An Overview of the Verification of SET

Giampaolo Bella[1,2], **Fabio Massacci**[3], **Lawrence C. Paulson**[1]

[1] Computer Laboratory, University of Cambridge
J J Thomson Avenue, Cambridge CB3 0FD, UK
e-mail: {gb221,lcp}@cl.cam.ac.uk
[2] Dipartimento di Matematica e Informatica, Università di Catania
Viale A. Doria 6, I-95125 Catania, Italy
e-mail: giamp@dmi.unict.it
[3] Dipartimento di Informatica e Telecomunicazioni, Università di Trento
Via Sommarive 14, I-38050 Povo (Trento), Italy
e-mail: massacci@ing.unitn.it

Abstract This paper describes the verification of Secure Electronic Transaction (SET), an e-commerce protocol by VISA and MasterCard. The main tasks are to comprehend the written documentation, to produce an accurate formal model, to identify specific protocol goals, and finally to prove them. The main obstacles are the protocol's complexity (due in part to its use of digital envelopes) and its unusual goals involving partial information sharing. Our verification efforts show that the protocol does not completely satisfy its goals, although the flaws are minor. The primary outcome of the project is experience with verification of enormous and complicated protocols.

1 Introduction

The last 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 theorem-proving procedures for security verification have boosted a field which outsiders believe populated only by "Yet-Another-Look-at-Needham-Schroeder" papers.

Protocol verification techniques fall into several categories. A general-purpose model-checker can verify protocols, as pioneered by Lowe and his colleagues at Oxford [11]. A general-purpose proof tool can also be effective, as in Paulson's work [21]. Additionally, there exist several specialized protocol analysis tools. Most perform an exhaustive search in the spirit of

model checking; among the best is Meadows' NRL [17], which has deductive capabilities. Cohen's TAPS processes the protocol specification and verifies the desired properties using a resolution theorem prover [9]. Meadows [18] presents an exhaustive survey of recent methods.

Formal proof is preferable for establishing properties, while model-checking is best for finding attacks. Exhaustive search is only feasible if the model is kept as small as possible, for example by minimizing the number of permitted executions. If the assumptions are too strong, the absence of an attack does not guarantee correctness. Interactive proof tools are not automatic, but offer flexibility in expressing specifications and proofs. Models need not be finite and can therefore be more realistic.

Many practical industrial protocols have been formally verified using interactive or semi-interactive proof tools. Protocols like Kerberos IV [7], the Internet Key Exchange protocol [17], the Cybercash protocol [12], the TLS/SSL protocol [22], all yielded to automatic or semi-automatic tools. One particular protocol has proved to be particularly resistant to verification: the SET (Secure Electronic Transaction) protocol by Visa and Mastercard.

SET [14,15,16] has been proposed by a consortium of credit card companies and software corporations to secure e-commerce transactions. When a customer makes a purchase, the SET protocol guarantees authenticity of the transaction while keeping the customer's account details secret from the merchant and his choice of goods secret from the bank. Its appeal to researchers working in verification is the possibility of demonstrating that one's own verification technology is mature enough to cope with the demands of a huge, complex, industrial protocol.

Indeed, many researchers have worked on the problem: for instance Meadows and Syverson [19] have proposed a language for describing SET specifications but have not actually verified the protocol. Kessler and Neumann [10] 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 upon 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. Stoller [26] has proposed a theoretical framework for the bounded analysis of e-commerce protocols but has only considered an overly simplified description of the payment protocol of SET. Hui and Lowe [12] have proposed a general theory to transform a complex protocol into a simpler protocol while preserving any faults. However, they limited their actual analysis to the Cybercash protocol. The claim "we plan to apply our verification technology to SET" was a frequent conclusion to talks and papers at the end of the millennium. Yet, the protocol resisted most verification attempts.

Why is SET such a challenge for formal verification? The first obstacle is its documentation [13,14,15,16] which takes over 1000 pages. However, the main obstacle is the protocol itself. Protocols proposed in scientific journals

are typically short, straight-line programs: they seldom go beyond two levels of encryption and generate few secrets. Even more sophisticated protocols such as Optimistic Fair Exchange [2] or Group Protocols can be described in a few pages. Internet protocols such as IKE and TLS use cryptography rather sparingly compared to SET. SET has many features that make its verification unusual and hard.

The complex structure of SET makes it a benchmark for security protocol design and verification, whether or not it will be a commercial success. Such a gigantic protocol cannot be convincingly verified without tool support. However, tools require formal models. Even the task of designing an adequate formal model may be too much for human intuition.

We succeeded in analysing an abstract, but still highly complex, version of SET: the registration phases [5] and the purchase phase [4]. The difficulty consisted in digesting the specification and scaling up. This is a major result: our method scales to a level of complexity where intuition falters. Unfortunately, we discovered that the method, based on human interaction with a semi-automatic but powerful prover, has reached a point where the complexity of the proofs and the sheer size of the intermediate properties will require further advances to scale further.

The paper begins by outlining the SET protocol (Sect. 2). It briefly introduces the inductive approach and Isabelle (Sect. 3). It outlines our proofs of the registration protocols (Sect. 5) and the purchase protocols (Sect. 6) of SET. Finally, there are some general conclusions (Sect. 7).

2 The SET Protocol

People today pay for online purchases by sending their credit card details to the merchant. A protocol such as SSL or TLS keeps the card details safe from eavesdroppers, but does nothing to protect merchants from dishonest customers or vice-versa. SET addresses this situation by requiring cardholders and merchants to register before they may engage in transactions. A cardholder registers by contacting a certificate authority, supplying personal account details and his proposed signature verification key (the public half). Registration allows the authorities to vet an applicant, who if approved receives a certificate confirming that his public key has been registered. All orders and confirmations bear digital signatures, which provide authentication and could potentially help to resolve disputes.

A SET purchase involves three parties: the cardholder, the merchant, and the payment gateway (loosely speaking a bank). The cardholder shares the order information with the merchant but not with the payment gateway. He shares the payment information with the bank but not with the merchant. A SET *dual signature* accomplishes this partial sharing of information: the cardholder makes separate hashes of the order information and the payment information and signs the pair of hashes. Each other party receives the hash of the withheld information and the signature of the pair.

Each party can confirm that the hashes in their possession agree with the hash signed by the cardholder. In addition, cardholder and merchant compute equivalent hashes for the payment gateway to compare. He confirms their agreement on the details withheld from him.

All parties are protected. Merchants do not normally have access to credit card numbers. Moreover, the mere possession of credit card details does not enable a criminal to make a SET purchase[1]: he needs the cardholder's signature key and a secret number that the cardholder receives upon registration.

SET is a family of protocols. The five main ones are cardholder registration, merchant registration, purchase request, payment authorization, and payment capture. There are many additional minor protocols, for example to handle errors. SET is enormously more complicated than SSL, which merely negotiates session keys between the cardholder's and merchant's Internet service providers.

Let us briefly review its interesting features:

- Security bootstrapping is unusual: the initiator possesses no digital proof of identity and authenticates himself by filling in a registration form whose format is not specified. Authentication takes place outside the protocol, when the cardholder's bank examines the completed form.
- The protocol uses multiple nested encryptions and several message fields. These require abbreviations, make the manual unwinding of the specifications impossible and restrict analysis to tools supporting equational reasoning.
- SET uses *digital envelopes*. A digital envelope consists of two parts: one, encrypted using a public key, contains a fresh symmetric key K and identifying information; the other, encrypted using K, conveys the full message text. Digital envelopes keep public-key encryption to a minimum, but the symmetric keys complicate the reasoning. It hampers the usual model-checking technique to limit the state space (limiting different keys and nonces to an handful) as it would not even allow a single execution to complete, let alone two or more parallel ones;
- The goal of the protocol is to protect the information about merchandise from the bank and the information about credit from the merchant while authenticating the entire transaction. The partial sharing of information among the three peers leads to unusual protocol goals.
- SET has many alternative protocol paths that make it impossible 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.

Are these features or bugs? Though some security experts may claim that SET is badly designed because it was designed by a committee, others will

[1] Some optional features of SET (presumably demanded by commercial or credit practices) weaken these properties. A merchant can be authorized to receive credit card numbers and has the option of accepting payments without digital signatures.

rightly claim that many of these features are actually needed in any practical protocol. For example, alternative protocol paths are necessary in any practical scenario: recall that the task of an e-commerce protocol is first doing business, second doing it securely. Security-aware customers may have pre-registered with a financial institution and thus secured their credit cards from the merchant's eyes. Other customers may decide to trust the merchant and thus be content with a transaction secured against the outside world. From a merchant's perspective, all customers should be able to conclude a purchase, whether they bothered to pre-register or not.

This paper is intended to summarize our work on the SET protocol: the issues, the results and the lessons learned. Detailed descriptions of the verification are published elsewhere [4, 5, 6].

3 Isabelle and Inductive Protocol Verification

We used the Isabelle theorem prover with the inductive approach to protocol verification, building on the previous experience on a wide range of protocols, including industrial ones such as Kerberos [7] and TLS [22].

Isabelle/HOL [20] is an interactive proof tool for higher-order logic. Isabelle provides a simplifier, a predicate calculus theorem prover, a choice of proof languages, and automatic generation of LaTeX documents. Isabelle's support for inductive definitions is particularly strong, both in its specification language and in its prover automation. However, other tools for higher-order logic could be suitable, provided they fully support conditional equational reasoning.

The inductive approach [21] verifies protocols using the standard techniques of operational semantics. An inductive definition defines the possible executions of a system consisting of the honest protocol participants and an active attacker, the Spy. An execution comprises any number of attempted protocol runs and is a trace of message transmissions and other events.

Authentication and agreement are expressed using safety properties over traces and proved by induction over traces. For example, we can prove that any trace containing a particular event x must also contain some other event y. Secrecy properties are hardest to prove. For example, if we are concerned with the secrecy of a certain key K, then we must prove $K \neq K'$ for each key K' that might be compromised. Every encrypted message produces a case split, since we must prove that K is secure whether or not the encrypting key is. Protocols with many steps or many options can then generate a huge number of cases. Despite the difficulties, we can use established techniques and tools in our attempt to prove secrecy.

Most protocols, even esoteric ones like non-repudiation and fair exchange protocols, involve the standard cast of characters: Alice, Bob, possibly Charlie, and a trusted third party. SET is different: it has cardholders, merchants, payment gateways, and a hierarchy of certificate authorities. Changing Isabelle's theory of protocols to use SET's cast of characters was easy.

The model includes a set of honest agents, whose long-term keys can never become compromised. (Arguably, our model is too optimistic.) For typical protocols, where long-term keys are never transmitted, proving that they remain secure is trivial. The Spy controls another set of agents, with full access to their internal states. The Spy also controls the network and retains every transmitted message. Session keys may become compromised, for example if they are sent to compromised agents.

A standard theory of messages and their constructors underlies these inductive models. Messages in our model form a recursive datatype (equivalently, a free algebra). A nonce can never equal an agent name or a session key, for example. Such assumptions are more realistic than one might expect: different kinds of items are likely to have different lengths and even a different bit-wise encoding. Consider concatenation of messages, which may seem to be inherently associative. The ISO-DER encoding of a sequence of six random numbers has a bit-wise encoding different from the concatenation of a pair of sequences of three numbers.

Encryption is injective in our theory. Only one key can decrypt a ciphertext, which can yield only one plaintext. This assumption is plainly false for low-level applications of encryption, where using the wrong key yields a plaintext of random bits. However, it is correct provided "each encrypted message contains sufficient redundancy to allow a principal who decrypts it to verify that he has used the right key," to quote Burrows et al. [8, p. 237]. Most research on protocol verification relies on this assumption.

Our model does not allow reasoning about exclusive-OR. Exclusive-OR breaks down our representation of messages as a free algebra, since it satisfies several equations. Exclusive-OR is associative, commutative and self-cancelling. Intuitively, the problem is that the exclusive-OR of two messages can potentially yield a message of any form. Fortunately, SET uses exclusive-OR only in one place: at the end of Cardholder Registration, to compute the so-called PANSecret. We do not attempt to prove the secrecy of the PANSecret, merely of the two random numbers used in its calculation. Proving the secrecy of the PANSecret would require additional assumptions in order to exclude the possibility that the exclusive-OR could yield an existing secret. As discussed in Sect. 5.4 below, SET's use of exclusive-OR introduces a vulnerability.

4 Modelling Issues

Researchers compete to produce the fastest automatic tools. However, the main obstacle to protocol verification lies in digesting the documentation and producing a formal model. Understanding hundreds of pages of text is a massive undertaking. Meticulous care is essential to avoid defining an incorrect model.

The main SET documents are the *Business Description* [14], the *Programmer's Guide* [16], and the *Formal Protocol Definition* [15]. SET is de-

fined using Abstract Syntax Notation One (ASN.1).[2] The *Programmer's Guide* presents each message format as a figure based on the underlying ASN.1 definition, augmented with a detailed English description of how to process each message. The *Formal Protocol Definition* consists of the *Programmer's Guide* with the ASN.1 notation inserted and the English text removed. Since the ASN.1 adds little to the figures, the formal protocol definition essentially consists of syntax without semantics. It describes the message formats but says nothing about how messages are processed. For that information, we had to rely on the *Programmer's Guide*.

The enormous size and complexity of the SET message formats demanded simplification. As we have discussed elsewhere [6,3], this was not always straightforward, forcing us to decide what constituted SET's core feature set. For example, we eliminated payment by instalments (since it can be modelled by repeated transactions) and modelled only authorized transactions (so unauthorized transactions were modelled by silent denial). Other researchers can make other choices.

Attacks against protocols often arise from unclear assumptions about the operating environment rather than from flaws in the protocols themselves. Experts can dispute whether the formal model accurately reflects the real world and thus whether the attack is realistic. Consider Lowe's famous attack [11] against the Needham-Schroeder public-key protocol: Alice talks to Charlie, who happens to be dishonest and proceeds to fool Bob. In this scenario, Charlie is a dishonest insider. However, Needham and Schroeder designed the protocol with the express purpose of protecting the honest insiders from outsiders.

SET has a much more complex environment and parts of its operation are specifically left "out of band." Our formal model has to make reasonable assumptions about these parts which are sketched in the *SET External Interface Guide* [13]. It also must specify which insiders can be compromised and innumerable other details. It also has to define the protocol goals, since the documentation outlines them only in general management terms.

5 Verifying The Registration Protocols

The cardholder registration protocol (Fig. 1) comprises three message exchanges between the cardholder and a certificate authority. In the first exchange, the cardholder requests registration and is given the certificate authority's public keys. In the second exchange, the cardholder supplies his credit card number, called the PAN, or Primary Account Number; he receives an application form suitable for the bank that issued his credit card. In the third exchange, the cardholder returns the completed application form; in addition, he delivers his public signature key and supplies a 20-byte secret number (the CardSecret). Finally, the cardholder receives a certificate that contains his public signature key and another 20-byte secret number,

[2] http://www.asn1.org

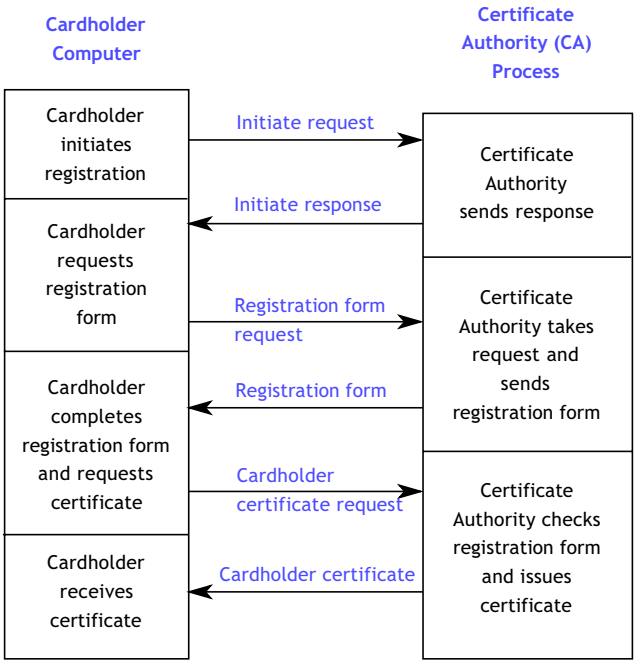


Fig. 1 Cardholder Registration

the PANSecret. The registration protocol for merchants is simpler: it has only two message exchanges and involves no credit card number.

Conceptually, cardholder registration is straightforward. Its chief peculiarities are that the cardholder is authenticated by the registration form containing the PAN, not by the knowledge of a secret key, and that long term keys can be created on the fly. The first point is critical for modelling and is discussed elsewhere [6]. The second point makes verification difficult.

5.1 Dynamic Creation of Long-Term Keys

Typical modelling of a public-private key pair associates each half to the agent holding it: there is a function mapping each agent's name to his public key. Thus each agent has precisely one public key, and therefore one private key. It simplifies the base step of all secrecy proofs: if the agent is not compromised, its private key is by definition not compromised. Each time a message is encrypted with a public key we can avoid the case split mentioned in the previous section: if the agent is not compromised, then the Spy cannot read messages encrypted using his public key. We can focus on the remaining trace and apply the inductive hypothesis.

If agents have more than one private key, case splits arise also on encrypting with public keys: one particular key could be compromised, or a

cardholder could use a key that is not compromised before step i but becomes compromised soon afterwards. Worse still, it is necessary to have a complex background theory on keys. At present we only have a type key where symmetric keys are distinguished by being self-inverse. As soon as we allow for the creation of asymmetric keys on the fly we must specify a lot more: one cannot generate the key of the root authority by chance, a public half of a key cannot be equal to a secret half of another asymmetric key, and so on and so forth. Also, a cardholder can, but is not obliged to, register a fresh key each time. This uncertainty makes the proofs hard and model checking simply impossible.

Indeed, in our first model of the cardholder registration protocol [6], we modelled these possibilities successfully. In the version presented below [5] we have reverted to the standard modelling approach, where one key pair for signature and one for encryption are syntactically associated to each agent, because it is more readable.

5.2 Key Dependency Chains

Another obstacle to verification—especially, proving secrecy—is SET's heavy use of digital envelopes. Digital envelopes can generate a trace where in message 1 there is a key encrypting a key for message 2, and so forth. To prove the secrecy of the last key one must prove the secrecy of all keys in the chain. Yahalom [23] and Kerberos [7] have a dependency chain of length one: one session key encrypts just one secret. With SET, the dependency chain has length two, or three if signing keys are generated dynamically. It may not sound like much of an increase, but it requires new proof techniques. Now that we have found these techniques, we can easily apply them to other protocols.

To cope with arbitrary dependency chains one needs to generalise the technique used for the Yahalom protocol [23]. We define a transitive relation specifying that in a given trace the loss of one key leads to the loss of another. In brief, the first key was used to encrypt the second key in some messages sent during that trace of events. This creates a dependency relation between the second key and the first key. Then we prove some lemmas that rule out dependencies or bound what can be lost. For instance, no key depends on the cardholder's secret key because no key is ever encrypted with the cardholder's public key. Another example: the secrecy of a key never used in a trace cannot depend on the secrecy of another key previously used in the trace. Also, if unused keys are lost to the Spy, then they must be held by compromised agents.

We have chosen to define the key dependency chain specifically for the protocol under verification. This approach is practical, though a protocol-independent treatment may superficially seem more attractive. For example, the generic Isabelle theory of protocol messages defines a relation yielding the keys necessary to decrypt some message belonging to a given set of

messages; the definition is independent of any particular protocol, and we have used it extensively in the proofs about SET. But inductive proofs may produce intermediate subgoals that span through many pages. Our current choice improves simplification, avoiding some case splits. We define the relation to refer to the specific protocol steps that produce the dependency chain. The other protocol steps, no matter how complicates, are ruled out by construction.

This treatment of the relation is safe, since the proofs will reveal any errors. If our relation omits some dependency, then our lemma bounding the possible losses will be useless for proving other secrecy theorems. Moreover, the case that we are unable to prove will indicate which dependency was missed.

In the current model of cardholder registration, the chain links only three items: two symmetric keys and one nonce. If asymmetric keys can be generated on-the-fly, the chain can become longer and the bounding lemmas more complex. Having both on-the-fly generation of asymmetric keys and digital envelopes adds more than the sum of their complexities.

5.3 Modelling the Fifth Message

Let us consider these points more precisely. Here is the fifth message, *Cardholder Certificate Request*:

$$\begin{aligned} 5.\ C \rightarrow CA : {} & \mathsf{Crypt}_{\mathrm{KC3}}(m,\ S), \\ & \mathsf{Crypt}_{\mathsf{pubEK}\ CA}(\mathrm{KC3}, \mathrm{PAN}, \mathrm{CardSecret}) \\ \text{where } m = {} & C,\ \mathrm{NC3},\ \mathrm{KC2},\ \mathsf{pubSK}\, C \\ \text{and } S = {} & \mathsf{Crypt}_{\mathsf{priSK}\, C}(\mathsf{Hash}(m, \mathrm{PAN}, \mathrm{CardSecret})) \end{aligned}$$

The cardholder chooses an asymmetric signature key pair. He gives the the public key, $\mathsf{pubSK}\, C$, and the number CardSecret to the certificate authority. This message is a digital envelope, sealed using the key KC3; it contains another key, KC2, which the certificate authority uses for encrypting the *Cardholder Certificate*:

$$\begin{aligned} 6.\ CA \rightarrow C : {} & \mathsf{Crypt}_{\mathrm{KC2}} \\ & (\mathsf{Sign}\ CA\ (C, \mathrm{NC3}, CA, \mathrm{NonceCCA}), \\ & \mathsf{Cert}_{CA}(\mathsf{pubSK}\, C, \mathrm{PANSecret}), \\ & \mathsf{Cert}_{\mathsf{RCA}}(\mathsf{pubSK}\ CA))) \\ \text{where PANSecret} = {} & \mathrm{CardSecret} \oplus \mathrm{NonceCCA} \end{aligned}$$

The certificate authority returns a certificate for the cardholder's public signature key. The certificate also includes the cryptographic hash of PANSecret. This 20-byte number is the exclusive-OR of the CardSecret and NonceCCA: a nonce chosen by the certificate authority. The purpose of these nonces is twofold: CardSecret will be used by the cardholder to confirm

purchases on top of the digital signature (the hash of CardSecret must be added for each payment instruction), PANSecret as a whole will be used to generate the "name" of the cardholder in the X.509 certificate format. In a nutshell, the name of the public key holder in the certificate will not be C but Hash(PANSecret,PAN).

The secrets KC3, KC2, NonceCCA form a dependency chain, requiring the new proof technique mentioned above. Removing the digital envelopes here would shorten the dependency chain—as would disposing with NonceCCA, as we recommend below.

Figure 2 presents the Isabelle specification of message 5. It is hard to read, but comparing it with the informal notation above conveys an idea of the syntax. The inductive definition consists of one rule for each protocol message, which extends a given trace. (Note that `#` is Isabelle syntax for the list "cons" operator. In the rule for message 5, the current trace is called *evs5*.) One of the rule's preconditions is that *CardSecret* must be fresh:

Nonce CardSecret $\notin$ *used evs5*

The nonce *NC3* and the two symmetric keys (*KC2* and *KC3*) must also be fresh. Other preconditions check that the cardholder has sent an appropriate instance of message 3 to the certificate authority and has received a well-formed reply. If the preconditions are satisfied, then *C* can generate the corresponding instance of message 5.

5.4 Security of the PANSecret

We did not discover any attacks against cardholder registration. However, we did discover a modification that would improve the protocol. Under reasonable assumptions, the PAN, PANSecret and other sensitive information remain secure. Among the reasonable assumptions is that certificate authorities are not compromised. Though this might be argued about a financial institution as such, it might be false about the institution's outsourced software. Here is a flaw: the PANSecret is computed as the exclusive-OR of CardSecret and NonceCCA, which gives the certificate authority full control over its value. One would like to be able to trust the certificate authorities, but banks have issued insecure Personal Information Numbers [1, p. 35]:

> One small upper-crust private bank belied its exclusive image by giving all its customers the same PIN. This was a simple programming error; but in another, more down-market institution, a programmer deliberately arranged things so that only three different PINs were issued, with the idea that this would provide his personal pension fund.

The remedy is trivial: compute the PANSecret by hashing instead of exclusive-OR. Another remedy is to leave its choice entirely to the cardholder's computer—after all, it exists for the cardholder's protection. If two

```
⟦evs5 ∈ set_cr;  C = Cardholder k;
 Nonce NC3 ∉ used evs5; Nonce CardSecret ∉ used evs5;
 NC3 ≠ CardSecret;
 Key KC2 ∉ used evs5; KC2 ∈ symKeys;
 Key KC3 ∉ used evs5; KC3 ∈ symKeys; KC2≠KC3;
 Gets C {|sign (invKey SKi) {|Agent C, Nonce NC2, Nonce NCA|},
          cert (CA i) EKi onlyEnc (priSK RCA),
          cert (CA i) SKi onlySig (priSK RCA)|}
      ∈ set evs5;
 Says C (CA i)
      {|Crypt KC1 {|Agent C, Nonce NC2, Hash (Pan (pan C))|},
       Crypt EKi {|Key KC1, Pan (pan C),
                           Hash {|Agent C, Nonce NC2|}|}|}
      ∈ set evs5⟧
⟹ Says C (CA i)
         {|Crypt KC3
          {|Agent C, Nonce NC3, Key KC2, Key (pubSK C),
           Crypt (priSK C)
             (Hash {|Agent C, Nonce NC3, Key KC2,
                    Key(pubSK C), Pan(pan C), Nonce CardSecret|})|},
          Crypt EKi {|Key KC3, Pan (pan C), Nonce CardSecret|}|}
    # evs5 ∈ set_cr
```

Fig. 2 Cardholder Registration in Isabelle (Message 5)

nonces are needed, one (PANSecret) disclosed to the Payment Gateway and another (CardSecret) disclosed only to a certificate authority, then let the cardholder generate both of them.

This modification would eliminate NonceCCA, and with it, the need to encrypt message 6, which would contain only public-key certificates. We could dispense with the key KC2 and eliminate the dependency chain KC3, KC2, NonceCCA. These changes would make the protocol simpler and more secure against a compromised certificate authority.

6 Verifying the Purchase Protocols

A SET purchase can involve three protocols: purchase request, payment authorisation, and payment capture. The first two of these often behave as a single protocol, which is how we model them. (We have yet to investigate payment capture.) The protocol is too complex to present here in full. Even the means of identifying the transaction is complicated. The cardholder and merchant may each have an identifying number; sometimes a third number is chosen. The choice of method is actually left open by SET designers. For the sake of simplicity, we discard all but one of the identification options, and use the merchant's transaction identifier.

The essential parameters of any transaction are the *order description* (presumably a text string) and the *purchase amount*. The cardholder forms

a dual signature on the order information and payment information, as outlined in Sect. 2, and sends it to the merchant. The merchant forwards the payment information, under his signature, to the payment gateway. Only the payment gateway can read the account details, which include the PAN and the PANSecret. If they are acceptable, he replies to the merchant, who confirms the transaction with the cardholder.

Other details of our model include an event to model the initial shopping agreement, which lies outside SET. Our model includes also the possibility of unsigned purchases. These allow unregistered cardholders to use SET using a credit card number alone and offer little protection to merchants. SET perhaps offers this option in order to provide an upgrade path from SSL. The leanest execution, in which everything is signed, runs for 6 messages.

An example illustrates the complexity of the dual signature. Message 3 is the actual purchase request from the cardholder to the merchant.

$$3.\ C \to M : \mathrm{PIDualSign}, \mathrm{OIDualSign}$$

Here, the cardholder C has computed

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

LID_M and XID are unique (but guessable) transaction identifiers generated by the merchant's software; Chall_C and Chall_M are nonces; the remaining fields are all derived from PAN, PANSecret, and CardSecret.

Because of the hashing, all the information appears repeatedly. Although in the real world the hash of any message is a short string of bytes, in the formal model the hash of message X is literally $\mathsf{Hash}\,X$: a construction involving X. The formal model of message 3 involves massive repetition. Most digital envelopes involve hashing, causing further repetition. Figure 3 presents this message using Isabelle syntax.

The SET documentation did not specify what properties to prove. We specified them ourselves, based on our interpretation of the Business Description. Obviously, the PAN and PANSecret must remain secure. Each party to a purchase must be assured that the other parties agree on all the essential details: the purchase amount, the transaction identifier, the order description, and the names of the other agents.

```
⟦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|};
 PIDualSign = {|sign (priSK C) {|Hash PIData, Hash OIData|},
                EXcrypt KC2 EKj {|PIHead, Hash OIData|} PANData|};
 OIDualSign = {|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 {|PIDualSign, OIDualSign|} # evsPReqS ∈ set_pur
```

Fig. 3 The Signed Purchase Request Message

We proved most of these properties, and some proofs were easy. Indeed, there were few theorems whose proofs were intrinsically difficult. The sheer number of theorems and supporting lemmas was an obstacle—many results had separate versions for signed and unsigned purchases. The complexity of theorem statements, caused by the complicated SET message formats, was an obstacle. Not knowing precisely what to prove was a major obstacle: if we had problems proving an assertion, we had to decide whether to weaken it somehow, to look harder in the SET documentation for some omitted field, or to try harder with the proof itself.

A typical agreement guarantee states that when the merchant sees a dual signature (in a Purchase Request), he is assured that it originated with the cardholder. The formal proof, like the intuitive one, argues that only the cardholder knows his private signature key. The proof uses induction, as usual, and applies three easily-proved technical lemmas. This theorem is important: by verifying the dual signature, specifically the transaction identifier XID, the merchant can be assured that he and the cardholder agree on the details of the purchase. The agreement guarantees between other pairs of agents are also easy to prove. However, the total effort, with the obstacles mentioned above, is considerable.

Secrecy proofs are always difficult. Simplification has to be set up carefully, or the subgoals will blow up exponentially. Lemmas of a peculiar form must be proved by induction. Fortunately, the necessary techniques appear to be similar for all protocols. We proved the secrecy of the session keys used in the digital envelopes and of nonces such as PANSecret. Secrecy of the PAN involves two theorems, depending upon whether the Purchase Re-

quest is signed or unsigned. We did not have to introduce new relations, as we did for Cardholder Registration.

A concern that emerged from our proof efforts is that many guarantees for cardholders and payment gateways depend upon the assumption that merchants are not compromised. This is not due to lack of effort on our side, but to an apparently unrelated feature of the design: the payment gateway is chosen by the merchant alone during the SET initiation process. His public and private keys are often used but his name is not confirmed (for example in a digital signature) either by the cardholder or by the merchant at any later stage. Thus, the payment gateway cannot be certain that the cardholder intended him to take part in the transaction. Message 3 involves six copies of the field XID (transaction identifier) and nine copies of the field PurchAmt (purchase amount), but it never mentions the identity of the intended payment gateway.

If the merchant is dishonest, there cannot be any guarantee that the right payment gateway has been selected. Furthermore, it is impossible to prove the agreement between the name and key of the payment gateway used by the cardholder and the name and key of the payment gateway authorizing the transaction. Although the failure of this property is disappointing, it does not appear to allow a significant attack. It could only be exploited by a rogue payment gateway, who could induce an anomalous execution on another gateway. However, presumably a rogue gateway would prefer the silent harvesting of credit card numbers to causing visible SET malfunctions. Thus, we reject the dualistic view that every protocol is either correct or vulnerable to attack. Anomalous executions that do little harm cannot be called attacks.

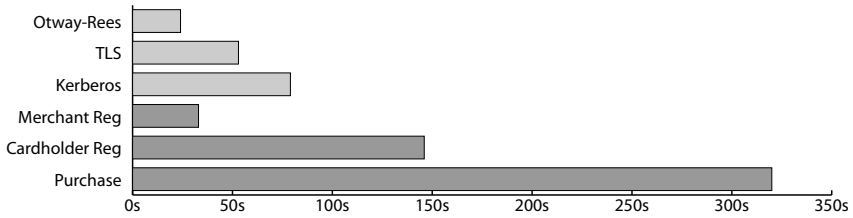
Digital envelopes complicate the statements of many guarantees. Agreement among principals obviously refers to important fields such as the order description and purchase amount. While we certainly hope the two parties will agree on which session key was used in a digital envelope, that property does not seem to be essential. We decided not to prove agreement on session keys because the value of this result did not justify the effort needed to prove it. Loosely speaking, we have proved that the keys on all locks (symmetric keys in digital envelopes) remained secret, that the contents of the luggage remained secret and unchanged, that the sender was authentic, but we have not proved that the luggage arrived with the same locks with which it was packed.

7 Conclusions

Our study demonstrates that enormous protocols such as SET are amenable to formal analysis. Such work is challenging, however. Understanding the documentation and defining a formal model can take months. Unfortunately, we did not record how much time we devoted to the various tasks and have to rely on memory. A student, Piero Tramontano, devoted about 28 weeks

to understanding SET under our supervision. While he concentrated on Cardholder Registration, much of this time was devoted to understanding the fundamentals of SET in general. Completing the Cardholder Registration proofs required the dependency relation described in Sect. 5.2 and took perhaps two weeks. Merchant Registration is simpler than Cardholder Registration and may have needed two weeks for its modelling and verification. For the Purchase phase, we may have devoted eleven weeks. These numbers are very approximate. We recall that modelling took longer than proof.

The proofs are still difficult. Isabelle may present the user with subgoals that are hundreds of lines long. Diagnosing a failed proof requires meticulous examination of huge and unintuitive formulae, where all abbreviations have been fully expanded. Such monstrosities impose a heavy burden on the computer too. A simplification step can take 10 or 20 seconds on a 1.8GHz processor. The bar chart shows the runtime required to execute the proofs for several protocols on a 1.8GHz machine. There are three SET protocols (dark shading) and three others (light shading). This data is suggestive rather than compelling, because minor changes to a proof script can cause substantial changes to the required runtime. It suggests that merchant registration is very simple. Cardholder registration requires more effort, partly because it is longer and partly because it demands more secrecy proofs. The purchase protocol is by far the most difficult one.



It is not clear whether model checking could cope with this protocol's complexity. Specialized verification tools are more powerful than Isabelle, but they are less flexible. Even for Isabelle, the burden on the human verifier is too high to be increased further.

The single greatest advance would be a method of abstraction allowing constructions such as the digital envelope to be verified independently. We could then model these constructions abstractly in protocol specifications. In the case of SET, we could replace all digital envelopes by their abstract version. Assertions would become more concise; proofs would become much simpler. Unfortunately, abstraction in the context of security is ill understood and can mask grave flaws [25].

The other advance depends on protocol designers: they should provide a Formal Protocol Definition worthy of the name. It should precisely specify several things:

1. a version of the message flow comprising the security features only,

2. a clear separation of features necessary to patch real-word cryptography (such as salt, which thwarts dictionary attacks) from abstract primitives (such as perfect hashing, encryption and signature),
3. the protocol's precise objectives, expressed as operational guarantees to each party,
4. the protocol's operating environment, including the threat model.

Notice that we are not advocating that formal verification should be used during the design (though it might be desirable eventually), nor that the Formal Protocol Definition should employ a logical formalism (designers would disagree on which one to use). We merely insist that the protocol documentation should clearly specify the items mentioned above. The implementers and maintenance staff would also benefit from a clear and precise specification. At present, we are forced to reverse engineer the protocol's core design from its documentation, and we have to guess what the protocol is supposed to achieve.

Acknowledgements. In the UK, the EPSRC grant GR/R01156/R01 *Verifying Electronic Commerce Protocols* supported Bella and Paulson. In Italy, MURST and CNR grants supported Massacci.

References

1. R. Anderson. Why cryptosystems fail. *Comm. of the ACM*, 37(11):32–40, Nov. 1994.
2. N. Asokan, M. Schunter, and W. M. Optimistic protocols for fair exchange. In *Proc. of the 4th ACM Conf. on Comm. and Comp. Sec. (CCS-97)*, pages 7–17. ACM Press and Addison Wesley, 1997.
3. G. Bella, F. Massacci, L. Paulson, and P. Tramontano. Making sense of specifications: the formalization of set (extended abstract). In B. Christianson, B. Crispo, and M. Roe, editors, *Proceedings of the 2000 Security Protocols Workshop*, Lecture Notes in Comp. Sci., pages 74–81. Springer-Verlag, 2000.
4. G. Bella, F. Massacci, and L. C. Paulson. The verification of an industrial payment protocol: The SET purchase phase. In V. Atluri, editor, *9th ACM Conference on Computer and Communications Security*, pages 12–20. ACM Press, 2002.
5. G. Bella, F. Massacci, and L. C. Paulson. Verifying the SET registration protocols. *IEEE J. of Selected Areas in Communications*, 21(1):77–87, 2003.
6. G. Bella, F. Massacci, L. C. Paulson, and P. Tramontano. Formal verification of cardholder registration in SET. In F. Cuppens, Y. Deswarte, D. Gollman, and M. Waidner, editors, *Computer Security — ESORICS 2000*, volume 1895 of *Lecture Notes in Comp. Sci.*, pages 159–174. Springer, 2000.
7. G. Bella and L. C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In Quisquater et al. [24], pages 361–375.
8. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426:233–271, 1989.

9. E. Cohen. TAPS: A first-order verifier for cryptographic protocols. In *Proc. of the 13th IEEE Comp. Sec. Found. Workshop*, pages 144–158. IEEE Comp. Society Press, 2000.
10. V. Kessler and H. Neumann. A sound logic for analysing electronic commerce protocols. In Quisquater et al. [24].
11. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems: second international workshop, TACAS '96*, volume 1055 of *Lecture Notes in Comp. Sci.*, pages 147–166. Springer, 1996.
12. G. Lowe and M. L. Hui. Fault-preserving simplifying transformations for security protocols. *J. of Comp. Sec.*, 9(3-46), 2001.
13. Mastercard & VISA. *SET Secure Electronic Transaction: External Interface Guide*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
14. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Business Description*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
15. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Formal Protocol Definition*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
16. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Programmer's Guide*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
17. C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer. In *SSP-99*, pages 216–231. IEEE Comp. Society Press, 1999.
18. C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1):44–54, 2003.
19. C. Meadows and P. Syverson. A formal specification of requirements for payment transactions in the SET protocol. In R. Hirschfeld, editor, *Proceedings of Financial Cryptography 98*, volume 1465 of *Lecture Notes in Comp. Sci.* Springer-Verlag, 1998.
20. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.
21. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. of Comp. Sec.*, 6:85–128, 1998.
22. L. C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Trans. on Inform. and Sys. Sec.*, 2(3):332–351, 1999.
23. L. C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *J. of Comp. Sec.*, 9(3):197–216, 2001.
24. J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors. *Computer Security — ESORICS 98*, volume 1485 of *Lecture Notes in Comp. Sci.* Springer, 1998.
25. P. Ryan and S. Schneider. An attack on a recurive authentication protocol. a cautionary tale. *Inform. Processing Lett.*, 65(15):7–16, 1998.
26. S. D. Stoller. A bound on attacks on payment protocols. In *Proc. 16th Annual IEEE Symposium on Logic in Computer Science (LICS)*, June 2001.