extended with a *Says* event from the Merchant to the Payment Gateway. This event uses the *EncB* primitive, which stands for encapsulation with baggage. It is defined in terms of the simple encapsulation primitive *Enc* as follows:

$$EncB\ SK\ K\ EK\ M\ b == \{|Enc\ SK\ K\ EK\{|M, Hash\ M, b|\}, M, b|\}$$
$$Enc\ SK\ K\ EK\ M == \{|Crypt\ K\ (sign\ SK\ M), Crypt\ EK\ (Key\ K)|\}$$

```
AuthReq:
 "⟦evsAReq ∈ set_pur;
   Key KM ∉ used evsAReq;  KM ∈ symKeys;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   HOD = Hash{|Number OrderDesc, Number PurchAmt|};
   OIData = {|Number LID_M, Number XID, Nonce Chall_C, HOD,
              Nonce Chall_M|};
   CardSecret k ≠ 0 -->
     P_I = {|sign (priSK C) {|HPIData, Hash OIData|}, encPANData|};
   Gets M {|P_I, OIData, HPIData|} ∈ set evsAReq;
   Says M C (sign (priSK M) {|Number LID_M, Number XID,
                              Nonce Chall_C, Nonce Chall_M,
                              cert P EKj onlyEnc (priSK RCA)|})
     ∈ set evsAReq;
      Notes M {|Number LID_M, Agent P, Transaction|}
       ∈ set evsAReq⟧
 ⟹ Says M P
        (EncB (priSK M) KM (pubEK P)
          {|Number LID_M, Number XID, Hash OIData, HOD|}   P_I)
     # evsAReq ∈ set_pur"

AuthResS:
 "⟦evsAResS ∈ set_pur;
   C = Cardholder k;
   Key KP ∉ used evsAResS;  KP ∈ symKeys;
   CardSecret k ≠ 0;  KC2 ∈ symKeys;  KM ∈ symKeys;
   P_I = {|sign (priSK C) {|Hash PIData, HOIData|},
           EXcrypt KC2 (pubEK P) {|PIHead, HOIData|} PANData|};
   PANData = {|Pan (pan C), Nonce (PANSecret k)|};
   PIData = {|PIHead, PANData|};
   PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
              Hash{|Number XID, Nonce (CardSecret k)|}|};
   Gets P (EncB (priSK M) KM (pubEK P)
            {|Number LID_M, Number XID, HOIData, HOD|}
            P_I)
     ∈ set evsAResS⟧
 ⟹ Says P M
       (EncB (priSK P) KP (pubEK M)
         {|Number LID_M, Number XID, Number PurchAmt|}
         authCode)
   # evsAResS ∈ set_pur"
```

Figure 3. Authorization request and response.

Take particular note of the condition

$$CardSecret\ k \neq 0 \longrightarrow$$

$$P\_I = \{|sign(priSK\ C)\{|HPIData,\ Hash\ OIData|\}, encPANData|\};$$

If this Cardholder is registered, then we verify the dual signature by checking that the signed hash $HPIData$ equals the hash sent as the third component of

the Purchase Request, and verify that the second component of the dual signature equals the hash of *OIData*. These essential checks can be neatly expressed without resorting to equality tests in the precondition of the rule: where two fields have to agree, we simply give them the same variable name. The same technique is used to express the Payment Gateway's verification of the dual signature.

Figure 4 contains the Purchase Response sent from the Merchant to the Cardholder. It is very simple, containing the transaction identifiers, Cardholder's nonce challenge, and hashed Purchase Amount, all digitally signed by the Merchant.

During the moding stage, the support for equational reasoning in Isabelle allows one to express messages like Purchase Request succinctly. Unfortunately, Isabelle's simplifier expands equations during proofs, producing subgoals many pages long. Handling such huge formulas requires additional memory and processor time and makes great demands on the human verifier. (See further discussion in Section 9.)

4.4. AN ASIDE ON DIGITAL ENVELOPES

Equations are necessary in our SET formalization because messages have many repeated fields. We have seldom used them before, even for complex protocols such as TLS or Kerberos. A message containing both *M* and *Hash M* involves a repetition of *M* when it is treated formally.

```
PRes:
 "⟦evsPRes ∈ set_pur;  KP ∈ symKeys;  M = Merchant i;
   Transaction = {|Agent M, Agent C, Number OrderDesc,
                   Number PurchAmt|};
   Gets M (EncB (priSK P) KP (pubEK M)
           {|Number LID_M, Number XID, Number PurchAmt|}
           authCode)
      ∈ set evsPRes;
   Gets M {|Number LID_M, Nonce Chall_C|} ∈ set evsPRes;
   Says M P
        (EncB (priSK M) KM (pubEK P)
          {|Number LID_M, Number XID, Hash OIData, HOD|} P_I)
     ∈ set evsPRes;
   Notes M {|Number LID_M, Agent P, Transaction|}
      ∈ set evsPRes⟧
 ⟹ Says M C
     (sign (priSK M) {|Number LID_M, Number XID, Nonce Chall_C,
                       Hash (Number PurchAmt)|})
     # evsPRes ∈ set_pur"
```

Figure 4. Purchase response.

Further repetitions arise from the *EXcrypt* digital envelope, as observed in the previous section. We could simplify this to a simple public-key encryption, as we have done in our informal presentation of the SET Purchase protocols on Section 3. Our first proofs made this very simplification. They were useful for identifying the main guarantees to be verified. However, the simplification resulted in a considerable loss of precision. The digital envelope not only admits the possibility of the symmetric key's being compromised but binds the message components more loosely. If we model the digital envelope by the direct encryption *Crypt EK* {|*M*, *m*|}, then *M* and *m* can only be compromised together. In the model, one can lose the symmetric key, thus disclosing *M* to the spy, without disclosing *m*. This difference in the security level is also witnessed by the difference of importance between *M* and *m* in the actual SET protocol. The message *M*, protected by a symmetric key, merely contains the order information (so what if the spy discovers your bad taste in music?), while *m*, protected by a public key encryption, contains the credit card details (deserving of stronger protection). The current formalization includes full digital envelopes, which makes the proofs considerably more complicated.

5. Verified Properties

The *Formal Protocol Definition* [21] does not formally specify the goals of SET. All we have are the explicit but imprecise requirements from the *Business Description* [20, page 6], which we quote here:

1. Provide confidentiality of payment information
2. Ensure integrity of all transmitted data
3. Provide authentication that a Cardholder is a legitimate user of a branded payment card account
4. Provide authentication that a Merchant can accept branded payment card transactions.

Such goals are of course broader than the ones that can be actually formalized, as they include business notions such as existence of trust relations. At this stage, a modeling decision must be made. We have used our judgment to transform these vague goals into a possible equivalent notion in the formal domain. Other researchers might have chosen different formalizations. We address the first two business requirements as follows:

1. Provide confidentiality of the PAN and the PANSecret between (uncompromised) cardholders and payment gateways, because this is the relevant payment information that remains in our model★

★ For instance, we have dropped the Bank Information, assuming that it is encoded in the PAN.

2. Ensure integrity of all transmitted data that is signed by some uncompromised party

These intermediate requirements are expressed as a number of theorems, which are presented below. Note that the second requirement implies that every message that appears to come from a certain party actually does come from that party, or else that party has been compromised.

Dealing with the last two business goals requires the combination of the results in this paper and our account on the cardholder and merchant registration phases [6]. To provide authentication that a cardholder is a legitimate user of a credit card brand means that there is a chain of trust from the cardholder digital credentials (PAN and signature key) and the credit card Issuer. For merchants, providing evidence that they can accept branded credit cards means that there is chain of trust from their digital credentials (public key for encrypting data sent to them, and private key signing data sent by them) to the corresponding SET authorities.

In the SET registration paper, we showed that Cardholders and Merchant can safely register their keys and that the confidentiality of the PAN coming from the SET Certification Gateway is also protected. If a Cardholder has a valid certificate from a SET Root, then he is a legitimate user of a credit card. Here, we start from the public/private key pairs of Cardholders and Merchants and combine the results of the previous paper with those of this paper each time we state in the hypothesis of a theorem that Cardholders or Merchants are not compromised.

Back to the formal model, our proofs followed the usual pattern suggested by Paulson [29]: possibility properties, regularity properties, secrecy properties, and finally the integrity and authenticity guarantees for the SET participants, namely, the Cardholder, Merchant, and Payment Gateway.

Possibility properties affirm that the protocol can run from start to finish and therefore that message formats are consistent between rounds. Possibility properties are logically trivial but are nontrivial to verify because of the size of the protocol. They say nothing about security but constitute a vital sanity check on the protocol definition. They are essential for protocols as complex as SET, where human experts cannot even read the complete view of the entire detailed and unfolded protocol. The formalized protocol may be secure simply because it cannot be run! Many researchers appear to ignore this sanity check. For the Purchase transaction, we proved possibility properties for both the signed and unsigned message flows.

Regularity properties are obvious consequences of the model: private keys cannot become compromised during a run, certificates signed by the Root Certification Authority are correct, and so forth. Their proofs are usually straightforward applications of induction followed by simplification and a small amount of classical reasoning.

As for *secrecy properties*, first we must prove that the symmetric keys used in digital envelopes are secure. From this lemma, we can prove that nonces encrypted using those keys are secure. Using these two lemmas, we can show that the first business requirement – confidentiality of the payment data – is satisfied. In particular, we have proved that both the PANSecret and the Cardholder's PAN remain secure. In one sense, these proofs require significant effort because their proof scripts occupy a substantial part of the proof script. However, thanks to Isabelle's level of automation, we proved them easily by building on our previous work on the Registration protocols [6]. Here is one example. A similar theorem expressing confidentiality of the PANSecret is not shown.

THEOREM 1 (PAN Secrecy, Signed). *If the Spy can get the PAN of a registered Cardholder, then the Cardholder has previously issued a Purchase Request involving a compromised Payment Gateway.*

Recall that *compromised* means under the Spy's control. This theorem is presented formally to illustrate how properties are specified in Isabelle.

```
⟦Pan (pan C) ∈ analz(knows Spy evs);  C = Cardholder k;
 CardSecret k ≠ 0;  evs ∈ set_pur⟧
⟹
 ∃P M KC2 PIDualSign1 PIDualSign2 other OIDualSign.
   Says C M {|{|PIDualSign1,
                EXcrypt KC2 (pubEK P) PIDualSign2 {|Pan(pan C),other|}|},
               OIDualSign|} ∈ set evs  ∧
   P ∈ bad
```

By *analz(knows Spy evs)* we denote the set of all messages that the Spy can deduce from his knowledge of the events of the generic trace *evs*. The inequality *CardSecret k) ≠ 0* expresses that *Cardholder k* is registered. The existentially quantified variables refer to the unknown (and irrelevant) items in the Purchase Request that was sent to the bad Payment Gateway (called *P*). Referring to the format of the Signed Purchase Request in Section 3 and in Figure 2, we see that the only relevant detail is *PANData*.

The proof, a typical confidentiality argument, involves induction followed by heavy equational simplification and the automatic elimination of trivial cases. It relies on a lemma on PANs that is proved by similar methods. The version for unsigned Cardholders must be stated and proved separately because the Purchase Request message has a different format.

The remaining theorems are essentially *integrity and authentication properties*. We adopted as a general guideline that the Cardholder, Merchant, and Payment Gateway should agree on all relevant details of the transaction. The Payment Gateway knows the Purchase Amount and credit card details. The

Merchant knows about the Order Description and Purchase Amount. The Cardholder knows both sets of information.

Most of these guarantees involve verifying digital signatures. Some of them also apply to unsigned purchases. On the whole, they are easily proved by induction; the most difficult ones rely on lemmas also proved by straightforward inductions. The main results appear below. Note that assumptions of the form *A* ∉ *bad* state that agent *A* is uncompromised. In some cases they are reasonable: when stating a guarantee for a Merchant, it is realistic to assume that the Merchant himself is uncompromised. In this case we do not mention it. In all other cases they mark an important assumption on our trust model: who, beside the partners of a message exchange, must be trusted for the protocol to be secure.

THEOREM 2. *When the Merchant receives* Authorization Response *from a trusted Payment Gateway, he knows that the Payment Gateway signed it, including the transaction identifiers and the purchase amount, which the Merchant can separately confirm.*

Here is the Isabelle formulation:

```
⟦MsgAuthRes = {|{|Number LID_M, Number XID, Number PurchAmt|},
                Hash authCode|};
 Crypt (priSK (PG j)) (Hash MsgAuthRes) ∈ parts (knows Spy evs);
 PG j ∉ bad;  evs ∈ set_pur⟧
⟹ ∃M KM KP HOIData HOD P_I.
      Gets (PG j)
           (EncB (priSK M) KM (pubEK (PG j))
                 {|Number LID_M, Number XID, HOIData, HOD|} P_I)
       ∈ set evs &
      Says (PG j) M
           (EncB (priSK (PG j)) KP (pubEK M)
                 {|Number LID_M, Number XID, Number PurchAmt|} authCode)
       ∈ set evs
```

Notice the condition *PG j* ∉ *bad*: if the Merchant can trust the *j*th Payment Gateway to be uncompromised (in both his data and his private keys), then Authentication Responses allegedly from the *j*th Payment Gateway do indeed come from him. The first conjunct in the conclusions of the theorem also tells us that the Payment Gateway has received the message with the authorization request and the corresponding XID.

The statement "When the Merchant receives Authorization Response" is not modeled literally, which would require a *Gets* event. Instead, we use the following condition:

```
Crypt (priSK (PG j)) (Hash MsgAuthRes) ∈ parts (knows Spy evs)
```

In plain English, it says "When the relevant part of the Authorization Response appears in the traffic." Indeed, *knows Spy evs* represents all past traffic, and *parts* denotes all parts of messages present there – whether they are visible to the Spy or not. This condition is necessary for the proof because, otherwise, the inductive hypothesis would be too weak.

Unfortunately, this theorem does not let us conclude that the Payment Gateway associates the transaction with this particular Merchant: the variable *M* in the conclusion is existentially quantified. That is because the Authorization Response message does not refer to *M*; abstractly, it is encrypted using *M*'s public key, but the digital envelope weakens that linkage. As so often in SET, signed messages lack explicitness: they should name the relevant parties rather than relying on the uniqueness of session identifiers.

THEOREM 3. *When the Merchant sees a dual signature from an uncompromised Cardholder, he can check (using LID_M) that it was intended for him and was issued by the Cardholder.*

Here is the formal version of this statement:

```
⟦MsgDualSign = {|HPIData, Hash OIData|};
 OIData = {|Number LID_M, etc|};
 Notes M {|Number LID_M, Agent P, extras|} ∈ set evs;
 Crypt (priSK C) (Hash MsgDualSign) ∈ parts (knows Spy evs);
 M = Merchant i;  C = Cardholder k;  C ∉ bad;  evs ∈ set_pur⟧
⟹ ∃PIData PICrypt.
     HPIData = Hash PIData &
     Says C M {|{|sign (priSK C) MsgDualSign, PICrypt|},
                OIData, Hash PIData|} ∈ set evs
```

Notice the condition *C* ∉ *bad*: The Merchant must assume that the Cardholder is uncompromised. This assumption may seem dubious, but it makes sense here. The intuition is that if a Cardholder has been compromised, her keys could be used by the Spy to sign arbitrary messages. No protocol can authenticate a user whose private keys are in the hands of hackers.

The remaining main theorems are for the Payment Gateway and the Cardholder and concern the last steps of the protocol.

THEOREM 4. *When a Payment Gateway sees a dual signature from uncompromised Cardholder and Merchant, he can verify that it originated with the given Cardholder for a transaction with the given Merchant. He can also verify that the Merchant intended him to handle the transaction.*

The formal version makes clear the requirement that the Merchant and Cardholder must be uncompromised. The conclusion asserts the existence of the

Merchant's *Notes* event from the initial shopping agreement and confirms the Cardholder's sending a signed Purchase Request.

```
⟦MsgDualSign = {|Hash PIData, HOIData|};
 PIData = {|PIHead, PANData|};
 PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
            TransStain|};
 Crypt (priSK C) (Hash MsgDualSign) ∈ parts (knows Spy evs);
 evs ∈ set_pur;  C ∉ bad;  M ∉ bad⟧
⟹ ∃OIData OrderDesc K j trans.
     HOD = Hash{|Number OrderDesc, Number PurchAmt|} &
     HOIData = Hash OIData &
     Notes M {|Number LID_M, Agent (PG j), trans|} ∈ set evs &
     Says C M {|{|sign (priSK C) MsgDualSign,
                  EXcrypt K (pubEK(PG j)) {|PIHead, Hash OIData|}
                          PANData|},
                OIData, Hash PIData|} ∈ set evs
```

THEOREM 5. *When the Cardholder receives Purchase Response from an uncompromised Merchant, he knows that the Merchant sent it. He also knows that the Merchant received a message signed by a Payment Gateway chosen by the Merchant to authorize the purchase.*

In the formal version, we see that the Merchant must be uncompromised. The conclusion asserts that the Merchant has participated in the initial shopping agreement, received an instance of Authorization Response, and sent an instance of Purchase Response.

```
⟦MsgPRes = {|Number LID_M, Number XID, Nonce Chall_C,
             Hash (Number PurchAmt)|};
 Gets C (sign (priSK M) MsgPRes) ∈ set evs;
 Notes C {|Number LID_M, Agent M, Agent C,
           Number OrderDesc, Number PurchAmt|} ∈ set evs;
 M ∉ bad;  evs ∈ set_pur⟧
⟹ ∃P KP trans.
     Notes M {|Number LID_M,Agent P, trans|} ∈ set evs &
     Gets M (EncB (priSK P) KP (pubEK M)
               {|Number LID_M, Number XID, Number PurchAmt|} authCode)
       ∈ set evs  &
     Says M C (sign (priSK M) MsgPRes) ∈ set evs
```

6. Failed Properties

What cannot be proved suggests potential vulnerabilities. It is impossible to prove that the Cardholder and Payment Gateway agree on the latter's identity.

Unless he trusts the Merchant, the Payment Gateway has no reason to believe that the Cardholder intended him to take part in the transaction. This lack of agreement occurs because the Cardholder does not sign anything that specifies the Payment Gateway.

If the original Payment Gateway is bad and can collude with a bad Merchant, then he can remove the encryption from the dual signature and communicate the Cardholder's allegedly confidential data to the Merchant, who can then send a new Authorization Request to an honest Payment Gateway. We do not expect to see this attack in the real world, primarily because a bad Payment Gateway has more lucrative crimes to commit. No amount of tinkering with SET can reduce the need for trust in Payment Gateways, who see the Cardholders' confidential account details. However, SET allows the Cardholder's software to abort the transaction *before* sending these confidential details if the proposed Payment Gateway is not certified by the same credit card company that issued the Cardholder's certificate [22, page 314]. This indicates that Cardholders are not expected to trust all Payment Gateways. It also confirms our view that the name of the Payment Gateway is an essential element of the transaction. Since we require all parties to agree on all essential elements, we certainly want them to agree on the choice of the Payment Gateway. The flaw can easily be fixed by inserting his identity into PIData.

It might be an interesting research issue to investigate what guarantees can be derived if the abortion of the protocol by the Cardholder in presence of a "wrong'' Payment Gateway certificate is modelled.

Digital envelopes complicated the proofs in various ways and further weakened our results for Payment Gateways. The simplified version of SET shown in Section 3 just uses public-key encryption, but our Isabelle model is closer to SET itself: public-key encryption is applied to a symmetric key, which is used to encrypt the bulk of the message. We had to prove secrecy of these symmetric keys, and the double encryptions caused case splits in subgoals. Also, we found it hard to prove that the symmetric keys were received intact. This may seem a peculiar thing to worry about because these keys are part of the security mechanism and not part of the data being transmitted. Still, it would be odd if Alice sent a digital envelope sealed with key K and Bob received this envelope but sealed with K'. These envelopes use hashing to establish a link between the two parts; recall the definition of `EXcrypt` near the end of Section 4. Again this problem is due to lack of explicitness: the key is not included in the hash, and it should be.

To summarize, the Payment Gateway can confirm neither the identity of the intended Payment Gateways nor the original symmetric key used in the Payment Information. This state of affairs is formalized as an ugly and unsatisfactory theorem.

THEOREM 6. *When a Payment Gateway receives an Authorization Request with a dual signature, he knows that Cardholder and Merchant packaged a*

Payment Instruction (not necessarily the one just received) for some Payment Gateway (not necessarily him) with some digital envelope (not necessarily the one just opened) where they agreed on certain details that he can check. Even if Purchase Amount is seen only by the Cardholder and not by the Merchant, both parties separately compute the hash of Order Description and Purchase Amount, and the Payment Gateway can compare them.

Figure 5 presents the Isabelle formulation. The variables *P_I'*, *P_I''*, *KC'*, *j'*, and so forth are necessary because we cannot prove that they are equal to the corresponding variables *P_I*, *KC*, *j*, and so forth. If we could prove these equalities, then we could simplify the form of the theorem considerably.

Theorem 4 should not be confused with Theorem 6. The former concerns a guarantee between Merchant and Payment Gateway; the latter concerns a guarantee between the Cardholder and the Payment Gateway. As we already explained, this asymmetry exists because the Payment Gateway is chosen by the Merchant and never mentioned by the Cardholder.

Other properties are customarily proved for authentication protocols. For instance, one can scan Lowe's [17] or Gollmann's [13] classification and check what variant of agreement is verifiable. This is a tricky question: we have eliminated fields that are immaterial to the main goals of the protocol as listed in the Business Description but that may be essential for other security properties.

```
⟦AuthReqData = {|Number LID_M, Number XID, HOIData, HOD|};
 KC ∈ symKeys;  C = Cardholder k;
 PANData = {|Pan (pan C), Nonce (PANSecret k)|};
 PIHead = {|Number LID_M, Number XID, HOD, Number PurchAmt, Agent M,
            TransStain|};
 PIData = {|PIHead, PANData|};
 PI_sign = sign (priSK C) {|Hash PIData, HOIData|};
 P_I = {|PI_sign, EXcrypt KC (pubEK(PG j)) {|PIHead, HOIData|} PANData|};
 Gets (PG j) (EncB (priSK M) KM (pubEK (PG j)) AuthReqData P_I)
    ∈ set evs;
 evs ∈ set_pur;  C ∉ bad;  M ∉ bad⟧
⟹ ∃OIData OrderDesc KM' trans j' KC' KC'' P_I' P_I''.
     HOD = Hash{|Number OrderDesc, Number PurchAmt|} &
     HOIData = Hash OIData &
     Notes M {|Number LID_M, Agent (PG j'), trans|} ∈ set evs &
     Says C M {|P_I', OIData, Hash PIData|} ∈ set evs &
     Says M (PG j') (EncB (priSK M) KM' (pubEK(PG j'))
                     AuthReqData P_I'') ∈ set evs &
     P_I' = {|PI_sign, EXcrypt KC' (pubEK (PG j'))
                      {|PIHead, Hash OIData|} PANData|} &
     P_I'' = {|PI_sign, EXcrypt KC'' (pubEK (PG j))
                      {|PIHead, Hash OIData|} PANData|}
```

Figure 5. P is (almost) assured that C and M agree: Theorem 6 in Isabelle syntax.

For instance, we have eliminated request–response identifiers that are recommended by Gong and Syverson [14] to make authentication protocols more robust and secure. Our weakened version of the protocol is sufficiently detailed for a careful verification of *some* properties, namely, the main properties that we identified in the specifications (see again Section 5). We do not claim that attacks against our abstract model necessarily work in full SET. If a researcher is interested in verifying sophisticated authentication properties, he should not use our model as is but should reconsider the steps that we have simplified.

7. Theorem-Proving Aspects

Verifying a large protocol places enormous demands on the theorem prover and its human user. The formulas tend to be large, complex, and unintelligible. Isabelle can simplify such formulas using rewrite rules, but the main weapon against such formulas is Isabelle's *classical reasoner* [28, 30]. This is essentially a tableau prover, but it applies known lemmas in addition to the basic rules of first-order logic. The user invokes the classical reasoner via tactics such as *blast* (pure tableau reasoning), `auto` (classical reasoning combined with rewriting to break up all subgoals), and `force` (a brute-force combination of classical reasoning and rewriting that attempts to prove a subgoal) and a number of user-defined tactics.

The verification of SET comprises six Isabelle theories totaling over 4,300 source lines. Two of these theories concern Cardholder and Merchant Registration. The other four are as follows.

MessageSET.thy contains a theory of messages for cryptographic protocol analysis, based on the one used in prior work but adapted for SET. The major modification is that we break down agents into Cardholders, Merchants, Payment Gateways, and Certification Authorities. We also introduce a new type (distinct from nonces) for PANs.

EventSET.thy describes the general theory of protocol events, building on the previous theory. The only SET-specific adaptation is to specify that the root certification authority is not compromised.

PublicSET.thy describes the complicated world of public keys, digital certificates, digital envelopes, and similar constructs used in SET. It defines 19 different functions and abbreviations.

Purchase.thy specifies and verifies Purchase Request and Payment Authorization, regarded as a single protocol.

Table I presents a number of summary statistics. As one can see from the table, the fully automatic proof tactics (`blast`, `auto`, `force`) are not sufficient to tame the problem even when given the required lemmas. Our proof strategy is

Table I. Script statistics

	MessageSET	EventSet	PublicSET	Purchase
Proved theorems	128	14	48	59
Total tactics	266	25	77	250
blast tactic	67	0	16	35
auto, *force* tactics	60	14	20	38
simp tactic	88	10	19	14
Inductions	47	6	4	30
Adding auto rules	42	1	23	47
Removing rules	1	2	0	7
New tactics defined	3	1	1	0

usually the following: identify the required form of induction; give the resulting subgoals to automatic tactics; try splitting up any remaining subgoals into separate cases; and look for ways to improve simplification. We may conclude that a lemma must be proved by induction, requiring a recursive application of the strategy.

In this section, we consider three difficult inductions in more detail. In each, we devote attention to one particular subgoal: the one corresponding to a signed Purchase Request. This is the most interesting message, and it illustrates the problem of abbreviations getting expanded in proofs.

The first example is a lemma required to prove the secrecy of symmetric keys. Similar lemmas are discussed in earlier papers [6, 29], but the proof dramatically blows up with the Purchase protocols. As always, the form of the theorem involves quantification and embedded implications (expressed using $\longrightarrow$), as well as set operators.

```
lemma symKey_compromise:
     "evs ∈ set_pur ⟹
      (∀SK KK. SK ∈ symKeys ⟶
        (∀K ∈ KK. K ∉ range(λC. priEK C)) ⟶
              (Key SK ∈ analz (Key‘KK ∪ (knows Spy evs))) =
              (SK ∈ KK ∨ Key SK ∈ analz (knows Spy evs)))"
```

Recall that induction generates a subgoal for every protocol step and also for the base case, the case for the spy, and a few other cases. We get 13 subgoals in all, each containing a copy or copies of the induction formula. Prior to simplification, we prepare some of the subgoals by adding certain facts that hold in those cases (these are consequences of existing assumptions). At this point, the Isabelle proof state displayed to the user is 418 lines long, the Purchase Request case taking up 93 lines of this. Such huge subgoals defy comprehension.

They can be tackled only by tools that prove them outright or at least break them down to a few manageable parts.

Here, simplification is the obvious step. Simplification is particularly taxing in secrecy proofs because the necessary rewrite rules tend to cause a lot of case splits. Simplification of this large proof state takes about 16 s (on a 2.5 GHz Apple G5 processor). The simplified Purchase Request case becomes comprehensible: it is 23 lines long and contains no complex expressions. Figure 6 displays this subgoal exactly as it appears to the Isabelle user. Six subgoals (comprising 191 lines) survive simplification. The one concerning the Spy is proved automatically by a special tactic dedicated to that purpose. The other five subgoals, including Purchase Request, are proved automatically by Isabelle's *blast* tactic. The total runtime to prove the six subgoals is under 1 s.

Now consider the proof of the secrecy of the PAN, Theorem 1. It differs from the previous secrecy proof in several respects. One obvious difference is the need for existential quantifiers in the induction formula. Prior to simplification, the Isabelle proof state is approximately as large and complicated as in the proof described above. Simplification takes only 8 s and leaves only three subgoals. Unfortunately, the Purchase Request case has blown up to an unintelligible 136 lines: the expansion of abbreviations did not lead to further simplifications. Fortunately, *blast* can prove this subgoal. Appropriate tactics prove the other two subgoals automatically, in a fraction of a second.

For a third example, consider Theorem 2. Although the proof is by induction, it is a regularity property. Such theorems tend to have simpler proofs than those involving secrecy. Just before simplification, although the proof state as a whole is unusually large, the Purchase Request case is only 55 lines long. Simplification

```
3. ⋀Chall_C Chall_M KC2 LID_M M OrderDesc P PurchAmt XID evsPReqS k SK KK.
     ⟦evsPReqS ∈ set_pur;
      ∀SK. SK ∈ symKeys ⟶
           (∀KK. (∀K∈KK. ∀i. priEK i ≠ K) ⟶
                 (Key SK ∈ analz (Key ` KK ∪ knows Spy evsPReqS)) =
                 (SK ∈ KK ∨ Key SK ∈ analz (knows Spy evsPReqS)));
      0 < CardSecret k; Key KC2 ∉ used evsPReqS; KC2 ∈ symKeys;
      Gets (Cardholder k)
       (sign (priSK M)
         {Number LID_M, Number XID, Nonce Chall_C, Nonce Chall_M,
          cert P (pubEK P) onlyEnc (priSK RCA)})
      ∈ set evsPReqS;
      Says (Cardholder k) M {Number LID_M, Nonce Chall_C} ∈ set evsPReqS;
      Notes (Cardholder k)
       {Number LID_M, Agent M, Agent (Cardholder k), Number OrderDesc,
        Number PurchAmt}
      ∈ set evsPReqS;
      SK ∈ symKeys; ∀K∈KK. ∀i. priEK i ≠ K⟧
     ⟹ Cardholder k ∉ bad ⟶
        P ∉ bad ⟶
        priEK P ∈ KK ⟶
        SK ∈ KK ∨ SK = KC2 ∨ Key SK ∈ analz (knows Spy evsPReqS) ⟶
        SK ∈ KK ∨ Key SK ∈ analz (knows Spy evsPReqS)
```

Figure 6. Secrecy proof: the purchase request subgoal, simplified.

proves this case and leaves only two subgoals. Figure 7 presents one of these subgoals as it is displayed to the user. It is rather incomprehensible compared with the one appearing in Figure 6. Fortunately, `blast` proves the two remaining subgoals automatically.

Given the complexity of these proofs, the attraction of fully automatic verifiers is obvious. The advantage of using a general-purpose tool such as Isabelle is its generality: we can change the formalization to model new types of protocols and new security environments.

8. Related Work

Only few others have attempted to verify SET. Stoller [34] has proposed a theoretical framework for the bounded analysis of e-commerce protocols but has considered only a hugely simplified description of the payment protocols of SET.

Meadows and Syverson [25] have proposed a language for describing SET specifications but have not actually verified the protocol. They have used the temporal language NPATRL (the NRL Protocol Analyser Temporal Requirements Language) for specifying a number of SET's requirements. Some requirements are technical, such as "honest principals will faithfully execute the protocol," while others directly address the protocol goals. The paper is not about verifying those requirements, which is left as future work. Instead, it concentrates on the difficulties in specifying them formally, an issue that concerns us, too.

```
1. /\X evsf.
      [[PG j ∉ bad; evsf ∈ set_pur;
       Crypt (priSK (PG j)) (Hash {|Number LID_M, Number XID, Number PurchAmt|})
       ∈ parts (knows Spy evsf) ⟶
       (∃i. (∃KM HOIData HOD P_I.
              Gets (PG j)
               (EncB (priSK (Merchant i)) KM (pubEK (PG j))
                 {|Number LID_M, Number XID, HOIData, HOD|} P_I)
              ∈ set evsf) ∧
            Says (PG j) (Merchant i)
             (Crypt (pubEK (Merchant i))
               (sign (priSK (PG j))
                 {|Number LID_M, Number XID, Number PurchAmt|}))
            ∈ set evsf);
       X ∈ synth (analz (knows Spy evsf))]]
      ⟹ Crypt (priSK (PG j))
          (Hash {|Number LID_M, Number XID, Number PurchAmt|})
        ∈ parts (insert X (knows Spy evsf)) ⟶
        (∃i. (∃KM HOIData HOD P_I.
               Gets (PG j)
                (EncB (priSK (Merchant i)) KM (pubEK (PG j))
                  {|Number LID_M, Number XID, HOIData, HOD|} P_I)
               ∈ set evsf) ∧
             Says (PG j) (Merchant i)
              (Crypt (pubEK (Merchant i))
                (sign (priSK (PG j))
                  {|Number LID_M, Number XID, Number PurchAmt|}))
             ∈ set evsf)
```

Figure 7. The Spy subgoal, simplified.

Kessler and Neumann [16] have extended an existing belief logic with predicates and rules to reason about *accountability*. Although accountability is not a stated goal of SET, it is clearly desirable. They concentrate on the Merchant's ability to prove to a third party that the Order Information originated with the Cardholder. Using the calculus of the logic, they conclude by pen and paper that the goal is met, so the Cardholder cannot repudiate the transaction. We have an equivalent result in Theorems 3 and 4: if a Merchant receives a dual signature from the network, then the Cardholder sent it (unless the Cardholder is compromised).

Most automatic protocol verification tools rely on finite-state model checking. In contrast, the inductive approach to verifying security protocols can deal with unbounded numbers of protocol participants, nonces, interleaved sessions, and so forth. This feature, combined to the mechanical support offered by the proof assistant Isabelle, turns out to be important in analyzing huge protocols such as SET.

For example, a careful analysis of the data underlying the customary table reporting new and old bugs on the Clark–Jacob library in model checking papers reveals that serious limitations were imposed on the number of agents, sessions, nonces, and keys. Our simplified SET protocol requires eight nonces and symmetric keys for a single protocol run from start to end. This is two or three times more than the number of fresh values allowed by the best finite state model-checkers. Parallel sessions would require an order of magnitude more nonces than what is currently feasible. Yet, failure of agreement on the identity of the Payment Gateway (Section 6) would remain invisible if we allowed for only one Gateway.

Recent work by Basin et al. [3] makes significant improvements over traditional model-checking. Infinite elements are represented by using *lazy data-types*: constructors that build data-types without evaluating their arguments. The potentially infinite messages that the Spy can introduce are treated using a dedicated symbolic representation. The resulting method is tested on a number of classical protocols and on two real protocols, IKE and SET. Unfortunately, a closer comparison with our contribution is impossible because details about their SET analyses had not been published at the time of this writing.

Many other formal approaches to protocol verification do not impose finite bounds. These include strand spaces [12], the Spi-calculus [1], and the work of Bozzano and Delzanno [10], who adopt a general-purpose bottom-up evaluation scheme for first-order linear logic. Meadows surveys many such approaches [24]. Methods such as the strand space or the Spi-calculus are useful for gaining insights into security protocol theory, but not for verifying real-world protocols. The method by Bozzano and Delzanno is promising, but it imposes certain conditions on the protocol theory that may not apply to SET. It seems clear that no pencil-and-paper method can verify real-world protocols.

We also mention Hui and Lowe's work [18]. They have proposed a general theory to transform a complex protocol into a simpler protocol while preserving

any faults. However, they limited their analysis to the Cybercash protocol. Furthermore, they offer no guidelines for the determination of the safe simplifying transformation. Only for authentication protocol is there a sufficient condition. When applied to SET, this would lead to either no simplification (all nonces and session keys are retained) or a trivial protocol (only the PAN is retained).

9. Conclusions

Until now, the most complex protocols analyzed by using the inductive method were Kerberos IV [9], TLS (the successor to SSL) [31], Shoup–Rubin (which adopts smartcards) [4], and the Registration protocols of SET [6]. The verification of the Purchase protocols of SET has still been an open problem.

We succeeded in analyzing an abstract, but still highly complex, version of SET's Purchase protocols. The difficulty consisted in digesting the specification and scaling up. This is a major result: our methods scale to a level of complexity where intuition falters. However, the proofs often generated huge subgoals spanning several pages of text, stretching the human interaction with the prover.

Where do we go from here? The analysis of more complex protocols probably requires further advances, either in automation or in user interfaces. Isabelle is a general-purpose proof assistant; a specialized protocol verifier might be able to do better. For example, the visualization of intermediate proof steps would be improved if we could avoid expanding abbreviations, with the consequent exponential blowup. No other protocols that we have seen make such heavy use of abbreviations, but they will become increasingly common as people try to verify industry standards. Research is therefore needed on how to reason in the presence of abbreviations.

A theory of abstraction and compositionality might allow the proof of a big protocol to be divided into smaller parts. We should be able to separate the correctness proof for digital envelopes from that of a protocol that uses digital envelopes. Both automatic and manual verifiers would benefit. The problem of compositionality is being tackled for manual analysis by Guttman et al. [15, Section 6] using strand spaces and by Durgin et al. [11] using a specialized protocol logic. The development and exploitation of such a theory are a major research problem, and we can expect any correctness proof for digital envelopes to impose conditions on how the protocol uses them. This still should lead to simpler proofs than at present, where we simply expand out the definitions of the envelopes.

The hardest task in our verification of SET has been that of digesting and abstracting the specifications. For us, the *Formal Protocol Definition* [21] is misleadingly named; it appears to consist of the *Programmer's Guide* [22] minus the information on how messages are handled. It should include explicit, formal statements of the protocol's goals. Complex protocols should be specified as

refinements of more abstract protocols, whereas at present protocol verifiers have to discover the abstract protocols themselves. It is a waste of effort, for the design must have evolved from an abstract protocol.

The myth that protocol verification is prohibitively expensive can be laid to rest. The cost of our efforts is small compared with the total cost of the SET design. The verification would have been cheaper and easier if we could have received the essential protocol directly from the designers. Verification should be an integral part of the design process.

From a security standpoint, it is customary to expect that every protocol is either correct or else vulnerable to attacks. However, SET lies in neither extreme. We were able to prove the most important goals, thereby giving grounds for reasonable confidence in SET. Yet, in the issue of agreement between the Cardholder and Payment Gateway on the latter's identity, we found that the property fails. This flaw is easy to fix.

Acknowledgements

F. Massacci was partially funded by grants EU IST/2001/37004 *WASP*, FIRB/RBNE0195K5 *ASTRO* and FIRB/RBAU01P5SS. The work at Cambridge was funded by the EPSRC grant GR/R01156/R01 *Verifying Electronic Commerce Protocols.*

References

1. Abadi, M. and Gordon, A.: A calculus for cryptographic protocols: The spi calculus, in *Proc. 4th ACM Conf. on Comm. and Comp. Sec. (CCS-97)*, ACM and Addison Wesley, 1997.
2. Abadi, M. and Needham, R. M.: Prudent engineering practice for cryptographic protocols, *IEEE Trans. Softw. Eng.* **22**(1) (January 1996), 6–15.
3. Basin, D., Mödersheim, S. and Viganò, L.: An on-the-fly model-checker for security protocol analysis, in E. Snekkenes and D. Gollmann (eds.), *Proc. 8th Eur. Symp. on Res. in Comp. Sec.*, Volume 2000 of *Lecture Notes in Comp. Sci.*, Springer, 2003, pp. 253–270.
4. Bella, G.: Inductive verification of smart card protocols, *J. Comput. Secur.* **11**(1) (2003), 87–132.
5. Bella, G., Massacci, F. and Paulson, L. C.: The verification of an industrial payment protocol: The SET purchase phase, in V. Atluri (ed.), *9th ACM Conf. on Comp. and Comm. Sec.*, ACM, 2002, pp. 12–20.
6. Bella, G., Massacci, F. and Paulson, L. C.: Verifying the SET registration protocols, *IEEE J. Sel. Areas Commun.* **21**(1) (2003), 77–87.
7. Bella, G., Massacci, F. and Paulson, L. C.: An overview of the verification of SET, *Int. J. Inf. Secur.* **4**(1–2) (2005), 17–28.
8. Bella, G., Massacci, F., Paulson, L. C. and Tramontano, P.: Formal verification of cardholder registration in SET, in F. Cuppens, Y. Deswarte, D. Gollman and M. Waidner (eds.), *Computer Security – ESORICS* 2000, volume 1895 of *Lecture Notes in Comp. Sci.*, Springer, 2000, pp. 159–174.

9. Bella, G. and Paulson, L. C.: Kerberos version IV: Inductive analysis of the secrecy goals, in Quisquater et al. [32], pp. 361–375.
10. Bozzano M. and Delzanno G.: Automated protocol verification in linear logic, in *Proc. 4th ACM Conf. on Principles and Practice of Declarative Programming (ACM PPDP'02)*, ACM and Addison Wesley, 2002, pp. 38–49.
11. Durgin, N., Mitchell, J. and Pavlovic, D.: A compositional logic for proving security properties of protocols, *J. Comput. Secur.* **11**(4) (2004), 677–721.
12. Fábrega, F. J. T., Herzog, J. C. and Guttman, J. D.: Strand spaces: Proving security protocols correct, *J. Comp. Secur.* **7** (1999), 191–220.
13. Gollmann, D.: What do we mean by entity authentication? in *Proc. 15th IEEE Symp. on Security and Privacy*, IEEE Comp. Society Press, 1996, pp. 46–54.
14. Gong, L. and Syverson, P.: Fail-stop protocols: An approach to designing secure protocols, in *Proc. 5th IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, September 1995.
15. Guttman, J.: Security goals: Packet trajectories and strand spaces, in R. Focardi and F. Gorrieri (eds.), *Foundations of Security Analysis and Design – Tutorial Lectures*, volume 2171 of *Lecture Notes in Comp. Sci.*, Springer, 2001, pp. 197–261.
16. Kessler, V. and Neumann, H.: A sound logic for analysing electronic commerce protocols, in Quisquater et al. [32].
17. Lowe, G.: A hierarchy of authentication specifications, in *Proc. 10th IEEE Comp. Sec. Found. Workshop*, IEEE Comp. Society Press, 1997, pp. 31–43.
18. Lowe, G. and Hui, M. L.: Fault-preserving simplifying transformations for security protocols, *J. Comput. Secur.* **9** (2001), 3–46.
19. Mastercard & VISA: *SET Secure Electronic Transaction*: *External Interface Guide*, May 1997. On the Internet at http://www.setco.org/set_specifications.html.
20. Mastercard & VISA: *SET Secure Electronic Transaction Specification: Business Description*, May 1997. On the Internet at http://www.setco.org/set_specifications.html.
21. Mastercard & VISA: *SET Secure Electronic Transaction Specification: Formal Protocol Definition*, May 1997. On the Internet at http://www.setco.org/set_specifications.html.
22. Mastercard & VISA: *SET Secure Electronic Transaction Specification: Programmer's Guide*, May 1997. On the Internet at http://www.setco.org/set_specifications.html.
23. Meadows, C.: Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer, in *SSP-99*, IEEE Comp. Society Press, 1999, pp. 216–231.
24. Meadows, C.: Formal methods for cryptographic protocol analysis: Emerging issues and trends, *IEEE J. Sel. Areas Commun.* **21**(1) (2003), 44–54.
25. Meadows, C. and Syverson, P.: A formal specification of requirements for payment transactions in the SET protocol, in R. Hirschfeld, (ed.), *Proc. Financial Cryptography 98*, volume 1465 of *Lecture Notes in Comp. Sci.* Springer, 1998.
26. Nipkow, T., Paulson, L. C. and Wenzel, M.: *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.
27. Paller, A.: Alert: Large criminal hacker attack on Windows NTE-banking and E-commerce sites. On the Internet at http://www.sans.org/newlook/alerts/NTE-bank.htm, Mar. 2001. SANS Institute.
28. Paulson, L. C.: Generic automatic proof tools, in R. Veroff (ed.), *Automated Reasoning and its Applications: Essays in Honor of Larry Wos*, chapter 3. MIT Press, 1997.
29. Paulson, L. C.: The inductive approach to verifying cryptographic protocols, *J. Comput. Secur.* **6** (1998), 85–128.
30. Paulson, L. C.: A generic tableau prover and its integration with Isabelle, *J. Univers. Comput. Sci.* **5**(3) (1999), 73–87.

31. Paulson, L. C.: Inductive analysis of the internet protocol TLS, *ACM Trans. Inf. Syst. Secur.* **2**(3) (1999), 332–351.
32. Quisquater, J.-J., Deswarte, Y., Meadows, C. and Gollmann, D. (eds.), *Computer Security – ESORICS* 98, volume 1485 of *Lecture Notes in Comp. Sci.* Springer, 1998.
33. RSA Laboratories. *PKCS-7: Cryptographic Message Syntax Standard*, 1993. On the Internet at http://www.rsasecurity.com/rsalabs/pkcs.
34. Stoller, S. D.: A bound on attacks on payment protocols, in *Proc. 16th Annual IEEE Symposium on Logic in Computer Science (LICS)*, June 2001.