A ROLE-BASED SPECIFICATION OF THE SET PAYMENT TRANSACTION PROTOCOL

Hideki Sakurada
NTT Communication Science Laboratories,
NTT Corporation,
3-1 Morinosato- Wakamiya, Atsugi, Kanagawa, 243-0198 Japan
sakurada0theory.brl.ntt.co.jp

Yasuyuki Tsukada
NTT Communication Science Laboratories,
NTT Corporation,
3-1 Morinosato- Wakamiya, Atsugi, Kanagawa, 243-0198 Japan
tsukada@theory.brl.ntt.co.jp

Abstract In this paper, we define a language for specifying security protocols concisely and unambiguously. We use this language to formally specify the protocol for payment transactions in Secure Electronic Transaction (SET), which has been developed by Visa and MasterCard.

In our language, a protocol is specified as a collection of processes. Each process expresses the role of a participant. In the role-based specification, the components that a participant sees in a message can be stated explicitly. This is important in specifying protocols like that for the SET payment transactions because in such protocols some message components are encrypted and invisible to some participants.

We simplify the SET payment transaction protocol into the exchanges of six messages. Because our future goal is to formally analyze the security properties that Meadows and Syverson discussed, we make the simplified protocol contain the parameters used in their security properties. And we also refrain from excessive simplification. For example, we use dual signature in the payment request message as it is specified in the SET specification books, while most of the other works do not use it. Our specification can serve as a starting point for a formal analysis of the protocol.

Keywords: Formal methods, security protocols, electronic commerce

1. Introduction

Security protocols are used in distributed systems to protect the secrecy of messages and to identify users. It is well known that designing them is an error-prone task. The most significant issues concerning security protocols are that (1) attacks on them may succeed even without breaking the cryptographic algorithms used and that (2) it may be difficult to make sure of the correctness of a small protocol that involves exchanges of only a few messages. Some examples of protocol failures are presented in (Anderson and Needham, 1995; Clark and Jacob, 1997).

Formal methods can be used to analyze security protocols. With the methods, protocols are specified and their security properties are verified. Indeed, many formal methods have been developed (Meadows, 1996; Paulson, 1998; Denker et al., 2000) and succeeded in finding errors in protocols or verifying their correctness (Burrows et al., 1990; Paulson, 1998). However, it is hard to apply these methods to large protocols. This is because large protocols are complex and there are no appropriate tools for analyzing such complex protocols. With a tool designed for small protocols, specifying complex protocols and their security properties is hard. Moreover, the obtained specifications tend to be lengthy and unintuitive. To avoid these difficulties, protocols are usually simplified and the simplified protocols are verified instead.

In this paper, we discuss the Secure Electronic Transaction (SET) protocol (SET Secure Electronic Transaction LLC, 1997a; SET Secure Electronic Transaction LLC, 1997b; SET Secure Electronic Transaction LLC, 1997c). In particular, we formally specify the payment transaction protocol that is a part of SET. This formal specification serves as a starting point of a formal analysis of the protocol.

SET has been developed by Visa and MasterCard for secure electronic commerce using payment cards. Over six hundred pages are needed to explain and specify it. There are some works on the formal specification and the analysis of the protocol (Lu and Smolka, 1999; Bolignano, 1997; Kessler and Neumann, 1998). However, they simplified the protocol excessively in order to reduce the complexity. For example, most of these simplified protocols did not use dual signature, which is one of the characteristics of SET. Since we aim at verifying security properties that Meadows and Syverson discussed in (Meadows and Syverson, 1998), we include in our simplified protocol the parameters that occur in the properties. We also make the simplified protocol use dual signature. In order to describe the specification concisely and unambiguously, we first define a protocol specification language. In our language, a protocol is specified as a collection of processes that express the roles of the

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

Figure 1. A typical message flow in the Needham-Schroeder shared-key protocol

participants in the protocol. This is useful for describing the specification of the SET payment transaction protocol.

The rest of this paper is organized as follows. We first define a language for specifying large security protocols concisely and unambiguously (Section 2). We then use it to specify the SET payment transaction protocol (Section 3). We finally summarize our results and mention some related works (Section 4).

2. Protocol Specification Language

Before presenting our protocol specification language, we briefly explain our design policy for it.

Security protocols are often explained by showing a typical message flow. For example, a typical message flow of the Needham-Schroeder shared-key protocol (Needham and Schroeder, 1978) is shown in Figure 1. The first line means that a participant A sends a message composed of her name, the name of the participant she wants to authenticate, and a fresh nonce (random number) to the authentication server S. The second line means that S replies with message $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$ to A . This message is obtained by encrypting N_A, B, a newly generated key K_{AB} to be shared by A and B, and $\{K_{AB}, A\}_{K_{BS}}$ with the key K_{AS}. The $\{K_{AB}, A\}_{K_{BS}}$ is obtained by encrypting K_{AB} and A with the key K_{BS}. A, B, and N_A on the second line refer to themselves on the first line, respectively. Since A is assumed to know K_{AS} and is not assumed to know K_{BS}, she can decrypt $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$ and can not decrypt $\{K_{AB}, A\}_{K_{BS}}$.

Explanations by showing a typical message flow are concise and intuitive. However, they can not explicitly handle what each participant can see in a message because each line expresses the sending and receiving of a message at the same time. For example, on the second and the third line in the previous example, A receives a messages that includes

$$\begin{tikzcd}
\{A,B,N_A\} \arrow[d, Rightarrow] & \rightarrow \\
\{N_A,B,K_{AB},X\}_{K_{AS}} \arrow[d, Rightarrow] & \leftarrow \\
X \arrow[d, Rightarrow] & \rightarrow \\
\{N_B\}_{K_{AB}} \arrow[d, Rightarrow] & \leftarrow \\
\{N_B-1\}_{K_{AB}} & \rightarrow
\end{tikzcd}$$

Figure 2. The initiator role in the Needham-Schroedcr shareti-key protocol

$\{K_{AB},A\}_{K_{BS}}$ and sends it to B. Without assumptions on the knowledge of A, it is not clear whether if she knows the content $\{K_{AB},A\}$ of the message or not,. This ambiguity may cause human-errors in specifying complex protocols that use cryptography frequently.

To avoid this problem, we specify a protocol as a collection of processes that express the roles of the participants in the protocol. To illustrate this, we show, in Figure 2, a process that are related to A's role in the previous example. Note that we use a variable X for the encrypted component in the message from S to A. It is clear that A sends the component X to B as it is.

Now we define our protocol specification language. Since we assume the Dolev-Yao (Dolev and Yao, 1981) model, we define the set of *messages* as an algebra made from participants' names, natural numbers (including nonces), and keys with tupling and cryptographic operations. The formal syntax of messages is as follows.

$$\begin{array}{lcll}
M & ::= & A & \text{; participant's name} \\
 & | & K & \text{; key} \\
 & | & N & \text{; natural number} \\
 & | & \{M_1,\cdots,M_n\} & \text{; tuple} \\
 & | & \{M\}_K & \text{; encryption of message } M \text{ using key } K \\
 & | & \mathsf{H}(M) & \text{; hash of message } M
\end{array}$$

H is a collision-free one-way hash function. We write K^{-1} for the decryption key of a key K. For example, $\{A,N_A\}_K$ is a message obtained by encrypting a tuple of A and N_A with K, where A, N_A, and K are an participant's name, a nonce, and a key, respectively.

Since our language has variables, we define the set of *terms* by extending the previous syntax with variables.

```
T  ::=  ...
     |  X    ; variable whose name is X
```

Because we usually use variables instead of concrete names, nonces, and keys, we regard A, N_A, K, etc. that occur in terms as variables unless otherwise noted explicitly.

We finally define the set of *processes* with the following syntax. We specify a protocol as the set of processes of its participants.

```
P  ::=  End            ; silent process
     |  Send T P       ; sending of message T
     |  Recv T P       ; receiving of message T
     |  New X P        ; generating of a fresh nonce X
     |  Let X = T P    ; binding of T to the local variable X
     |  Assert Q P     ; checking of proposition Q
```

We don't specify the receiver and the sender of a message in Send *TP* and **Recv** TP, respectively because we assume that there exist intruders that can capture any message on networks and can send any message they can construct. We understand that a process of the form New *X P* binds free occurrences of X in *P*. In other words, in a process New *X P*, the variables X that occur in *P* refer to the newly generated nonce *X*. We also understand that a process of the form Recv *T P* does pattern-matching and variable-binding. For example, a process Recv {N, H (N)} *P* accepts (2001, H(2001)}, where variable *N* is bound to the number 2001. The process however does not accept (2001, H(2002)}.

Assert *Q P* acts as *P* if proposition *Q* holds, otherwise it acts as End. The set of propositions depends on the system used for analysis. Since we use Isabelle (Paulson, 1994), a proof checker of higher-order logics, we can use any proposition in Isabelle.

As an example, we specify the role of A, the initiator, in the Needham-Schroeder shared-key protocol in Figure 3. The process is parametrized by her name A, the responder's name *B*, and the key K_{AS}.

3. A Specification of the SET Payment Transaction Protocol

In this section, we give a formal specification of the SET payment transaction protocol. Since our future goal is to verify security properties that include those which Meadows and Syverson discussed in (Meadows and Syverson, 1998), we simplify the protocol into the exchanges of six

```
$initiator(A, B, K_{AS}) =$
    New $N_A$
    Send $\{A, B, N_A\}$
    Recv $\{N_A, B, K_{AB}, X\}_{K_{AS}}$
    Send $X$
    Recv $\{N_B\}_{K_{AB}}$
    Send $\{N_B - 1\}_{K_{AB}}$
    End
```

Figure 3. The process that specifies the initiator role of the Needham-Schroeder shared-key protocol

messages that include the parameters used in their security properties. Meadows and Syverson developed a method to describe security properties flexibly and discussed the security properties that the payment transaction protocol is expected to satisfy. However, they did not specify the protocol formally. Our formal specification is needed in order to verify the security properties. We also make the simplified protocol use dual signature, which is one of the characteristics of the original protocol.

We first overview the SET payment transaction protocol. Three parties, a cardholder, a merchant, and a payment gateway, are involved in a payment transaction in SET. This protocol is invoked after the cardholder has completed browsing, selection, and ordering. One of the purposes of the protocol is to securely send the payment information, which includes the account number of the payment-card of the cardholder and the amount of money that he will pay for the order, to the payment gateway.

A typical message flow of the protocol is shown in Figure 4. We show only the six messages that our simplified protocol has. We also omit the structures of the messages in the figure. The cardholder and the merchant first exchange the identifiers of the transaction in *PInitReq* and *PInitRes* messages. The identifiers are referred to in subsequent messages. The cardholder then sends the purchase request message *PReq* to the merchant. This message includes the amount of money that the cardholder will pay and her payment-card number. She keeps the number secret from the merchant by encrypting a component that includes it. The merchant sends the gateway *AuthReq* message that includes the component. The gateway checks the validity of the payment-card number, processes the payment, and returns the result to the merchant in

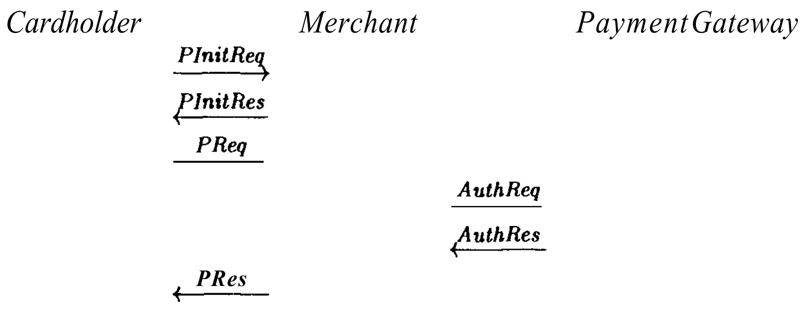

Figure 4. A typical message flow in the SET payment transaction protocol

$$
\begin{aligned}
\mathsf{L}(M_1, M_2) &= \{M_1, \mathsf{H}(M_2)\} \\
\mathsf{SO}(A_s, M) &= \{\mathsf{H}(M)\}_{\mathsf{SK}(A_s)} \\
\mathsf{S}(A_s, M) &= \{M, \mathsf{SO}(A_s, M)\} \\
\mathsf{E}(A_r, M, K) &= \{\{M\}_K, \{K^{-1}\}_{\mathsf{EK}(A_r)}\} \\
\mathsf{EX}(A_r, M_1, M_2, K) &= \{\{\mathsf{L}(M_1, M_2)\}_K, \{K^{-1}, M_2\}_{\mathsf{EK}(A_r)}\} \\
\mathsf{Enc}(A_s, B_r, M, K) &= \mathsf{E}(B_r, \mathsf{S}(A_s, M), K) \\
\mathsf{EncB}(A_s, B_r, M_1, M_2, K) &= \{\mathsf{Enc}(A_s, B_r, \mathsf{L}(M_1, M_2), K), M_2\}
\end{aligned}
$$

Figure 5. Operations on messages used in the SET payment transaction protocol

AuthRes message. The merchant receives it and sends the result to the cardholder in *PRes* message.

Various cryptographic operations are used in SET. We define each of the operations used in our protocol as a function on the set of messages in our language. The definitions are essentially the same as what Bella et al. did in their verification of the SET cardholder registration protocol (Bella et al., 2000). We show the definitions in Figure 5. The subscripts r and s of names of participants indicate that the participants appear as the receiver and the sender of a message, respectively. $\mathsf{L}(M_1, M_2)$ contains a linkage from message M_1 to message M_2. $\mathsf{SO}(A_s, M)$ is the signature of a participant A, on message M. $\mathsf{S}(A_s, M)$ is message M with the signature of A_s. Enc models a signed-then-encrypted message. EncB models a signed-then-encrypted message with an external baggage.

```
Cardholder(C, M, P, OD, PurchAmt, PAN, PANSecret) =
    New $RRPID_1$
    New $LID_C$
    New $Chall_C$
    // PInitReq
    Send $\{RRPID_1, LID_C, Chall_C\}$
    // PInitRes
    Recv $S(M, \{\{LID_C, LID_M, XID\}, RRPID_1, Chall_C, Chall_M\})$
    Let TransID = $\{LID_C, LID_M, XID\}$
    New ODSalt
    New $RRPID_2$
    Let PANData = $\{PAN, PANSecret\}$
    Let PIHead = $\{TransID, H(OD), PurchAmt\}$
    Let OIData = $\{TransID, RRPID_2, Chall_C, H(OD), ODSalt\}$
    Let PIData = $\{PIHead, PANData\}$
    New K
    // PReq
    Send $\{ \{SO(C, \{H(PIData), H(OIData)\}),$
                $EX(P, L(PIHead, OIData), PANData, K)\},$
            $\{OIData, H(PIData)\}\}$
    // PRes
    Recv $(S(M, \{TransID, RRPID_2, Chall_C\}))$
```

Figure 6. The cardholder process in the SET payment transaction protocol

EK and SK are the functions that relate each participant to his public encryption key and his public signature key, respectively.

The processes of a cardholder, a merchant, and a payment gateway are shown in Figures 6, 7 and 8, respectively.

Here, C, *M*, and *P* are the names of a cardholder, a merchant, and a payment gateway, respectively. *OD, PAN, PurchAmt* and *AuthReqAmt* are an order description, the account number of a payment-card, the amount of money that a cardholder will pay, and the amount of money that a merchant requires, respectively. *PANSecret* is used to prevent guessing attacks on *PAN*. *ValidPANSet* is the set of valid *PANS*. It does not appear in the SET specification books. We introduce it to model the authentication of payment-cards. Dual signature is used in the *PReq* message. The message is composed of the following three parts: $SO(C, \{H(PIData), H(OIData)\})$, $EX(P, L(PIHead, OIData), PANData, K)$ and $\{OIData, H(PIData)\}$.

```
$Merchant(M,C,P, OD, ODSalt, AuthReqAmt) =$
  // $PInitReq$
  Recv $\{RRPID_1, LID_C, Chall_C\}$
  New $LID_M$
  New $XID$
  New $Chall_M$
  Let $TransID = \{LID_C, LID_M, XID\}$
  // $PInitRes$
  Send $S(M, \{TransID, RRPID_1, Chall_C, Chall_M\})$
  Let $OIData = \{TransID, RRPID_2, Chall_C, H(OD), ODSalt\}$
  // $PReq$
  Recv $\{\{SO(C, HPIData, H(OIData)) PIBody\},$
        $\{OIData, HPIData\}\}$
  New $RRPID_3$
  New $K_1$
  // $AuthReq$
  Send EncB( $M, P, \{RRPID_3, TransID, AuthReqAmt\},$
                   $\{SO(C, \{HPIData, H(OIData)\}), PIBody\}, K_1)$
  // $AuthRes$
  Recv Enc($P, M, \{RRPID3, TransID, AuthAmt\}, K2)$
  Assert $AuthReqAmt = AuthAmt$
  // $PRes$
  Send $S(M, \{TransID, RRPID_2, Chall_C\})$
```

Figure 7. The merchant process in SET payment transactions

```
$Gateway(P, C, M, ValidPANSet) =$
  // $AuthReq$
  Recv EncB( $M, P, \{RRPID_3, TransID, AuthReqAmt\}$
               $\{ SO(C, \{ H(\{\{TransID, HOD, PurchAmt\}, PANData\}),$
                          $HOIData\}),$
                 $EX(P, \{ \{TransID, HOD, PurchAmt\},$
                          $HOIData\}, PANData, K_1)\})$
  Assert $PANData \in ValidPANSet$
  Assert $PurchAmt = AuthReqAmt$
  Let $AuthAmt = AuthReqAmt$
  New $K_2$
  // $AuthRes$
  Send Enc($P, M, \{RRPID_3, TransID, AuthAmt\}, K_2)$
```

Figure 8. The gateway process in SET payment transactions

The second part cannot be decrypted by a merchant and should be passed to a payment gateway. The content of the third part should be read by a merchant. The first part is the signature on *{*H*(PIData),* H*(OIData)}*. A participant who receives either of the last two parts can compute {H*(PIData),* H *(OIData)}* and can check the signature.

4. Concluding Remarks

In this paper, we have defined a language for specifying security protocols and have used it to formally specify the SET payment transaction protocol. In our language, a security protocol is specified as a collection of processes. Each process defines the role of a participant. This is useful in specifying complex protocols concisely and unambiguity.

We have simplified the SET payment transaction protocol and have specified it formally. We aim at verifying various security properties of the protocol including those that Meadows and Syversion discussed in (Meadows and Syverson, 1998). Our specification can serve as a starting point for a formal analysis that take into account dual signature of the protocol.

We have already implemented our specification language on the Isabelle theorem prover (Paulson, 1994) and have written the specification in it. We are also developing a protocol execution model and a language to describe security properties concisely. In the execution model, a state of a participant is modeled as a process in our language and an environment, a set of variable-value pairs. The environment corresponds to the data that the participant uses. For example, in a key exchange protocol, the environment of a participant may include the name of the agent that a participant will talk with and the key she will exchange. The environments can also be used to describe security properties concisely. In the previous example, the agreement between the participants about the key can be expressed as coincidence between parts of the environments of participants. We plan to describe security properties that the SET payment transaction protocol should satisfy in our language and to verify them. We further have to make clear the correspondence between the original payment transaction protocol used in actual e-commerce and the simplified version we presented in this paper.

We finally mention some related works. There are a lot of works applying formal methods to protocol analyses. We will mention a few languages used to specify protocols in these works. CSP (Hoare, 1985) is used to specify security protocols in many protocol verification systems (Schneider, 1997; Roscoe, 1995). It seems that protocol specifications in our language can be easily translated into a collection of CSP processes and that tools for CSP can be used to verify the security.

Cervesato (Cervesato, 2001a; Cervesato, 2001b) proposed a protocol specification language, called Typed MSR. It is a kind of multiset re-writing system. His language also uses role-based descriptions. Protocol specifications in our language are more concise than those in his language because, in his language, predicates that correspond to the state of each participant must be explicitly written.

There are some works on security analyses of the SET protocol. Lu and Smolka (Lu and Smolka, 1999) formally specified the protocol as CSP processes and verified five correctness properties of the protocol using the FDR (Formal Systems Ltd, 1998) model checker. They however did not analyze dual signature and did not assume the existence of intruders in their analysis.

Meadows and Syverson (Meadows and Syverson, 1998) developed a security specification language for their protocol analyzer (Meadows, 1996). They also discussed the security properties that the SET payment transaction protocol is expected to satisfy. However, they did not give the specification of the protocol formally, and they left the actual verification of the security for future work. As far as we know , no result on the verification has been published yet. Our specification can serve as a starting point of a formal verification of security properties they discussed.

Bolignano (Bolignano, 1997) proposed a method to analyze security protocols. He took a protocol that resembles SET as an example. He has not completed the analysis of SET itself as far as we know.

Bella et al. (Bella et al., 2000) analyzed the cardholder registration protocol in SET. The protocol is used to exchange certificates needed in the payment transactions. They use the inductive method (Paulson, 1998) for their analysis.

Kessler and Neumann (Kessler and Neumann, 1998) defined a logic to treat the accountability of participants in electronic commerce protocols. They used their logic to analyze the accountability of a merchant in SET. They took into account dual signature, although they treated only the *PReq* message.

Acknowledgments

The authors thank Kazuo Ohta, Akira Takura, and Kiyoshi Shirayanagi for their helpful comments and encouragement.

References

Anderson, R. and Needham, R. (1995). Programming satan's computer. In *Computer Science Today: Recent Trends and Developments,* volume 1000 of *LNCS,* pages 426-440. Springer-Verlag.

Bella, G., Massacci, F., Paulson, L. C., and 'Tkamontano, P. (2000). Formal verification of cardholder registration in SET. In *6th European Symposium on Research in Computer Security (ESORICS'00),* volume 1895 of *LNCS,* pages 159-174. Springer-Verlag.

Bolignano, D. (1997). Towards the formal verification of electronic commerce protocols. In *10th IEEE Computer Security Foundations Workshop,* pages 133-146.

Burrows, M., Abadi, M., and Needham, R. (1990). A logic of authentication. *ACM Transactions on Computer Systems,* 8(1):18-36.

Cervesato, I. (2001a). Typed MSR: Syntax and examples. In *Information Assurance in Computer Networks: Methods, Models, and Architectures for Network Security (MMM-ACNS'01),* volume 2052 of *LNCS,* pages 159-177. Springer-Verlag.

Cervesato, I. (2001b). Typed multiset rewriting specifications of security protocols. In *1s Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT'00),* ENTCS. Elsevier. To appear.

Clark, J. and Jacob, J. (1997). A survey of authentication protocol literature: Version 1.0. Technical report, Department of Computer Science, University of York.

Denker, G., Millen, J., and Rueß, H. (2000). The CAPSL integrated protocol environment. SRI Technical Report SRI-CSL-2000-02, SRI International.

Dolev, D. and Yao, A. C. (1981). On the security of public key protocols (extended abstract). In *22nd Annual Symposium on Foundations of Computer Science,* pages 350-357. IEEE.

Formal Systems Ltd (1998). FDR2 user manual.

Hoare, C. A. R. (1985). *Communicating Sequential Processes.* Prentice Hall.

Kessler, V. and Neumann, H. (1998). A sound logic for analysing electronic commerce protocols. In *5th European Symposium on Research in Computer Security (ESORICS'98),* volume 1485 of *LNCS,* pages 345-360. Springer-Verlag.

Lu, S. and Smolka, S. (1999). Model checking SET Secure Electronic Transaction Protocol. In *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'99),* pages 358-365. IEEE.

Meadows, C. (1996). The NRL protocol analyzer: an overview. *Journal of Logic Programming,* 26(2):113-131.

Meadows, C. and Syverson, P. (1998). A formal specification of requirements for payment transactions in the SET protocol. In *Financial Cryptography '98,* volume 1465 of *LNCS,* pages 122-140. Springer Verlag.

Needham, R. and Schroeder, M. (1978). Using encryption for authentication in large networks of computers. *Communications ofthe ACM,* 21(12):993-999.

Paulson, *L.* C. (1994). *Isabelle: A Generic Theorem Prover,* volume 828 of *LNCS.* Springer-Verlag.

Paulson, L. C. (1998). The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85-128.

Roscoe, A. W. (1995). Modelling and verifying key-exchange protocols using CSP and FDR. In *8th IEEE Computer Security Foundations Workshop*, pages 98-107.

Schneider, S. (1997). Verifying authentication protocols with CSP. In *10th IEEE Computer Security Foundations Workshop*, pages 3-17.

SET Secure Electronic Transaction LLC (1997a). SET secure electronic transaction book 1: Business description.

SET Secure Electronic Transaction LLC (1997b). SET secure electronic transaction book 2: Programmer's guide.

SET Secure Electronic Transaction LLC (1997c). SET secure electronic transaction book 3: Formal protocol definition.