The Verification Logic for Secure Transaction Protocols

Qingfeng Chen and Chengqi Zhang

Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia
qchen@it.uts.edu.au, chengqi@it.uts.edu.au,
Tel: +61 2 95144534 FAX: +61 2 95144535

Abstract

In common sense, electronic transactions through secure transaction protocols are secure. However, not all so-called secure transaction protocols are secure. To verify the protocols this paper proposes the NDL (non-monotonic dynamic logic). A novel idea of fail-negate, namely non-monotonic, is proposed in NDL. Thus some security properties are believed to be true if we, based on current conditions, cannot prove they are false. On the other hand, the dynamic property in NDL converts the transaction into an action sequences. To evaluate the logic, two instances are illustrated. From the evaluation, it is convinced that the NDL is effective and promising.

Keywords: Security; Secure protocol; Electronic commerce; Non-monotonic; Verification.

1. Introduction

Electronic commerce on the Internet experiences explosive growth. The facts listed below should be good evidence:

Visa, a major credit card issuer, says that the number of online business transaction completed annually is 15.3 billion in 1998 and estimates it will reach 100 billion in 2002 [3].

Security is a necessary concern in e-commerce. Nearly everyday, a news organization reports a potential security violation on the Internet. Security in electronic commerce depends heavily on the use of secure protocols. They may safeguard the communications between the principals such as the key-exchange and authentication. Therefore the secure protocols have become the key component of electronic commerce systems.

During the past decade, there have been remarkable efforts devoted to develop the method, theories, logics and formal methods, and supporting tools. These efforts can aid the designer of protocols in detecting and correcting weaknesses and redundancies in the protocol design stage. For instance, several protocols [1][5] have been revealed to have some flaws. Current researches have been divided into two classes: Attack-construction approach endeavours to construct possible attacks using algebraic properties of the algorithms in the protocols, and inference-construction approach endeavour to construct inferences using specialised logics based on a notion of knowledge and "belief".

Attack-construction approach is the construction of probable attack sets based on the algebraic properties of a protocol algorithm, involving the work of Dolev and Yao [9], the NRL Analyzer [10], and the Interrogator Model [11]. These work figure out how to ensure authentication and security properties, they do not depend on the validity of a proposed logic.

Inference-construction approach utilizes modal logic similar to those that have been developed for the analysis of the evolution of knowledge and belief in distributed systems. This includes the BAN logic in [4], the AT logic in [12] and the GNY logic in [13][14]. The BAN logic abstracts all but the information intended to be conveyed from the sender to the recipient. Thus, it is simple and has been successfully applied to discover flaws in variety of authentication protocols (e.g., [11][2]). There are some extensions of the above three logics. The BG logic (developed by Brackin) [8] is an extension of the GNY logic, and is used by software that automatically proves authentication properties of cryptographic protocols. The AUTLOG logic (an advanced logic for authentication) [16] is based on the BAN logic. This logic proposes a predicate 'recognize' and extends the role of the operator 'see'. Meanwhile, an eavesdropper is simulated in the logic.

Most of the existing methods focus on the verification of the key management protocols and authentication protocols. Two of the main limitations in e-commerce methods are as follows.
1. Because the attack-construction approach suffers from a huge state space, it has low efficiency in detecting attacks on large secure transaction protocols.

2. The inference-construction approach is still complicated when used for the verification of secure protocols. For example, BAN logic requires a large number of assumptions that a secret remains secret during the execution of protocols [15], and the GNY logic contains many rules that have to be considered at each stage.

E-commerce protocols, however, are inherently complex and a number of published protocols of this nature contain subtle errors. The methods for verification of e-commerce protocols are not quite as mature as those used for authentication protocols. Kailar has proposed a special-purpose logic, which is used for the analysis of secure electronic commerce protocols that require accountability [7]. This logic is more suited to the analysis of accountability than are the belief logics. Meadows and Syverson have designed a belief logic to analyze a single message suitable for all related protocols. Thereby, we advocate a method used to verify the protocol.

The rest of this paper is organized as follows. Section 4 gives the axioms, inference rules and the reasoning format. Section 5 gives two instances for application of NDL. We conclude in the last Section.

2. A Computational Model

The public environment is an open network that is composed of principals connected by communication links. All kinds of messages on these links, such as account number and key, constitute the communication between principals. The principal can operate the message, such as modification, on a link. An open network in fact consists of a set of principals, messages and processes. Following the rules in protocols the principals can interact with each other for accomplishing a common task (e.g., to encrypt a message). Interactions are based on messages that are conveyed via a communication facility.

The secure transaction protocol is a method used to secure transactions over an open network and carried out by the principals. The protocol is organised into several stages by message transmissions, such as a request for certificating authority, in which cryptography is used to provide confidentiality of information to ensure data integrity and to authenticate the participants. A special execution of protocol is called a ‘session’. For convenience, this paper only consider sessions that appear to end successfully. The overhead issues are excluded.

At each stage of a protocol session, every principal has a finite set of received data items, which it had before the session began, or which were extracted from the messages sent to it before, or during the current stage.

Also, if principal A authenticates the messages m sent from principal B, then A believes m. This indicates that principal A has enough confidence in message m.

After receiving a message, the principal starts verification of the message by using the inference rule and related axioms. The message and verification form the initial set of received data items and their confidences for processing next time. During the verification, if the principal cannot conclude that ‘principal B knows message m’, then the principal believes ‘B does not know m’.

To explore a flaw-free protocol, NDL is proposed in this paper. The NDL is a natural and practical analysis strategy for the verification of secure protocols. It has well-integrated techniques taken from single-rank logic, dynamic logic and non-monotonic logic, for efficiently identifying flaws. For simplicity, we assume that all principals involved in a transaction do not divulge their secret. In addition, we assume that the cryptographic algorithms and communication protocols are sound during the verification.

3. Basic Notations and Statements

Generally, uppercase X, Y, A, B, C, and CA (Certificate Authorities) range over particular principals. m₁, m₂, ...., mₙ denote specific messages. (In this paper, keys and the encrypted messages are regarded as messages either.) T denotes a specific timestamp. “Generate” and “Send” denote specific actions. (Encryption and digital signature are some mapping operation on message, but not action.) “Knowledge state” denotes a specific knowledge relation between principal...
and message. "Authentication" denotes a specific belief relation. 

**Fresh** is applied to a process that asserts this term was created for a current session and not recorded from an earlier session. A principal believes a term is fresh if it can identify the term as created for the current session whenever this term comes into its possession. To consider the possibility that communication keys may be compromised and the timestamp may prevent replays, we now introduce the timestamp $T$ for freshness of message. If the clock synchronization of both parties is difficult, a trusted third party can intervene as a notary and use its own clock as a reference [22]. Actually, the timestamp has the additional benefit of replacing the two-step handshake in the Needham and Schroeder protocol for a public key system, but the exposure of a user’s private keys cannot be eliminated by the timestamp. Unless compromises are reported to the AS (authentication server) and public keys are obtained from the AS immediately prior to use, timestamps are useful for validating the time integrity of keys [6].

**Authentication** denotes specific belief relations. The purpose of authentication of participants is to reduce, if not eliminate, the risk that intruders might masquerade under legitimate appearances to pursue unauthorized operations.

**Function word:**

$\text{Encrypt}(X, k, m)$: This represents the operation that message $m$ is encrypted under the symmetric key $k$.

$\text{Sign}(X, m)$: This represents the operation that message $m$ is encrypted under the public key $k$, namely $\text{Kpb}(X)$, $\text{Kpv}(X)$, $\text{Spb}(X)$, and $\text{Spv}(X)$ listed below.

$\text{Hash}(m)$: This represents the message digest of message $m$ encoded by the one-way hashing algorithm. The one-way hash function has the property that given the output it is difficult to determine the input.

$\text{Kpb}(X)$: The public key-exchange key of $X$.

$\text{Kpv}(X)$: The private key-exchange key of $X$.

$\text{Spb}(X)$: The public signature key of $X$.

$\text{Spv}(X)$: The private signature key of $X$.

$m_1, m_2, \ldots, m_n$: The combination of messages $m_1, m_2, \ldots, m_n$.

**Predicate:**

$\text{Know}(X, m)$: $X$ knows message $m$. It is possible that $X$ generates the message $m$ by itself or receives $m$ from $Y$.

$\text{Auth}(X, Y, m)$: $X$ authenticates message $m$ sent by $Y$ and $m$ has not been modified.

**Action:**

$\text{Generate}(X, m)$: $X$ generates the message $m$.

$\text{Send}(X, Y, m)$: $X$ sends the message $m$ to $Y$.

If $\alpha$ and $\beta$ are basic sequences of action, then $\alpha \circ \beta$, which is the conjunction of $\alpha$ and $\beta$, can be treated as an action sequences either.

**Assertion:**

$$P \vdash \alpha \in Q$$

$P$ and $Q$ present a set of formulae; and $\alpha$ denotes an action sequences. This assertion means if $P$ is true then $\alpha$ can be executed, and $Q$ will be true if $\alpha$ can be performed successfully.

**Abbreviation:** We define four expressions, abbreviated to simplify the conjunctions of several function words, since these conjunctions repeatedly appear in our logical system.

$\text{Sign}(X, m) = \langle m, S(<ID_X, H(m), \text{Spv}(X)) \rangle$: This means that plaintext $m$ and $X$'s identifier $ID_X$ are attached to $X$'s digital signature.

$\text{Sign}(X, m)_T = \langle m, S(<ID_X, T, H(m), \text{Spv}(X)) \rangle$: Inserting the timestamp $T$ into $\text{Sign}(X, m)$.

$\text{So}(X, m) = S(<ID_X, H(m), \text{Spv}(X))$: This means that identifier $ID_X$ was attached to $X$’s digital signature before $X$ encrypted the message digest of $m$ in the private signature key $\text{Spv}(X)$.

$\text{So}(X, m)_T = S(<ID_X, T, H(m), \text{Spv}(X))$: Inserting the timestamp $T$ into $\text{So}(X, m)$.

$\text{Cert}(X) = \text{Sign}(\text{CA}, <X, \text{Kpb}(X))$: This means that the key-exchange certificate of $X$.

$\text{Cert}(X) = \text{Sign}(\text{CA}, <X, \text{Spb}(X))$: This means that the signature certificate of $X$.

4. Inference Framework

In this section, we present the axioms, inference rules, and inference format comprising the accumulation property.

4.1 Axiom

(1) **Encryption**

1-1 $\text{Know}(X, m) \land \text{Know}(X, k) \rightarrow \text{Know}(X, \text{Encrypt}(m, k))$

This means that, if $X$ knows message $m$ and symmetric key $k$, then $X$ knows $\text{Encrypt}(m, k)$ by using $k$ to encrypt message $m$.

1-2 $\text{Know}(X, m) \land \text{Know}(X, \text{Kpb}(Y)) \rightarrow \text{Know}(X, \text{Encrypt}(m, \text{Kpb}(Y)))$

This means that, if $X$ knows message $m$ and public key-exchange key $\text{Kpb}(Y)$ of $Y$, then $X$ knows $\text{Encrypt}(m, \text{Kpb}(Y))$ by using $\text{Kpb}(Y)$ to encrypt message $m$ attached timestamp $T$.

(2) **Key Allocation**

2-1 $\text{Know}(X, \text{Kpb}(\text{CA}))$

2-2 $\text{Know}(X, \text{Spb}(\text{CA}))$

2-3 $\text{Know}(X, \text{Kpv}(Y))$

2-4 $\text{Know}(X, \text{Spv}(Y))$

That is, $X$ knows public key-exchange key $\text{Kpb}(\text{CA})$ of CA (Certificate Authorities). $X$ knows public signature key $\text{Spb}(\text{CA})$ of CA. $X$ knows exchange and private signature key itself.
(3) Decryption

3-1 Know(X, k) \land Know(X, E(m, k)) \rightarrow Know(X, m)

That is, if X knows symmetric key k and E(m, k), then X knows the plain text of message m by using k to decrypt E(m, k).

3-2 Know(X, Kpv(Y)) \land Know(X, S(m, Kpb(Y))) \rightarrow Know(X, m)

That is, if X knows private key-exchange key Kpv(Y) of Y and刺激 S(m, Kpb(Y)), then X knows the plain text of message m by using the Kpv(Y) to decrypt S(m, Kpb(Y)).

(4) Signature

4-1 Know(X, m) \rightarrow Know(X, H(m))

That is, if X knows message m, then X knows H(m).

4-2 Know(X, m) \land Know(X, Spv(Y)) \rightarrow Know(X, S(r, H(m), Spv(Y))]

That is, if X knows message m and private signature key Spv(Y) of Y, then X knows S(r, H(m), Spv(Y)) by using Spv(Y) to encrypt H(m).

(5) Authentication

5-1 Know(X, m) \land Know(X, S(<ID_Y, T, H(m)>) \land Spv(Y))]

That is, if X knows message m and authenticates Y that has not been modified, then X can authenticate Y by checking that message m and comparing the result with a newly generated message digest of message m.

5-2 Know(X, m) \land Auth(X, Y, H(m)) \rightarrow Auth(X, Y, m)

That is, if X knows message m, and authenticates Y that has not been modified, then X can authenticate Y sent m, and m has not been modified by using Spv(Y) to decrypt S(<ID_Y, T, H(m)>) \land Spv(Y)) and comparing the result with a newly generated message digest of message m. Simultaneously, X verifies that a message is not replays by checking that the operation of digital signature in secure protocol.

5-3 Know(X, Spv(Y)) \land Know(X, S(<ID_Y, T, Spv(Y)))

That is, if X knows Spv(Y), and S(<ID_Y, T, Spv(Y))) >, then X can authenticate Spv(Y) by using the public signature key of CA to decrypt the encrypted message and checking the timestamp included in the message.

(6) Separation

6-1 Know(X, <m_1, ..., m_n>) \land ... \land Know(X, m_n)

That is, if X knows the conjunction of messages m_1, m_2, ..., m_n, then X knows every element of this conjunction, or vice versa.

6-2 Auth(X, Y, <m_1, ..., m_n>) \rightarrow Auth(X, Y, m_1) \land ... \land Auth(X, Y, m_n)

That is, if X authenticates that Y ever sent a compound of messages m_1, m_2, ..., and m_n, that has not been modified, then X authenticates that Y ever sent every element of it and none of it have not been modified.

Based on the notations, statements and axioms described above, two theorems listed below can be naturally proved.

Theorem 1: Know(X, m) \land Know(X, S0(Y, m_r) \land Know(X, Spv(Y))

[Proof]:

(1) Know(X, m)

(2) Know(X, S0(Y, m_r)

(3) Know(X, Spv(Y))

(4) Know(X, S(<ID_Y, T, H(m)>) \land Spv(Y))]

(5) Auth(X, Y, m)

(6) Know(X, m) \land Know(X, S0(Y, m_r) \land Know(X, Spv(Y))

\rightarrow Auth(X, Y, m)

The proof can easily be constructed by using the authentication axiom 5-1 and the definition of S0(Y, m_r), which is an equivalent expression of S(<ID_Y, T, H(m)>) \land Spv(Y))]

Theorem 2: Know(X, Sign(Y, m_r) \land Know(X, Spv(Y))

[Proof]:

(1) Know(X, Sign(Y, m_r)

(2) Know(X, Spv(Y))

(3) Know(X, S(<ID_Y, T, H(m)>) \land Spv(Y))]

(4) Know(X, m)

(5) Know(X, S(<ID_Y, T, H(m)>) \land Spv(Y))]

(6) Know(X, So(Y, m))

(7) Auth(X, Y, m)

(8) Know(X, Sign(Y, m_r) \land Know(X, Spv(Y))

\rightarrow Auth(X, Y, m)

Sign(Y, m_r) is an equivalent expression of S(<ID_Y, T, H(m)>) \land Spv(Y)). The above simple theorems are common processes in secure protocol and provide a compendious and perspicuous way describing the operation of digital signature in secure protocol.

4.2 Inference Rules

(R-1) Revelation

Know(X, m) \models \exists_{m} Know(Z, m)

states that the secret cannot be protected when plain text is transferred in a practical distributed system and an open network. The rule, in fact, provides a convenient way to simulate the eavesdropper during the communication.

(R-2) Generation

\models \exists_{m} Know(X, m)
2002 International Conference on Intelligent Information Technology Proceedings

says that if message \( m \) is generated by \( X \) then \( X \) itself must know \( m \).

(R-3) Accumulation

\[
P \vdash Q \\
P \vdash \alpha Q
\]
says that if the conclusion \( Q \) has been proved to be true, then it remains true after action \( \alpha \). Actually, this rule cannot accommodate to any secure protocols for two reasons: (1) if the keys are altered during the execution of protocol, then the knowledge about the keys is outdated; (2) if the protocols do not provide a memory function, the knowledge about message \( m \) can be forgotten. Therefore, the logic system mentioned in this paper can only be applied to protocols satisfying the following two conditions: (1) the keys cannot be replaced during the execution of the protocol; (2) every principal must store the knowledge of the message. SET and X.509 Certificate Policy satisfy these two conditions in terms of our current verification. To the exceptional circumstances, we will illustrate them in another paper.

(R-4) Union

\[
P \vdash \alpha Q, O \vdash \beta R \\
P \vdash \alpha \beta R
\]
says that if the conclusion \( Q \) of the former action \( \alpha \) is the premise of the later action \( \beta \), then \( \alpha \) and \( \beta \) can be synthesized by acting the original conclusion \( R \) of \( \beta \) as the ultimate conclusion. In fact, the synthesis of actions can be regarded as a kind of algebraic operation with the union property.

(R-5) Non-monotonic

\[
\forall P \vdash \alpha \text{Auth}(X, Y, m) \\
P \vdash \alpha \sim \text{Auth}(X, Y, m)
\]
says that if we cannot conclude that a principal \( X \) successfully authenticates \( m \) after a sequence of action \( \alpha \), then we non-monotonously assume that \( X \) does not believe the validity of \( m \). Here, Non-monotonic means based on the only knowledge, premise \( P \) and action \( \alpha \), if we cannot conclude \( X \) know \( m \), then it is reasonable to suppose \( X \) does not know \( m \); but it is quite possible to conclude \( X \) know \( m \) as \( P \) or \( \alpha \) is extended, so the non-monotonic assumption will be correspondingly modified in terms of the new instance. This, in fact, reflects a typical phenomenon in the verification of transaction protocols: it is very difficult to affirmatively and unconditionally judge what principal \( X \) does not know since the message can be intercepted. However, we can accept that \( X \) should not know \( m \) if the paths are limited for \( X \) to understand \( m \).

4.3 Inference Format

This inference format stems from the accumulation property described above where \( e, \alpha_1, \alpha_2, \ldots, \alpha_n \), and \( \epsilon \) express an action sequences and formula set respectively.

\( \epsilon \) (empty sequence of action)

\[
f_{01}, f_{02}, \ldots, f_{0m_0}
\]

\( \alpha_1 \)

\[
f_{11}, f_{12}, \ldots, f_{1m_1}
\]

\( \alpha_2 \)

\[
f_{21}, f_{22}, \ldots, f_{2m_2}
\]

\( \vdots \)

\( \alpha_n \)

\[
f_{n1}, f_{n2}, \ldots, f_{nm_n}
\]

The above expression can be compressed by using an assertion to describe the procedure of deduction. The union of formula is derived from the definition of the assertion, for instance, \( \{ f_{01}, f_{02}, \ldots, f_{0m_0} \} \) is a formula, so each \( f_{00} \) is a subset of this formula. Furthermore, the synthesis of assertion is, in fact, derived from inference rule R-3 and R-4. This format can be described by another expression as shown below:

\[
\{ f_{01}, f_{02}, \ldots, f_{0m_0} \} \vdash \alpha_1 \{ f_{11}, f_{12}, \ldots, f_{1m_1} \}
\]

\[
\{ f_{01}, f_{02}, \ldots, f_{0m_0} \} \vdash \alpha_1, \alpha_2 \{ f_{21}, f_{22}, \ldots, f_{2m_2} \}
\]

\( \vdots \)

\[
\{ f_{01}, f_{02}, \ldots, f_{0m_0} \} \vdash \alpha_1, \alpha_2, \ldots, \alpha_n \{ f_{m1}, f_{m2}, \ldots, f_{nm_n} \}
\]

This inference format is based on the accumulation and dynamic properties of transaction protocols. It is well known that a message can be modified or intercepted during transmission, so an eavesdropper can impersonate the sender or receiver and continue the transaction. In this paper, we assume these problems, which have been detected until now, are protected by existing technologies such as [6][20][21]. Therefore we do not need to be too concerned about network communication, eavesdropping, hashing, and encryption algorithms. Thus, this paper concentrates on how the NDL may be used on the verification of secure transaction protocols, rather than how to construct a secure protocol.
5. Verification instances of secure protocols in NDL

To evaluate the framework NDL, two instances are used to demonstrate the use of the NDL when verifying a secure protocol.

Example 1: Distribution of Public keys in Needham and Schroeder's protocol.

Timestamp can be added to Needham and Schroeder's protocols for public key systems. This has been proposed by Denning [6]. In this instance, we show that our logic detects a known flaw in the protocol.

The protocol opens with A consulting the authentication server AS in the clear to find B's public key. The exchange can be described in a series of "actions sequence".

\[
\begin{align*}
1 & \rightarrow A: \alpha_1 = \text{Send}(A, AS, \langle A, B \rangle) \\
\text{Send}(AS, A, S(<B, \text{Sph}(B), \text{Sph}(AS)>) & \downarrow A \rightarrow B: \alpha_2 = \text{Send}(A, B, S(\langle 1D_A, A \rangle, \text{Sph}(B))) \\
\text{Sph}(B) & \downarrow B \rightarrow AS: \alpha_3 = \text{Send}(B, AS, S(<1D_B, B \rangle, \text{Sph}(B))) \\
\text{Sph}(AS) & \downarrow A \rightarrow B: \alpha_4 = \text{Send}(A, B, S(\langle 1D_A, 1D_B \rangle, \text{Sph}(B))) \\
\text{Sph}(B) & \downarrow B \rightarrow A: \alpha_5 = \text{Send}(B, A, S(\langle 1D_B, 1D_A \rangle, \text{Sph}(B))) \\
\end{align*}
\]

where \text{Sph}(AS) denotes AS's private signature key and \text{Sph}(B) is B's public signature key. A is presumed to know AS's public signature key (derived from 2-2) since AS is an authority.

We start the verification from the goal \text{Auth}(A, AS, \text{Sph}(B)) and let \text{Cert}(B) = \langle B, \text{Sph}(B) \rangle. Formula P denotes the set of premises; \alpha is the combination of a series of actions, and Formula \theta denotes the object we want to verify.

\[
P' = \{\text{Know}(A, \text{Sph}(AS))\},
\]

\[
\alpha = \text{Generate}(l, A, B),
\]

\[
\theta = \{\text{Auth}(A, AS, \text{Sph}(B))\}.
\]

**Proof:**

\[
\begin{align*}
(1) & \rightarrow A: \text{Auth}(A, AS) \\
(2) & \rightarrow \text{Generate}(A, \langle A, B \rangle) \\
(3) & \rightarrow \text{Know}(A, \langle A, B \rangle) \\
(4) & \rightarrow \text{Send}(A, AS, \langle A, B \rangle) \\
(5) & \rightarrow \text{Know}(AS, \langle A, B \rangle) \\
(6) & \rightarrow \text{Know}(AS, \langle A, B \rangle) \\
(7) & \rightarrow \text{Generate}(AS, S(\langle CK, \text{Sph}(CA) \rangle)) \\
(8) & \rightarrow \text{Know}(AS, S(\langle CK, \text{Sph}(CA) \rangle)) \\
(9) & \rightarrow \text{Send}(AS, A, S(\langle CK, \text{Sph}(CA) \rangle)) \\
(10) & \rightarrow \text{Know}(A, S(<B, \text{Sph}(B)), \text{Sph}(AS))) \\
(11) & \rightarrow \text{Auth}(A, AS, \text{Sph}(B)) \downarrow (1)(3)(10)[5-3][R-5]
\end{align*}
\]

The final conclusion is that A fails to authenticate the validity of \text{Sph}(B) sent by AS since the existing conditions do not satisfy the requirement of axiom (5-3). We may conclude non-monotonically that A does not believe the validity of \text{Sph}(B) encrypted by AS due to the non-monotonic rule (R-5). Thus, the message is rejected and an appropriate response message is returned to the \text{AS} issued the \text{Sph}(B). AS may need to revoke B's public signature key and identify it.

The reason for this is that the intruder was able to intercept the message from (4) and (9) and replay the encrypted component from message (10). The weakness that allowed the attack lies in the fact that this protocol did not add a timestamp to step (9). This would have made the recipient suspect that the message was a replay.

Example 2: One section of Cardholder Registration in SET protocol.

Based on the SET protocol developed by Visa and MasterCard in May of 1997, on the advice of GTE, IBM, Microsoft, Netscape, RSA [17][18][19], we describe a simple example of registration, which is started when the cardholder \text{C} request an copy of \text{CA}'s key-exchange certificate. There are two principals in this instance: cardholder \text{C} and certificate authority \text{CA}.

\[
\begin{align*}
C & \rightarrow CA: \alpha_1 = \text{Send}(C, CA, \text{InitReq}) \\
\text{CertS}(C) & \rightarrow C: \alpha_2 = \text{Send}(C, CA, \text{CertS}(CA), \text{CertK}(CA), S(\text{CertRes}(\text{Sph}(CA)))) \\
\text{CertS}(C) & \rightarrow \text{Sph}(CA) \\
\end{align*}
\]

Here, \text{InitReq} denotes initial request sent by \text{C} to \text{CA}. When \text{CA} receives the request, it generates response \text{InitRes} and digitally signs it. It then sends the response along with the \text{CA} certificate to \text{C}. After verifying the \text{CA} certificate, \text{C} generates registration form \text{RegForReq} and encrypts the message with a randomly generated symmetric key \text{k}. This key, along with the \text{C}'s account number \text{AcctNum} is then encrypted with \text{Sph}(CA). \text{C} transmits these messages to \text{CA}.

We begin the verification from the goal \text{Auth}(C, CA, \text{InitRes}). The definitions of \text{P}, \alpha and \theta have the same meaning as above.

\[
P = \{\text{Know}(C, \text{Sph}(CA))\},
\]

\[
\alpha = \text{Generate}(\text{C}, \text{InitRes}) \downarrow \alpha_1 \downarrow \alpha_2 = \text{Generate}(\text{C}, \text{CertS}(\text{CA}), \text{CertK}(\text{CA}), \text{CertRes}(\text{Sph}(\text{CA}))) \downarrow \alpha_3 \downarrow \text{Generate}(\text{C}, \langle S(\text{RegForReq}, \text{k}), \text{CertS}(\text{CA}), \text{CertK}(\text{CA}), \text{CertRes}(\text{Sph}(\text{CA}))) \rangle \downarrow \alpha_4
\]

\[
\theta = \{\text{Auth}(C, CA, \text{InitRes})\}.
\]

**Proof:**

\[
\begin{align*}
(1) & \rightarrow \text{Know}(C, \text{Sph}(CA)) \\
(2) & \rightarrow \text{Generate}(C, \text{InitRes}) \\
(3) & \rightarrow \text{Know}(C, \text{InitRes}) \\
(4) & \rightarrow \text{Send}(C, CA, \text{InitRes}) \\
(5) & \rightarrow \text{Know}(CA, \text{InitRes}) \\
(6) & \rightarrow \text{Generate}(CA, \langle \text{CertS}(CA), \text{CertK}(CA), \text{CertRes}(\text{Sph}(\text{CA}))) \rangle \downarrow \text{InitRes}(\text{Sph}(\text{CA}))) \downarrow \alpha_5 \\
(7) & \rightarrow \text{Know}(CA, \langle \text{CertS}(CA), \text{CertK}(CA), \text{CertRes}(\text{Sph}(\text{CA}))) \rangle \downarrow \text{InitRes}(\text{Sph}(\text{CA}))) \downarrow \alpha_6 \\
(8) & \rightarrow \text{Send}(CA, <\text{CertS}(\text{CA}), \text{CertK}(\text{CA}), \text{CertRes}(\text{Sph}(\text{CA}))) \rangle \downarrow \alpha_7 \\
(9) & \rightarrow \text{Know}(CA, \langle \text{CertS}(\text{CA}), \text{CertK}(\text{CA}), \text{CertRes}(\text{Sph}(\text{CA}))) \rangle \downarrow \alpha_8
\end{align*}
\]
If a protocol with flaws is placed in a practical drawback when the protocol is in the design stage, the protocol designer must be very careful to detect every possible subtle flaw. In fact, the protocol designer believes certificate is secure. In fact, the protocol designer cannot substitute for the identifier. The intruder can replay this message in a later transaction. This problem is common in the literature [23]. The identifier is a random number and is used only once, but it is not easy for all intruder to turn the content of certificate. Although some secure protocols have been developed. However, they do not perfectly solve the problem. The consequences are not serious since it is not easy for an intruder to turn the content of the certificate. However, this certainly represents an attack since it leads to an intruder holding incorrect belief. C believes that it is C who thought it was talking to C.

6. Conclusions

The explosion of publicity of electronic commerce has enormously impacted the financial services industry. This also causes some security risks associated with sending unprotected financial information across public networks such as confidentiality and integrity. Therefore, some secure protocols have been developed. However, many secure protocols still suffer from some subtle defects. In spite of a lot of researches on this subject, a variety of formal methods have been developed for analyzing secure protocols, for instance, BAN logic, GNY logic, BIGNY logic, and AUTLOG logic. But the result is not as good as people expect.

This paper proposes a new logic NDL for the verification of secure protocols. It is based on the dynamic and non-monotonic properties. Meanwhile, NDL presents more flexibility than the existing logics used in the analysis of secure transaction protocols.

In particular, the verification can automatically halt in answer to any unsuccessful authentication under our framework. Therefore, we can reduce the authentication of security problems. In addition, by introducing the notation of identifier and timestamp, which help the designer of protocol to detect attacks, we can protect against replays. The instances presented in Section 5 demonstrate that NDL is useful for finding some known defects. In addition, it detects some subtle flaws in the SET protocol. Thus, NDL is effective and hopeful.

Reference:


