Verifying the SET Protocol: Overview

Lawrence C. Paulson

Computer Laboratory, University of Cambridge
JJ Thomson Avenue, Cambridge CB30 0FD, England
lcp@cl.cam.ac.uk

Abstract. The project to verify SET, an e-commerce protocol, is described. 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. Brief examples are taken from the registration and purchase phases. The protocol does not completely satisfy its goals, but only minor flaws have been found. The primary outcome of the project is experience with handling enormous and complicated protocols.

1 Introduction

SET (Secure Electronic Transaction) is an e-commerce protocol devised by Visa and MasterCard. It enables credit card holders to pay for purchases while protecting their personal information, which includes both their account details and their purchasing habits. Most research on protocol verification focuses on simple protocols that have simple objectives. One reason for verifying SET is to demonstrate that verification technology is mature enough to cope with the demands of a huge, complex industrial protocol.

Protocol verification techniques fall into several categories. A general-purpose model-checker can verify protocols, as pioneered by Lowe and his colleagues at Oxford [7]. A general-purpose proof tool can also be effective, as in my work [13]. 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 [11], which has deductive capabilities. Cohen's TAPS processes the protocol specification and verifies the desired properties using a resolution theorem prover [6].

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.

My colleagues and I have verified [2,3] the main phases of the SET protocol using the inductive approach and the theorem prover Isabelle. A substantial proportion of the effort was devoted to understanding the documentation rather than to proving properties. This paper is a brief overview of the project, referring to other papers that describe the separate tasks.

The paper begins by outlining the SET protocol (Sect. 2). It briefly introduces the inductive approach and Isabelle (Sect. 3). It discusses the issues we faced in converting the documentation into a formal model (Sect. 4). It outlines our proofs of the registration protocols (Sect. 5) and the payment protocol (Sect. 6). 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 security details and the public half of his proposed signature key. Registration allows the authorities to vet an applicant, who if approved receives a certificate confirming that his signature key is valid. 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 (essentially 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 while allowing all parties to confirm that they are handling the same transaction. The method is simple: each party receives the hash of the withheld information. The cardholder signs the hashes of both the order information and the payment information. Each party can confirm that the hashes in their possession agrees with the hash signed by the cardholder. In addition, the 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; he needs the cardholder's signature key and a secret number that the cardholder receives upon registration. The criminal would have better luck with traditional frauds, such as ordering by telephone. It is a pity that other features of SET (presumably demanded by merchants) weaken these properties. A merchant can be authorized to receive credit card numbers and has the option of accepting payments given a credit card number alone.

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 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. Because of this complexity, much of which is unnecessary, the protocol is hardly used. However, SET contains many features of interest:

– The model is unusual. In the registration protocols, the initiator possesses no digital proof of identity. Instead, he authenticates himself by filing a registration form

whose format is not specified. Authentication takes place outside the protocol, when the cardholder's bank examines the completed form.
– The dual signature is a novel construction. The partial sharing of information among three peers leads to unusual protocol goals.
– SET uses several types of *digital envelope*. 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 many symmetric keys complicate the reasoning. Most verified protocols distribute just one or two secrets.

3 Isabelle and Inductive Protocol Verification

My colleagues and I used the Isabelle theorem prover with the inductive approach to protocol verification. It is not clear that model checking could cope with this protocol's complexity. Specialized verification tools are more powerful than Isabelle, but less flexible. Most protocols, even esoteric ones like non-repudiation and fair exchange protocols, involve the standard cast of characters: Alice, Bob, 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 inductive approach [13] 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. An execution comprises any number of attempted protocol runs and is a trace of message transmissions and other events. A standard theory of messages and their operations underlies these inductive models. Safety properties are proved by induction on traces. For example, we can prove that any trace containing a particular event x must also contain some other event y; such properties can express authentication or agreement. 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. Huge case analyses can arise. Despite the difficulties, we can use established techniques and tools in our attempt to prove secrecy.

The model includes a set of honest agents. Typically we can prove (perhaps optimistically) that their long-term keys cannot become compromised. 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. If a key becomes compromised then the spy can read all ciphertexts encrypted using that key, and if it has been used to encrypt other keys, then the consequential losses cascade. Proving secrecy in this situation requires special techniques, which I have presented for the Yahalom protocol [15] and applied also to Kerberos [5].

Messages in our model have types. A nonce can never equal an agent name or a session key, for example. Such assumptions can be defended: in the real world, different kinds of items are likely to have different lengths. However, our model does not allow reasoning about operators like exclusive-OR. Because $(X \oplus Y) \oplus Y = X$, exclusive-

OR can yield a result of essentially any type. Reasoning about exclusive-OR probably requires a bit-level formalization.

Isabelle/HOL [12] 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. We have applied the inductive approach to a wide range of protocols, including industrial ones such as Kerberos and TLS [14].

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* [8], the *Programmer's Guide* [10], and the *Formal Protocol Definition* [9]. SET is defined using Abstract Syntax Notation One (ASN.1).[1] The programmers 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 programmers 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 the syntax without the semantics. We derived our model from the programmer's guide.

The enormous size and complexity of the SET message formats demanded simplification. As we have discussed elsewhere [4], this was not always straightforward. A field might be described as optional and yet seem to play an essential role. Additional simplifications were necessary, forcing us to decide what constituted SET's core feature set. One detail that we eliminated was payment by instalments. Most payment cards provide payment by instalments anyway, so SET does not have to provide a similar mechanism. However, critics might reject this reasoning.

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. For example, Lowe's famous attack [7] against the Needham-Schroeder public-key protocol relies on the possibility that insiders can be compromised. However, Needham-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 undefined parts. 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.

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

5 Verifying The Registration Protocols

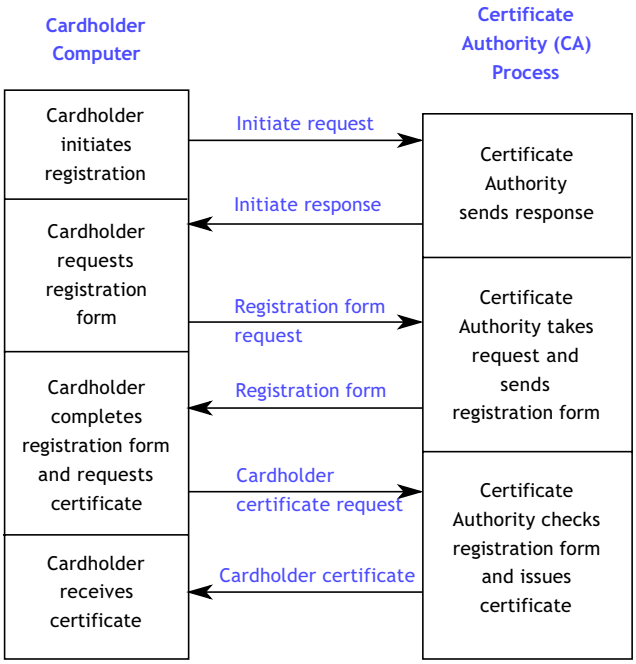


Fig. 1. Cardholder Registration

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, the PANSecret. The registration protocol for merchants is simpler: it has only two message exchanges and involves no credit card number. My colleagues and I verified both cardholder registration and merchant registration. Cardholder registration is the one I discuss below.

Conceptually, cardholder registration is straightforward. Its chief peculiarity is that the cardholder is authenticated by the registration form, not by the possession of a secret key. The protocol as defined in SET however is difficult to verify, mainly because it employs digital envelopes. While Yahalom and Kerberos have a dependency chain of length one — one session key encrypts just one secret — with digital envelopes

the dependency chains could be arbitrarily long. (In the current model of cardholder registration, the chain links only three items, though at one point it was longer.)

I was able to generalise the previous technique [15] in order to cope with arbitrary dependency relationships. A relation must be defined in higher-order logic, identifying the protocol events that cause one secret to depend upon another. (This relation is necessarily transitive.) Lemmas must be proved, saying in effect that the loss of a key can cause no losses other than the obvious ones given by the relation. Such lemmas put a bound on the consequential losses. The proofs employ induction and the intermediate subgoals can be many pages long.

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

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

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

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

The CA 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 CA. The cardholder must use the PANSecret to prove his identity when making purchases.

Cardholder registration does not have to be this complicated. Since the cardholder has a private signature key, why does he also need the PANSecret? If he really does need the PANSecret to prove his identity, why must the CA contribute to its calculation through NonceCCA? The point of such an calculation is to avoid sending the secret across the network, but the cardholder must disclose the PANSecret each time he makes a purchase. Eliminating NonceCCA would eliminate 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, as we shall see.

Figure 2 presents the Isabelle specification of message 5. You will find it 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's syntax for the list "cons" operator. In message 5,

the current trace is called *evs5*.) One of the rule's preconditions is that *CardSecret* must be fresh:

```
Nonce CardSecret ∉ 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 CA and has received a well-formed reply. If the preconditions are satisfied, then *C* can generate the corresponding instance of message 5.

```
⟦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)

My colleagues and I did not discover any attacks against cardholder registration. Under reasonable assumptions, the PAN, PANSecret and other sensitive information remain secure. However, merely by inspection, I observed a flaw. The PANSecret is computed by exclusive-OR, 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.

6 Verifying the Purchase Phase

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. 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. 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 above 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 Primary Account Number and the PANSecret. If they are acceptable, he replies to the merchant, who confirms the transaction with the cardholder.

A look at message 3 illustrates the complexity of the dual signature:

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

Here, the cardholder C has computed

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

Figure 3 presents this message using Isabelle syntax. 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.

Other details of our model include a dummy message to model the initial shopping agreement, which lies outside SET. Our model includes the possibility of unsigned

```
⟦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

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.

Because the SET documentation did not tell us what properties to prove, we specified them ourselves. Obviously, the PAN and PANSecret must remain secure. Correctness also means that each party to a purchase must be assured that the other parties agree on all the essential details: namely, the purchase amount, the transaction identifier, the order description, and the names of the other agents. We were able to prove most of these properties. Digital envelopes cause further problems, however. 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 is not essential. Given the choice of either devoting much effort to proving agreement on session keys or ignoring them, I adopted the latter course.

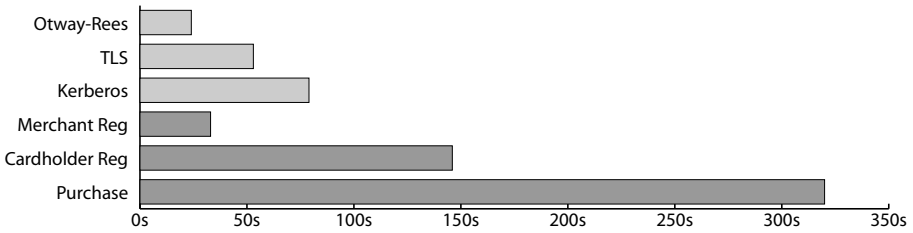
Agreement fails in one important respect: the payment gateway cannot be certain that the cardholder intended him to take part in the transaction. Message 3 involves six copies of the field XID (transaction identifier) and nine copies of the field PurchAmt (purchase amount), but it never mentions the identity of the intended payment gateway! 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 would presumably prefer harvesting credit card numbers to causing anomalous SET executions. Thus, we must reject the dualistic view that every protocol is either correct or vulnerable to attack. Anomalous executions that do little harm cannot be called attacks.

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. Some assertions are too long to be comprehensible, comprising a dozen or two lines of formalism. Whether those assertions are specifications or theorem statements, their incomprehensibility raises the possibility that they could be misinterpreted.

During an interactive proof, Isabelle may present the user with subgoals that are hundreds of lines long. Such monstrosities impose a heavy burden on the computer. A simplification step can take 10 or 20 seconds on a 1.8 gigahertz processor. Diagnosing a failed proof requires meticulous examination of huge and unintuitive formulae, where all abbreviations have been fully expanded.

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 dramatic 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 phase is twice as difficult again.



I doubt that existing methods can cope with protocols that are more complicated than SET. (Perhaps such protocols should not exist.) 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 simple encryptions. Assertions would become more concise; proofs would become much simpler. Abstraction in the context of security is ill understood, however, and can mask grave flaws [16].

The other advance can happen now, if protocol designers will co-operate. They should provide a Formal Protocol Definition worthy of the name. It should not employ a logical formalism — people would disagree on which one to use — but it should precisely specify several things:

1. an abstract version of the message flow, comprising the core security features only
2. the protocol's precise objectives, expressed as guarantees to each party
3. the protocol's operating environment, including the threat model

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. The SET verification is joint work with Giampaolo Bella, Fabio Massacci and Piero Tramontano. Bella also commented on this paper. The EPSRC grant GR/R01156/R01 *Verifying Electronic Commerce Protocols* supported the Cambridge work. In Italy, CNR and MURST grants supported Massacci.

References

1. R. Anderson. Why cryptosystems fail. *Comm. of the ACM*, 37(11):32–40, Nov. 1994.
2. 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.
3. 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.
4. 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*, LNCS 1895, pages 159–174. Springer, 2000.
5. G. Bella and L. C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Computer Security — ESORICS 98*, LNCS 1485, pages 361–375. Springer, 1998.
6. 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.
7. 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*, LNCS 1055, pages 147–166. Springer, 1996.
8. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Business Description*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
9. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Formal Protocol Definition*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
10. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Programmer's Guide*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
11. 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.
12. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002. LNCS Tutorial 2283.
13. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. of Comp. Sec.*, 6:85–128, 1998.
14. L. C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Trans. on Inform. and Sys. Sec.*, 2(3):332–351, 1999.
15. L. C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *J. of Comp. Sec.*, 9(3):197–216, 2001.
16. P. Ryan and S. Schneider. An attack on a recurive authentication protocol. a cautionary tale. *Inform. Processing Lett.*, 65(15):7–16, 1998.