Making Sense of Specifications: the Formalization of SET (Extended Abstract)

Giampaolo Bella[1], Fabio Massacci[2,3], Lawrence C. Paulson[1], and Piero Tramontano[3]

[1] Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG, England
e-mail: {gb221,lcp}@cl.cam.ac.uk

[2] Dip. di Ingegneria dell'Informazione, Universià di Siena
via Roma 56, 53100 Siena, Italy
e-mail: massacci@dii.unisi.it

[3] Dip. di Informatica e Sistemistica, Università di Roma I "La Sapienza"
via Salaria 113, 00198 Roma, Italy
e-mail: {massacci,tramonta}@dis.uniroma1.it

1 Introduction

The last ten years, since the seminal work on the BAN logic [6], have seen the rapid development of formal methods for the analysis of security protocols. But security protocols have also developed rapidly, becoming more and more complex. Protocols for electronic commerce are the *béte noir*: six pages were enough to describe the Needham-Schroeder protocol in 1978 [15], six hundred pages were not enough to describe the SET protocol of VISA and Mastercard [11–13] twenty years later.

This increase in complexity is a challenge to both protocol designers and protocol verifiers. The sheer size of the documentation makes it likely to contain bugs and inconsistencies. The same size makes it difficult to analyze of protocol formally. Model-checking approaches are likely to founder in the enormous state space, while inductive proof may become unfeasible in the presence of so many subgoals to consider.

After Kailar's [8] analysis of simple electronic commerce protocols, there have been attempts to model more realistic protocols such as Kerberos [2], TSL/SSL [18], Cybercash coin-exchange [4, 5], and C-SET (a version of SET reduced for smart cards) [3]. However, to the best of our knowledge, no attempt has been made to formally analyze the SET protocol. This paper describes the work we have done in this direction.

2 The SET Protocol

The SET protocol [11] has been proposed by VISA as a standard protocol for securing electronic transactions on the Web. It has been standardized by a large consortium including the major credit card companies (VISA, Mastercard, American Express) and software corporations (Microsoft, Netscape etc.). SET aims to protect sensitive card-holder information but does not support non-repudiation.

The overall architecture of SET is based on a rooted hierarchy of *Certification Authorities* (CA). The top level is a trusted root followed by centralized certification authorities corresponding to credit card brands. Two levels down are certification authorities (corresponding to banks) that deal with customers. The task of these CAs is to provide customers with public keys and digital signature certificates for signature and encryption. It is worth noting that CAs do not need to have access to the private keys of customers, who must generate and safeguard their own keys.

Customers of this systems can be grouped in two major classes: *Card-holders* (C) and *Merchants* (M). Besides them, it is foreseen to have *Payment Gateways* (PG) to play the role of clearing-houses [16]: their task is to settle the payment requests made by Merchants and Cardholders when buying goods. The protocol descriptions are complex [11–13] and make heavy references to the PKCS standards by RSA Security [19, 20].

The cryptography algorithms used in SET are of three main types:

1. public key cryptography for encryption
2. public key cryptography for signature
3. symmetric key encryption (typically for session keys and digital envelopes)
4. hashing and message digests

The first two kinds of keys are deemed to be different, despite being based on the same algorithm. These basic types are composed in various forms: for instance one may use a strong signature that signs a whole document or a weak one that signs only a digest. Or one may prepare a digital envelope by encrypting (with a public key) a symmetric key and the hash of a message plus nonces and then concatenate this message with proper envelope which is the actual message plus the hash of symmetric keys, etc.

The SET protocol consists of five phases. The first two phases are used by the agents participating in the protocol to register their keys and getting the appropriate certificates. The remaining three phases constitute the electronic transaction itself.

Card-holder Registration. This is the initial step of the protocol for card-holders. The agent C sends to a certification authority CA the information on the credit card he wants to use, the CA replies with a registration form, which is compiled by C and sent back together with the signing key that C wants to register. Then the CA checks that the credit card is valid (this step is outside the protocol) and releases the certificate for C who stores it for future use. All this information (such as credit card information) must be protected and this makes the protocol steps complicated. A couple of things are worth noticing:

– a user may register as many public keys as he wants to;

– the user identity is not stored in the certificate but just an indirect information on its primary account number (PAN): that is, its credit card number.

Merchant Registration. This phase is analogous for the Merchant M. In contrast with the card-holder registration phase, M can register a public key for encryption and a public key for signature. The process is shorter because there is no confidential information to be protected.

Purchase Request. We reach this phase if the card-holder has decided to buy something. C sends to M the order and the instruction for payment. M process the orders and forwards some of the payment instruction to the PG. This last step is needed because SET aims to keep some card-holder information confidential. The PG, in cooperation with credit card issuers, banks and other institutions, checks that everything is all right (this step is outside SET). Then we are ready for the next stage.

Payment Authorization. The PG sends the payment authorization to M and M sends to C the confirmation and possibly the purchased goods. C acknowledges the result and M passes to the next stage.

Payment Capture. In this last phase, M sends to the PG one or more authorizations and the corresponding card-holder authorizations. The PG checks that everything is satisfactory and replies back to M. The actual funds transfer from C to M is done outside the protocol by the credit card company.

3 General Principles Behind the Formalization

The formal analysis of a complex protocol like SET requires two steps. The first step tries to obtain a high level description of the protocol, eliminating some of the technology dependent features. It should clearly describe the operating environment and the semantics of the messages.

Once a protocol reaches the stage of an RFC (Request for Comments), it is often specified in an elaborate but unsatisfactory manner. The meanings of the fields of a message are only given informally. Often it is hard to know precisely what the recipient of a message will do with it or when a particular message will be sent. It is often unclear what counts as a successful outcome. Such ambiguities can be resolved by discussion and reflection, but there is no guarantee that other readers of the specification will interpret it in the same way.

When it comes to syntax, we have the opposite problem: too much detail. The fields of each message are specified down to the last bit. Contemporary theorem-proving technology is not up to the job of reasoning about real-world cryptographic algorithms. For an abstract understanding of protocol correctness, it does not matter whether hashing is done using MD4 or SHA-1; we merely expect hashing to be collision-free and non-invertible. Similarly, the RFC will name specific cryptographic algorithms, when for abstract purposes it only matters that encryption guarantees confidentiality and (perhaps) integrity. We have to assume perfect encryption, even though this may hide some protocol flaws. Having abstracted away the particular algorithms, we can also simplify those parts of a protocol concerned with choosing the algorithm, though we need to ensure that such pragmatic fields are received intact.

We make the further assumption that there is no type confusion. For instance, if two certificates have the same form except that one has a key where another has a nonce, then they can never coincide in our model. This aspect agrees with the real world, since typically the two certificates differ in other details, such as the length of the key/nonce field. However, for each protocol that we model, we have to check that the certificate formats indeed satisfy this assumption. Since data sent in clear can never give security guarantees, type confusion is relevant only for integrity-protected data.

As usual in protocol verification, we assume a worst-case environment. An active attacker controls the network and also possesses a long-term key and other credentials. We can model weaker assumptions, such as a passive eavesdropper, if the protocol is intended to work in that type of environment.

4 On Modelling with Inconsistencies

The second step in the formalization process involves a deeper understanding of the specifications and, possibly, a further simplification of the protocol to obtain a manageable analysis. We eliminate protocol steps or messages that are not relevant to a particular claim. For instance, recall that we assume that no message can be decrypted without the key. Then, if the specification says that the only purpose of a nonce in a message is to avoid a dictionary based attack and that it will not be checked by the recipient then we can dispense with this nonce. So the obvious way to analyze of SET is as follows:

– eliminate all optional parts (which should always give a SET compliant protocol)
– eliminate all parts which are introduced because of non-perfect cryptography (which are inessential for the formal analysis)

As one would expect, it is difficult to understand the specification of a complex protocol such as SET. Moreover, the SET designers also made clear that "if a discrepancy is found between the Formal Protocol Definition and the other books, the Formal Protocol Definition should be followed."

We found that this does *not* work. The first major problem is that the Formal Specification [13] is not sufficient to clarify key issues: it does not specify what agents has to do with messages (do they have to check nonces?) or how certificates are handled. One must look elsewhere, such as in the Programmer's Guide [12] or in the Business Description [11].

Unfortunately, this is not a safe procedure. The Business description [11], and in particular its figures, are most misleading. For instance, the description of the Payment Authorization phase suggests that each merchant will receive the primary account number of the customer, whereas this is not the case. So we were forced to use the Programmer's Guide [12] as the ultimate reference.

The second, more serious, problem is that many "options" are not really optional. So one cannot omit the options and obtain a working protocol. For example, the specification of the Card-holder registration phase specifies that if the software of the card-holder can use symmetric encryption then it should encrypt a certain message using a symmetric key and send the key in another message. The field where the key is stored is tagged as optional. Even though everything seems to imply that it can be left out, it cannot be. Elsewhere it is said that if the key field is missing then the Certification Authority reports an error. To solve the inconsistency the only way is to fall back on the Programmer Guide [12] where, however, things are only informally explained.

Another example is the optional use of certificates. For instance, the card-holder registration phase is structured in such a way that it can be interrupted and resumed much later. (We do not know whether this is intended.) Thus, the Certification Authority always sends the certificates (or better its thumb-prints) for the signing keys. The rationale that we figured out is that the CA may want to offer the card-holder the possibility of always getting the most recent key there is. However, certificates for encryption are always optional and sometimes even missing. Yet, the card-holder can receive a message encrypted with a public key for which he may never have seen a certificate if we leave options out.

Other inconsistencies can be found in the use of nonces, which are sometimes sent, sometimes are optional and, worse still, even when they are sent they may not be checked in subsequent messages (so one may wonder what they are for).

These inconsistencies could be solved easily if there were a precise specification of what each step of the protocol should accomplish. Unfortunately, there is no equivalent of the specifications as set forth by Abadi and Needham [1], Gollman [7] and Lowe [9] for authentication protocols.

The broad goals are clear: at the end of the Card-holder registration phase, the Card-holder C should have registered his public key his chosen Certification Authority. They are also spelled out in the Business Description [11, p. 4]. However, many questions remain unanswered:

1. How many keys can a card-holder register with a certification authority?
2. Can two different users register the same key with the same certification authority?
3. Should confidentiality of primary account numbers be protected?
4. Can be the registration phase be interrupted and resumed at any later stage?
5. Should each message be linked to a previous message to identify the protocol run?

Even such reasonably high level goals are not specified in the Protocol Description and, as we have already noticed, the Business Description can be misleading.

For instance, we would expect that Question 2 deserves a straightforward answer: No. Yet in the Business Description [11, pp. 25, 44] and in the Programmer's Guide [12, pp. 146-161], no such requirement is mentioned. So, it seems that SET lets two users have the same private key.

Question 3 is more intriguing. It is explained at great length in the Programmer's Guide description of the Card-holder Registration phase that the card-holder account will not be visible in the certificate. So, when modelling the protocol, this seems a crucial requirement. However, when looking at the Business description of another protocol step, when the Payment Gateway transmits the card-holder data to the Merchant for the payment authorization, the information about the Card-holder are transmitted in clear. In this case, we have decided to be on the safe side and stick to the Programmer's guide.

We can only prove properties of a protocol after we have arrived at a formal model of it. As we have indicated, we had only general indications from the SET book, so we had to rely on common sense.

5 The Isabelle Formalization

Notwithstanding the obstacles mentioned above, we have formally specified the SET Cardholder Registration phase (SET-CR in the sequel) using the inductive method [17, 18]. Compared with other protocols that have been analyzed formally, SET has some unusual features. These include (1) several certification authorities may participate in the protocol and (2) the cardholders are entitled to issue and submit their own keys for certification. The inductive method may be easily extended to account for these features.

The datatype for agents must allow for a single root certification authority and an unbounded number of certification authorities

```
datatype agent = RCA | CA nat | Friend nat | Spy
```

which requires minor updates to a few of the existing lemmas.

The protocols analyzed so far by practically any formal method [9, 10, 14, 17, 18, 21, 22] presuppose that every agent possesses long-term key. By contrast, SET-CR aims at establishing such association for card-holders. Some certification authority creates the association at the end of the protocol by means of a certificate. Therefore, we only define a pair of signature keys (one public, one private) for the root certification authority; each certification authority has two pairs, for signature and encryption. For the sake of convenience, we refer to these as *crucial keys.*

Existing models of key-distribution protocols issue session keys that are fresh. Our SET-CR model does not require freshness of the key pairs that a card-holder wants certified. He can submit any pairs that differ from the crucial keys, perhaps to different authorities in separate protocol runs.

6 Conclusions

The official descriptions of SET [11–13] describe the composition of messages in minute detail, while failing to provide a satisfactory semantics for those messages. A formal specification by the inductive method does provide an operational semantics for the protocol and identifies several ambiguities in the documentation.

Our work has focussed on the cardholder registration phase. We have unveiled some potentially dangerous omissions. Different agents may collude and register the same key with different authorities. The protocol will not prevent them from obtaining the requested certifications. We have not yet investigated the consequences of this scenario.

Our future work will cover the remaining phases of the protocol. Having formalized the specification, the greatest task is behind us. We expect to be able to derive further properties of SET with an acceptable amount of effort.

References

1. M. Abadi and R. M. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.
2. G. Bella and L. C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In *Proceedings of the 5th European Symposium on Research in Computer Security*, volume 1485 of *Lecture Notes in Computer Science*, pages 361–375. Springer-Verlag, 1998.
3. D. Bolignano. An approach to the formal verification of cryptographic protocols. In *Proceedings of the 3th ACM Conference on Communications and Computer Security (CCS-96)*, pages 106–118, 1996.
4. S. Brackin. Automatic formal analyses of two large commercial protocols. In *Proceedings of the DIMACS Workshop on Design and Formal Verification of Security Protocols*, page 14pp, September 1997.
5. S. Brackin. Automatically detecting authentication limitations in commercial security protocols. In *Proceedings of the 22nd National Conference on Information Systems Security*, page 18pp, October 1999.
6. M. Burrows, M. Abadi, and R. Needham. A logic for authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.
7. D. Gollmann. What do we mean by entity authentication? In *Proceedings of the 15th IEEE Symposium on Security and Privacy*, pages 46–54. IEEE Computer Society Press, 1996.
8. R. Kailar. Reasoning about accountability in protocols for electronic commerce. In *Proceedings of the 14th IEEE Symposium on Security and Privacy*, pages 236–250. IEEE Computer Society Press, 1995.
9. G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, pages 31–43. IEEE Computer Society Press, 1997.
10. G. Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6(18-30):53–84, 1998.
11. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Business Description*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
12. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Programmer's Guide*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
13. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Protocol Definition*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.

14. C. A. Meadows. Formal verification of cryptographic protocols: A survey. In *Advances in Cryptology - Asiacrypt 94*, volume 917 of *Lecture Notes in Computer Science*, pages 133–150. Springer-Verlag, 1995.
15. R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
16. D. O'Mahony, M. Peirce, and H. Tewari. *Electronic payment systems*. The Artech House computer science library. Artech House, 1997.
17. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
18. L. C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Transactions on Information and System Security*, 2(3):332–351, 1999.
19. RSA Laboratories, RSA Security - Available electronically at `http://www.rsasecurity.com/rsalabs/pkcs`. *PKCS-6: Extended-Certificate Syntax Standard*, 1993.
20. RSA Laboratories, RSA Security - Available electronically at `http://www.rsasecurity.com/rsalabs/pkcs`. *PKCS-7: Cryptographic Message Syntax Standard*, 1993.
21. P. Syverson and C. Meadows. A formal language for cryptographic protocol requirement. *Designs, Codes and Cryptography*, 7(xxx):27–59, 1996.
22. P. F. Syverson and P. C. van Oorschot. On unifying some cryptographic protocols logics. In *Proceedings of the 13th IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1994.