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 weakness 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 analys of the evolution of knowledge and belief in distribute systems. This includes the BAN logic in [4], the AT log in [12] and the GNY logic in [13][14]. The BAN log abstracts all but the information intended to be convey from the sender to the recipient. Thus, it is simple, a has been successfully applied to discover flaws ir variety of authentication protocols (e.g., [1][2]). There some extensions of the above three logics. The BG logic (developed by Brackin) [8] is an extension of GNY logic, and is used by software that automatic proves authentication properties of cryptogra protocols. The AUTLOG logic (an advanced logi authentication) [16] is based on the BAN logic. This l proposes a predicate 'recognize' and extends the mea of the operator 'see'. Meanwhile, an eavesdropper is simulated in the logic.

Most of the existing methods focus on the verific of the key management protocols and authenti protocols. Two of the main limitations in ex 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 [17] have designed a language for describing SET specifications, but have not actually verified the protocol. Also, Kessler and Neumann [24] have designed a belief logic to analyze a single message of the payment phase of SET.

So, it is very difficult to develop a method that can be suitable for all related protocols. Thereby, we advocate a logical framework, NDL, for secure transaction protocols. The NDL is referred as an inference-construction approach. Compared to traditional techniques, the NDL has some features, such as *fail-negate*, *dynamic*, and *non-monotonic*. These properties enable us to resolve two key problems:

- confidentiality: the intruder may detect and intercept some commercial information as message is travelled across the network,
- integrity: attacker may alter the message content during the transmission between the originator and the recipient.

To evaluate the logic, two instances are illustrated. From our evaluation, it is convinced that the NDL is effective and promising.

The rest of this paper is organized as follows. Section 2 presents a computational model. In Section 3, a formal description of basic notations and statements of NDL are provided. Section 4 gives the axioms, inference rules and the reasoning format. Section 5 gives two instances for the 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) ranger over particular principals. m_1, m_2, ..., and m_n 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 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:

$E(m,k)$: This represents the operation that message m is encrypted under the symmetric key k.

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

$H(m)$: 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.

$Kpb(X)$: The public key-exchange key of X.

$Kpv(X)$: The private key-exchange key of X.

$Spb(X)$: The public signature key of X.

$Spv(X)$: The private signature key of X.

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

Predicate:

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

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

Action:

$Generate(X, m)$: X generates the message m.

$Send(X, Y, m)$: X sends the message m to Y.

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

Assertion:

$$P \vdash_{\alpha} Q$$

P and Q present a set of formulae; and α denotes an action sequences. This assertion means if P is true then α can be executed, and Q will be true if α 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 logic system.

$Sign(X, m)=<m, S(<ID_X, H(m)>, Spv(X))>$: This means that plaintext m and X's identifier IDx are attached to X's digital signature.

$Sign(X, m)_T = <m, S(<ID_X, T, H(m)>, Spv(X))>$: Inserting the timestamp T into $Sign(X, m)$.

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

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

$CertK(X) = Sign(CA, <X, Kpb(X)>)$: This means that the key-exchange certificate of X.

$CertS(X) = Sign(CA, <X, Spb(X)>)$: 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 $Know(X, m) \wedge Know(X, k) \rightarrow Know(X, E(m, k))$

This means that, if X knows message m and symmetric key k, then X knows $E(m, k)$ by using k to encrypt message m.

1-2 $Know(X, m) \wedge Know(X, Kpb(Y)) \rightarrow Know(X, S(<T, m>, Kpb(Y)))$

This means that, if X knows message m and public key-exchange key $Kpb(Y)$ of Y, then X knows $S(<T, m>, Kpb(Y))$ by using $Kpb(Y)$ to encrypt message m attached timestamp T.

(2) **Key Allocation**

2-1 $Know(X, Kpb(CA))$

2-2 $Know(X, Spb(CA))$

2-3 $Know(X, Kpv(X))$

2-4 $Know(X, Spv(X))$

That is, X knows public key-exchange key $Kpb(CA)$ of CA (Certificate Authorities); X knows public signature key $Spb(CA)$ of CA; X knows exchange and private signature key itself.

(3) **Decryption**

3-1 Know(X, k) ∧ Know(X, E(m, k)) → 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)) ∧ Know(X, S(m, Kpb(Y))) → Know(X, m)

That is, if X knows private key-exchange key $Kpv(Y)$ of Y and $S(m, Kpb(Y))$, 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) → Know(X, H(m))

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

4-2 Know(X, m) ∧ Know(X, Spv(Y)) → Know(X, S(H(m), Spv(Y)))

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

(5)**Authentication**

$$5\text{-}1\ Know(X, m) \wedge Know(X, S(<ID_Y, T, H(m)>, Spv(Y))) \wedge Know(X, Spb(Y)) \xrightarrow{|Clock - T| < \Delta t_1 + \Delta t_2} Auth(X, Y, m)$$

This means that, if X knows message m, $S(<ID_Y, T, H(m)>, Spv(Y))$, and $Spb(Y)$, then X can authenticate Y sent m, and m has not been modified by using $Spb(Y)$ to decrypt $S(<ID_Y, T, H(m)>, Spv(Y))$ and comparing the result with a newly generated message digest of m. Simultaneously, X verifies that a message is not replays by checking that $|Clock - T| < \Delta t_1 + \Delta t_2$, where Clock is the local time. Δt_1 is an interval representing the normal discrepancy between the server's clock and the local clock, and Δt_2 is an interval representing the expected network delay time.

5-2 Know(X, m) ∧ Auth(X, Y, H(m)) → Auth(X, Y, m)

That is, if X knows message m, and authenticates Y sent $H(m)$ that has not been modified, then X can authenticate Y sent m, and m has not been modified.

$$5\text{-}3\ Know(X, Spb(Y)) \wedge Know(X, S(<Y, T, Spb(Y)>, Spv(CA))) \wedge Know(X, Spb(CA)) \xrightarrow{|Clock - T| < \Delta t_1 + \Delta t_2} Auth(X, Y, Spb(Y))$$

That is, if X knows $Spb(Y)$, and $S(<Y, T, Spb(Y)>, Spv(CA))$, then X can authenticate $Spb(Y)$ 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, <m1, ..., mn>) ↔ Know(X, m1) ∧ ... ∧ Know(X, mn)

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

6-2 Auth(X, Y, <m1, ..., mn>) → Auth(X, Y, m1) ∧ ... ∧ Auth(X, Y, mn)

That is, if X authenticates that Y ever sent a compound of messages $m_1, m_2, \ldots\ldots$, *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.

Theorem1: $Know(X, m) \wedge Know(X, S0(Y, m)_T) \wedge Know(X, Spb(Y)) \xrightarrow{|Clock - T| < \Delta t_1 + \Delta t_2} Auth(X, Y, m)$

[Proof]:

(1) *Know(X,m)*	[premise]
(2) *Know(X, So(Y, m)T)*	[premise]
(3)*Know(X, Spb(Y))*	[premise]
(4)*Know(X, S(<IDY, T, H(m)>, Spv(Y)))*	[definition]
(5)*Auth(X, Y, m)*	(1)(3)(4)[5-1]
(6)*Know(X, m)∧Know(X, So(Y, m)T)∧Know(X, Spb(Y)) →Auth(X, Y, m)*	(1)(2)(3)(5)[→ +]

The proof can easily be constructed by using the authentication axiom 5-1 and the definition of $So(Y, m)_T$, which is an equivalent expression of $S(<, T, H(m)>, Spv(Y))$.

Theorem 2: $Know(X, Sign(Y, m)_T) \wedge Know(X, Spb(Y)) \xrightarrow{|Clock - T| < \Delta t_1 + \Delta t_2} Auth(X, Y, m)$.

[Proof]:

(1)*Know(X, Sign(Y, m)T)*	[premise]
(2)*Know(X, Spb(Y))*	[premise]
(3)*Know(X, <m, S(<IDY, T, H(m)>, Spv(Y))*	(1)[definiton]
(4)*Know(X, m)*	(3)[6-1]
(5)*Know(X, S(<IDY, T, H(m)>, Spv(Y)))*	(3)[6-1]
(6)*Know(X, So(Y, m))*	(5)[definition]
(7)*Auth(X, Y, m)*	(2)(4)(6)[Theorem 1]
(8) *Know(X, Sign(Y, m)T) ∧ Know(X, Spb(Y)) → Auth(X, Y, m).*	(1)(2)(7)[→ +]

Sign(Y, m)T is an equivalent expression of *<m, S(<IDY, T, H(m)>, Spv(Y))*. The above two 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

$$Know(X, m) \vdash_{Send(X, Y, m)} Know(Z, m)$$

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

$$\vdash_{Generate(X, m)} Know(X, m)$$

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

(R-3) Accumulation

$$\frac{P \vdash Q}{P \vdash_{\alpha} Q}$$

says that if the conclusion Q has been proved to be true, then it remains true after action α. 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

$$\frac{P \vdash_{\alpha} Q,\ Q \vdash_{\beta} R}{P \vdash_{\alpha \circ \beta} R}$$

says that if the conclusion Q of the former action α is the premise of the later action β, then α and β can be synthesized by acting the original conclusion R of β 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

$$\frac{\not\Rightarrow P \vdash_{\alpha} Auth(X, Y, m)}{P \mid\sim_{\alpha} \neg Auth(X, Y, m)}$$

says that if we cannot conclude that a principal X successfully authenticates m after a sequence of action α, 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 α, 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 α 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 too limited for X to understand m.

4.3 Inference Format

This inference format stems from the accumulation property described above where ε, α_1, α_2, ... α_n and f_n express an action sequences and formula set respectively.

ε (empty sequence of action)

$$f_{01}, f_{02} \ldots\ldots f_0 m_0$$

α_1

$$f_{11}, f_{12} \ldots\ldots f_1 m_1$$

α_2

$$f_{21}, f_{22} \ldots\ldots f_2 m_2$$

$\vdots$

α_n

$$f_{n1}, f_{n2} \ldots\ldots f_n m_n$$

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, $\{f_{01}, f_{02} \ldots\ldots f_0 m_0\}$ is a formula, so each f_{0i} 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:

$$\{f_{01}, f_{02} \ldots\ldots f_0 m_0\} \vdash \alpha_1 \{f_{11}, f_{12} \ldots\ldots f_1 m_1\}$$

$$\{f_{01}, f_{02} \ldots\ldots f_0 m_0\} \vdash \alpha_1 . \alpha_2 \{f_{21}, f_{22} \ldots\ldots f_2 m_2\}$$

$$\vdots$$

$$\{f_{01}, f_{02} \ldots\ldots f_0 m_0\} \vdash \alpha_1 . \alpha_2 . \ldots\ldots . \alpha_n \{f_{n1}, f_{n2} \ldots\ldots f_n m_n\}$$

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".

$A \rightarrow AS$: $\alpha_1 = Send(A, AS, <A, B>)$ (1.1)

$AS \rightarrow A$: $\alpha_2 = Send(AS, A, S(<B, Spb(B)>, Spv(AS)))$ (1.2)

$A \rightarrow B$: $\alpha_3 = Send(A, B, S(<ID_A, A>, Spb(B)))$ (1.3)

$B \rightarrow AS$: $\alpha_4 = Send(B, AS, <B, A>)$ (1.4)

$AS \rightarrow B$: $\alpha_5 = Send(AS, B, S(<Spb(A), A>, Spv(AS)))$ (1.5)

$B \rightarrow A$: $\alpha_6 = Send(B, A, S(<ID_A, ID_B>, Spb(A)))$ (1.6)

$A \rightarrow B$: $\alpha_7 = Send(A, B, S(ID_B, Spb(B)))$ (1.7)

where $Spv(AS)$ denotes AS's private signature key and $Spb(B)$ 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 $Auth(A, AS, Spb(B))$ and let $CK=<B, Spb(B)>$. Formula P denotes the set of premise; α is the combination of a series of actions; and Formula Q denotes the object we want to verify.

$P = \{Know(A, Spb(AS))\}$.

$\alpha = Generate(A, <A, B>) \circ \alpha_1 \circ Generate(AS, S(CK, Spv(CA))) \circ \alpha_2 \circ Generate(A, S(<ID_A, A>, Spb(B))) \circ \alpha_3$

$Q = \{Auth(A, AS, Spb(B))\}$.

Proof:

(1) $Know(A, Spb(AS))$ [2-2]
(2) $Generate(A, <A, B>)$ [action]
(3) $Know(A, <A, B>)$ (2)[R-2]
(4) $Send(A, AS, <A, B>)$ (3) [1.1]
(5) $Know(AS, <A, B>)$ (4)[R-1]
(6) $Know(AS, B)$ (5)[6-1]
(7) $Generate(AS, S(CK, Spv(CA)))$ (6)[2-4]
(8) $Know(AS, S(CK, Spv(CA)))$ (7)[R-2]
(9) $Send(AS, A, S(CK, Spv(AS)))$ (8) [1.2]
(10) $Know(A, S(<B, Spb(B)>, Spv(AS)))$ (9)[R-1]
(11) $\neg Auth(A, AS, Spb(B))$ (1)(3) (10)[5-3][R-5]

The final conclusion is that A fails to authenticate the validity of $Spb(B)$ 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 $Spb(B)$ 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 AS issued the $Spb(B)$. 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 C request an copy of CA's key-exchange certificate. There are two principals in this instance: cardholder C and certificate authority CA.

$C \rightarrow CA$: $\alpha_1 = Send(C, CA, InitReq)$ (2.1)

$CA \rightarrow C$: $\alpha_2 = Send(CA, C, <CertS(CA), CertK(CA), S(<InitRes, H(InitRes)>, Spv(CA))>)$ (2.2)

$C \rightarrow CA$: $\alpha_3 = Send(C, CA, <E(RegFormReq, k), S(<AcctNum, k>, Kpb(CA))>)$ (2.3)

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

We begin the verification from the goal $Auth(C, CA, InitRes)$. The definitions of P, α and Q have the same meaning as above.

$P = \{Know(C, Spb(CA))\}$,

$\alpha = Generate(C, InitReq) \circ \alpha_1 \circ Generate(CA, <CertS(CA), CertK(CA), InitRes, S(H(InitRes), Spv(CA))>) \circ \alpha_2 \circ Generate(C, <E(RegFormReq, k), S(<AcctNum, k>, Kpb(CA))>) \circ \alpha_3$

$Q = \{Auth(C, CA, InitRes)\}$.

Proof:

(1) $Know(C, Spb(CA))$ [2-2]
(2) $Generate(C, InitReq)$ [action]
(3) $Know(C, InitReq)$ (2)[R-2]
(4) $Send(C, CA, InitReq)$ (3) [2.1]
(5) $Know(CA, InitReq)$ (4)[R-1]
(6) $Generate(CA, <CertS(CA), CertK(CA), InitRes, S(H(InitRes), Spv(CA))>)$ (5)[2-4][action]
(7) $Know(CA, <CertS(CA), CertK(CA), InitRes, S(H(InitRes), Spv(CA))>)$ (6)[R-2]
(8) $Send(CA, C, <CertS(CA), CertK(CA), InitRes, S(H(InitRes), Spv(CA))>)$ (7)[2.2]
(9) $Know(C, <CertS(CA), CertK(CA), InitRes, S(H(InitRes), Spv(CA))>)$ (8)[R-1]

(10)*Know(C, <CertS(CA), CertK(CA)>)* (9)[6-1]
(11)*Know(C, CertS(CA))* (10)[6-1]
(12)*Know(C, CertK(CA))* (10)[6-1]
(13)*Know(C, S(H(InitRes), Spv(CA)))* (9)[6-1]
(14) $\neg$*Auth(C, CA, InitRes)*
(1)(9)(13)[6-1][Theorem 1][R-5]

In the specifications of SET protocol, the Cardholder identifies the Chall-EE by deciding whether it is equal to the one sent in the *CardCInitReq*. Actually, if the challenge-response mechanisms work well the Chall-EE needs not be hidden from an intruder, but if not we have to wonder the efficiency of Chall-EE, such as the interception of the Chall-EE by intruder. Also, we have to concern the compromise of Cardholder. Thereby, we think the principal should verify that their messages are not replays by checking that $|Clock - T| < \Delta t_1 + \Delta t_2$. The SET protocol doesn't give detailed description on that so a special timestamp instead of Chall-EE is introduced here for the sake of security.

During the verification, *C* fails to authenticate the validity of *InitRes* sent by *CA*, thus the authentication should be halted immediately for the non-monotonic rule (R-5). The flaw detected for *InitRes* encrypted by *Spv(CA)* does not include the timestamp and identifier, so 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 the certificate is a long-term word, so the certificate cannot substitute for the identifier.

(1) C → *Z (CA): InitReq*
(2) *Z(C)* → *CA: InitReq'*
(3)*CA*→*Z(C): <CertS(CA), CertK(CA), InitRes', S(H(InitRes'), Spv(CA))>*
(2') *Z(C)* → *CA: InitReq''*
(3')*CA*→*Z(C):<CertS(CA), CertK(CA), InitRes'', S(H(InitRes''), Spv(CA))>*
(4)*Z(CA)*→*C:<CertS(CA), CertK(CA),InitRes', S(H(InitRes'), Spv(CA))>*

Here the intruder *Z* intercepts the initial request from *C* to *CA* and replaces it with a new initial request *initReq'*. It then sends the result to *CA* as message (2). *CA* replies with message (3). *Z* impersonates *C* to produces a new message (2') and sends it to *CA*. *CA* answers *C* with a corresponding message (3'), and then *Z* intercepts it. At last, *Z* impersonates *CA* to send an outdated message (4) that is intercepted by *Z* from message (3). *C* cannot authenticate the message since it does not include a timestamp and identifier. This attack is used continuously until *C* needs to bring about authentication between *C* and *CA* again. The best solution is to include a timestamp and identifier in the message sent by *CA*, even though you think certificate is secure. In fact, the protocol designer must be very careful to detect every possible subtle drawback when the protocol is in the design stage. Otherwise, if a protocol with flaws is placed in a practical environment (for instance stock-trading). It may cause a great loss if intruder detects the flaws.

These two instances convince that NDL is useful for verifying the secure protocols.

From the above observations, the proposed framework NDL is effective and promising. Example 1, derived from Needham and Schroeder's protocols for public key systems, is regarded as secure. Although author uses double handshake to assure the information remains secret during the communication, we have proved it has a flaw. Example 2 is drawn from SET protocol that is commonly thought to be a standard for future electronic commerce. In our approach, we detect a subtle flaw there. 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:*CA* believes that it is *C* who thought it was talking to *CA*.

6. Conclusions

The explosion of publicity of electronic commerce has enormously impacted on 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 defect in spite of a lot of researches on this subject. So a variety of formal methods have been developed for analysing secure protocols, for instance, BAN logic, GNY logic, BGNY logic, and AUTLOG logic. But the result is not as good as people expects.

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 suitable than the existing logics used in the analysis of secure transaction protocols.

In particular, the verification can automatically halts 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:

[1]Needham R. and Schroeder M., Using Encryption for Authentication in Large Networks of Computers, *Comm. of the ACM*, **21(12):** 993-999, Dec. 1978.
[2]Neuman B. and Ts'o T., Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, **32(9):**33-38, September 1994.
[3]Liu R., Visa sees a Smarter way, September 28, 1998.
[4]Burrows M., Abadi M., Needham R., A logic for Authentication, *ACM Transactions on Computer Systems*, **8(1):**18-36, February 1990.
[5]Gritzalis S., Security Protocols over open networks and distributed systems: Formal methods for their Analysis, Design, and Verification, *Computer Communications*, **22(8):**695-707, May 1999.
[6]Denning D., Sacco G., Timestamp in Key Distribution Protocols, *Communications of ACM* **24(8):**533-536, August 1981.
[7]Kailar R., Reasoning about Accountablity in Protocols for Electronic Commerce, *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, (1995) 236-250, IEEE Computer Society Press.
[8]Brackin S., A HOL Extension of GNY for Automatically Analyzing Cryptographic Protocols, *Proceeding of the 1996 IEEE Computer Security Foundations Workshop IX*, (1996) 62-76, IEEE Computer Society Press.
[9]Dolev D., Yao A., On the Security of Public Key Protocols, *IEEE Transaction on Information Theory*, 29(2):198-208, 1983.
[10]Meadows C., The NRL Protocol Analyzer: An overview, *Journal of Logic Programming*, **26(2):** 113-131, 1996.
[11]Millen J.C., The Interrogator Model, *Proceeding of the 1995 IEEE Symposium on Security and Privacy*, (1995) 251-260, IEEE Computer Society Press.
[12]Abadi M., Tuttle M., A semantics for a logic of authentication, *Proceedings of the 10th Symposium on Principles of Distributed Computing*, pages 201-216. ACM, August 1991.
[13]Gong L., Handling infeasible specifications of cryptographic protocols, *Proceedings of Computer Security Foundations Workshop IV*, pages 99-102, Franconia NH, June 1991, IEEE.
[14]Gong L., Needham R., and Yahalom R., Reasoning about belief in cryptographic protocols, *Proceeding of the Symposium on Security and Privacy*, pages 234-248, Oakland, CA, May 1990, IEEE.
[15]Gong L., Syverson P., Fail-Stop Protocols: An Approach to Designing Secure Protocols, *5th International Working Conference on Dependable Computing for Critical Applications*, pages 44-55, Sep, 1995.
[16]Kessler V., Wedel G., AUTLOG – an advanced logic of authentication, *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, Los Alamitors, IEEE Computer Society Press, pages 90-99, 1994.
[17] Meadows C., Syverson P., A formal specification of requirements for payment transactions in the SET protocol. Proceedings of Financial Cryptography 98, Lecture Notes in Comp. Sci. Springer-Verlag, 1998.
[18]SET Secure Electronic Transaction Specification, Book 1: *Business Description, Version 1.0*, May 31, 1997.
[19]SET Secure Electronic Transaction Specification, Book 2: *Programmer's Guide, Version 1.0*, May 31, 1997
[20]Leonard N., Foner A., security architecture for multi-agent matchmaking, *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS96)* , pages 80-86. AAAI Press, 1996.
[21]Huberman B., Franklin M., Hogg T., Enhancing privacy and trust in electronic communities, *Proceedings of ACM electronic commerce 99*, pages 78-86, 1999.
[22]Kohl J., Neuman B, "The Kerberos Network Authentication Service", Version 5 RFC, Drft No.4, Network Working Group, MIT Project Athena, December, 1990.
[23]Syverson P., A Taxonomy of Replay Attacks, *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 131-136. IEEE Computer Society Press, 1994.
[24]Kessler V., Neumann H., A sound logic for analysing electronic commerce protocols, *Proceeding of the 5th Eur. Sym. on Res. in Comp. Science, Lecture Notes in Comp Science*. Springer-Verlag, 1998.