1998 International Computer Symposium
Workshop on Cryptology and Information Security
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

SPECIFICATION OF CRYPTOGRAPHIC PROTOCOLS USING INTERVAL TEMPORAL LOGIC

Kong Wai Meng and Li Xiao Shan
Faculty of Science and Technology
University of Macau, Macau
E-mail:{m946320, fstxsl}@umac.mo

ABSTRACT

In this paper Interval Temporal Logic (ITL) [1] is used to specify cryptographic protocols. We choose *Kerberos* as a case study because of its apparent temporal characteristics. Using ITL, the communication protocol of *Kerberos* is specified formally. Inference rules of *timed belief* are presented. Therefore the temporal properties of *Kerberos* can be specified and verified.

1 INTRODUCTION

Cryptographic protocols are naturally related to time. For example, acceptance of obsolete messages can cause serious security problem. The trust relation between principals of protocol can vary time by time. Keys, especially session keys between communication agents, live within limited time. In [2] the model of timed belief is introduced. It shows that each belief eventually expires. Therefore, the treatment of time is one of the most important aspects in analyzing cryptographic protocols.

Interval Temporal Logic (ITL) [1] is a linear-time temporal logic with a discrete model of time. As mentioned above, the belief has a limited life. Thus it is useful to specify the temporal behavior of cryptographic protocols formally.

We choose *Kerberos* as a case study in this paper. *Kerberos*[3][4][5] is an authentication protocol with significant temporal properties. It uses *timestamp* as a *nonce* to verify the correctness of communication in protocol. The credential of authentication, called *Ticket*, has its limited life.

The remaining sections of the paper are organized as follows. Section 2 introduces *Kerberos* authentication protocol. ITL is briefly overviewed in Section 3. Section 4 gives the model of system and defines some notations. The protocol *Kerberos* is specified in ITL formally in Section 5. Section 6 presented the inference rules about timed belief. The property of *Kerberos* is described in Section 7. Finally, Section 8 draws a conclusion.

2 INTRODUCTION OF *KERBEROS*

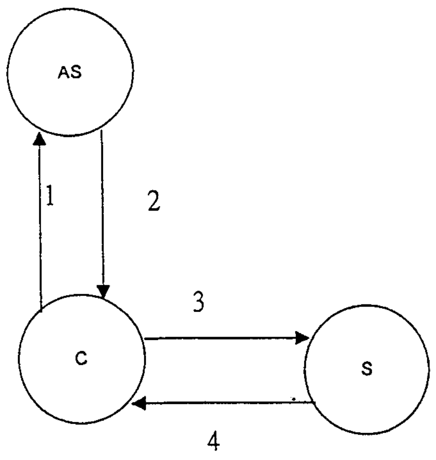
Kerberos developed as part of MIT's Project *Athena*, is a distributed authentication service that allows a process (a *client*) to run on behalf of a principal (a user) to prove its identity to a *server* without sending data across the network that might allow an attacker or the server to subsequently impersonate the principal. *Kerberos* optionally provides integrity and confidentiality for data sent between the client and server.

Authentication is the verification of the identity of a party who generates some data, and of the integrity of the data. A *client* is the party whose identity is verified. The *server* is the party who demands assurance of the principal's identity. The traditional authentication is *password-based*. However passwords can be collected particularly by eavesdropping. Password based authentication is also inconvenient because users do not want to enter a password each time when they access a network service. This has led to the *assertion-based* authentication.

The *Kerberos* Authentication System uses a series of encrypted messages to prove to a server that a client is running on behalf of a particular user. The *Kerberos* protocol (see Figure 1) is based partly on the Needham and Schroeder authentication protocol.

The client C and server S do not initially share an encryption key. Whenever C authenticates it to S it relies on the authentication server AS to generate a new encryption key $K_{C,S}$ and distribute it securely to both parties. This new encryption key $K_{C,S}$ is called a *session key*. The *Kerberos* ticket $TK_{C,S}$ is used to distribute it to S.

Kerberos Ticket $TK_{C,S}$ is a credential issued by AS,



1 Ticket request $C \rightarrow AS$: C, S, n

2 Ticket request $AS \rightarrow C$: $\{K_{C,S}, n\}_{K_{AS,C}}$, $\{TK_{C,S}\}_{K_{AS,S}}$

3 Authentication request $C \rightarrow S$: $\{A_C\}_{K_{C,S}}$, $\{TK_{C,S}\}_{K_{AS,S}}$

4 Authentication response $S \rightarrow C$: $\{A_C.\text{timestamp}\}_{K_{C,S}}$

C - Client

S - Server

AS -Kerberos Authentication Server

Figure 1 Simplified Kerberos protocol

encrypted with certain key for security. Among other information, the ticket contains the random session key $K_{C,S}$ that will be used for authentication of the principal to *S*, the names of the principal to whom the session key is issued with its *lifetime*. $TK_{C,S}$ is only valid within its lifetime, a finite time interval.

The ticket is sent to *C* who forwards it to *S* as part of the authentication request rather than sends directly to the server. Because $TK_{C,S}$ is encrypted with the server key $K_{AS,S}$, which is known only by *AS* and *S*, it is not possible for *C* to modify the ticket. The structure of $TK_{C,S}$ is:

$$<S, C, \text{lifetime}, K_{C,S}>$$

Authenticator A_C is another credential used in *Kerberos*, which proves that *C* is in possession of $K_{C,S}$ by encrypting some identity information in the session key. A_C contains identifier of this principal and the timestamp:

$$<C, \text{timestamp}>$$

In this protocol (Figure 1), *C* requires the ticket $TK_{C,S}$ and the session key $K_{C,S}$ for *S* with which it communicates. When *C* wishes to create an association with *S*, the client uses the ticket request and response, messages 1 and 2 in Figure 1, to obtain $TK_{C,S}$ from *AS*. In the request, *C* sends *AS* its claimed identity, the name of the server, and a nonce that will be used to match the ticket with respect to the request. In its response, the authentication server returns $K_{C,S}$ encrypted with the shared key between client and authentication server $K_{AS,C}$, which is generated from the password introduced by the user on *C*, together with $TK_{C,S}$ encrypted with the shared key $K_{AS,S}$ between *AS* and *S*.

Messages 3 and 4 in Figure 1 show the authentication request and response. The authentication request is through this exchange that *C* proves to *S* that it knows the session key $K_{C,S}$ embedded in $TK_{C,S}$. There are two parts to the authentication request, a ticket and an authenticator A_C, which contains the timestamp and the identifier of *C*.

In message 4, *S* generates an authentication response by extracting the timestamp from A_C, and returns it to *C*, all encrypted using $K_{C,S}$, to prove its identity to *C*.

At this point, *C* and *S* convince that the another is authentic. They share $K_{C,S}$ and it can be assumed to be safe that a reasonably recent message encrypted in that key originated with the other party.

To make sure that the received message is fresh, principals need to check the timestamp additionally since an attacker can intercept an authenticator and replay it later to impersonate the user. Thus the following mechanism is used: If the timestamp of the received message is within a specified window centered around the current time on the receiver, and if the timestamp has not been seen on other messages within that window, the message is accepted as authentic.

The description of *Kerberos* above reflects the nature of the protocol, instead of a completed description with all details.

In *Kerberos*, there are two important temporal characteristics. One is that timestamp is used as a nonce in the protocol to guarantee that messages are received within a specified 'time window'. Using this mechanism, the authenticity of messages is verified. It is considered that timestamp is also used as a nonce in message 1 and 2 to simplify the further discussion. The other is that the ticket used to distribute session key has its expiration time. In the remaining part of the paper, we will concentrate on specifying *Kerberos* and its characteristics.

3 INTRODUCTION OF ITL

Interval Temporal Logic (ITL) [1] is a temporal logic for specifying and reasoning the behavior of a dynamic system on discrete state intervals or periods of time. In ITL an interval σ is a sequence of states $\sigma_0, \sigma_1, \ldots, \sigma_{|\sigma|}$. Its length is $|\sigma|$. Each state maps variable A, B... to data values. ITL as an extension of the first order logic contains conventional logical operators $\wedge$, $\neg$, $\forall$ and $\exists$. Meanwhile it also has three temporal operators.

3.1 Primitive temporal operators

The operator *skip* has no operands and it is true in an interval if and only if the interval has length 1

```
skip   •   •
```

The formula *S;T* is true on an interval σ with the states σ_0, $\sigma_1 \ldots \sigma_{|\sigma|}$ if and only if the interval can be chopped into two sequential parts and they share a single state σ_k for $0 \le k \le |\sigma|$. The sub-formula *S* is true on the right part $\sigma_0 \ldots \sigma_k$ and the sub-formula T is true on the left part $\sigma_k \ldots \sigma_{|\sigma|}$.

```
S;T    •   •   •   •   •   •
       |     S     |   T   |
```

A formula S^* is true on an interval if and only if the interval can be chopped into zero or more sequential parts and the sub-formula *S* is true on each. An empty interval always satisfies any formula of the form S^*.

```
S*     •   •   •   •   •   •
       |   S   |   S   | S |
```

3.2 Others useful derived operators

Other ITL operators used in this paper, are defined as follows:

$I = a$. This formula is true on an interval σ if and only if the value of I at the start of σ equals a.

$\bigcirc S \equiv_{\text{def}} skip \,;\, S$. S holds on next state.

$\Diamond S \equiv_{\text{def}} true;\, S$. S sometimes must be true.

$\Box S \equiv_{\text{def}} \neg(\Diamond \neg S)$. S is never false. Thus S is always true.

$more \equiv_{\text{def}} skip$; true. Non-empty interval

$empty \equiv_{\text{def}} \neg\, more$. Empty interval

$fin\ S \equiv_{\text{def}} (empty \supset S)$. This formula is true if and only if S holds on the final state of the interval.

$halt\ S \equiv_{\text{def}} (S \equiv empty)$. If the formula holds, this is a terminate interval.

$stable\ e \equiv_{\text{def}} \exists a \bullet \Box(e = a)$. Here e's value does not change in the interval.

4 MODEL DEFINITION

In this section, we will develop the model of principal interaction. First we define a discrete time model. The elements involved in the communication and some notations about principals of the protocol communication are also formalized, such as messages and keys. Finally the essential interactions between principals and temporal relation notation are specified in ITL.

4.1 Time Model

We assume that there is a global clock in our system model. A global state variable T is used to indicate the clock time.

T is a monotonically increasing state variable as follows:

- $T \in Nat$, where $Nat = \{0,1,\ldots,n, n+1,\ldots\}$
- $\bigcirc T \ge T$

4.2 Messages, Keys and Encryption

The set of messages, *Msg* can be defined inductively as follows:

- An atomic token including keys, nonce, non-decomposable message and name of principal, is a message.
- If M is a message and K is a key, then the encryption of message $\{M\}_K$ is also a message.
- If $M_1,\ldots,M_n \in Msg$, then $(M_1, M_2,\ldots,M_n) \in Msg$, called a composed message.

In this paper we only concern the security property of the protocol, which are independent of the underlying message representation and encryption scheme. First we assume the messages are independent. Second we assume that the only way to obtain any information from an encrypted message is to have the appropriate key, i.e., the probabilistic factor in decryption is not considered.

4.3 Principals and state variables

Let *Principal* be the set of all concerned interacting principals. The principals interact with each other by sending and receiving messages.

Each principal's state is represented by a set of local state variable. V_p denotes the value of state variable V that is local to p, where $p \in Principal$.

The following state variables are used to formalize the behavior of principals:

- The state variable α_p represents the set of messages received by the principal p.
- To describe communication actions described in Section 2, the state variable β_p is introduced to store the set of messages in the buffer of transmission, which are not accepted by the principal and checked whether they are within the specified window or not.

4.4 Actions

Some atomic actions of essential communication are defined as the predicates on the state variables. To specify those actions involved in the protocol, we also use a global variable N (Network) to represent the communication network as a set of messages.

We usually need a particular action in ITL specification for indicating some state variables that do not change while other action executes. The definition of action *Unchanged* is given as follows:

$$Unchanged(\{V_1, V_2, \ldots, V_n\}) \equiv_{\text{def}} stable(V_1) \wedge stable(V_2) \wedge \ldots \wedge stable(V_n)$$

where $\{V_1, V_2, \ldots V_n\} \in \omega$ and ω is a set of all state variables involved in the behavior of the protocol.

Now *send, receive, write can be* defined as follows, in which $X, Y \in Principal$ and $M \in Msg$.

$$send(X, Y, M) \equiv_{\text{def}} \bigcirc N = N \cup \{(X, Y, M)\} \wedge Unchanged(\omega - \{T, N\})$$

It means that X sends a message M to Y through N. The tuple (X, Y, M) is obtained by N, i.e., the message will not be transmitted to the destination immediately. There is a delay caused by network.

$$\begin{aligned} receive(X, Y, M) \equiv_{\text{def}} & \bigcirc N = N - \{(Y, X, M)\} \wedge \\ & \{(Y, X, M)\} \in N \wedge \\ & \bigcirc\beta_X = \beta_X \cup \{(X, Y, M)\} \wedge \\ & Unchanged(\omega - \{T, N, \beta_X\}) \end{aligned}$$

The tuple of message is received from network medium by X. This tuple is written to the buffer of X. Now the message arrives the destination really through network.

After verifying the authenticity of received message, the principal actually obtains the message. Thus we define

$$\begin{aligned} write(X, Y, M) \equiv_{\text{def}} & \bigcirc\alpha_X = \alpha_X \cup \{(X, Y, M)\} \wedge \\ & \bigcirc\beta_X = \beta_X - \{(X, Y, M)\} \wedge \\ & Unchanged(\omega - \{T, \beta_X, \alpha_X\}) \end{aligned}$$

That is the message M sent from Y to X. While changing α_X by adding new messages, the message is cleared from buffer β_X.

5 THE SPECIFICATION OF KERBEROS

5.1 Assumption

1. The *Kerberos* authentication server has jurisdiction over the session keys among server, client and itself.
2. The case of concurrent ticket requests to AS is not considered.
3. The essential transmissions shown in Figure 1 must happen sequentially.
4. The principal out of C, S, AS will not be considered in this paper.

5. The communication medium used in the protocol always functions well.

5.2 Protocol specification

In *Kerberos*, we have:

$\{AS, S, C\} \in Principal$

where AS is *Kerberos* Authentication Server, S is Server (Application Server) and C is Client. Please refer to Figure 1.

We divide the four steps of protocol into two formulas: *TicketRequest* and *Authenticate*. The *TicketRequest* includes step 1 and step 2. It obtains the ticket for authentication. The step 3 and step 4 comprise *Authenticate* by which C identifies itself to S. The actions *Authenticate* could occur more than once, because the ticket can be used more than once in the authentication of C to S. The *Authenticate* will be stopped whenever the ticket expires. The action only runs while the invariant predicate *TicketValid* holds.

In order to make discussion simply, four essential message transmissions 1, 2 ,3 and 4 shown in Figure 1are specified separately as KB_1, KB_2, KB_3 and KB_4. We also use M_1, M_2, M_3, M_4 to denote the four messages in the essential transmissions:

$$M_1 = (C, S)$$
$$M_2 = (\{K_{C,S}\}_{K_{AS,C}}, \{TK_{C,S}\}_{K_{AS,S}})$$
$$M_3 = (\{A_c\}_{K_{C,S}}, \{TK_{C,S}\}_{K_{AS,S}})$$
$$M_4 = (\{A_C.timestamp\}_{K_{C,S}})$$

Considering the assumption 3 in Section 5.1, it is necessary to have a additional condition. It is:

$$(\forall i, j \in \{1, 2, 3, 4\} \bullet i \neq j \supset \Box \neg (KB_i \wedge KB_j))$$

Now the *Kerberos* protocol can be defined in ITL as follows:

$$KB \equiv_{\text{def}} ((TicketRequest; Authenticate^*) \wedge TicketValid)^*$$

Each essential transmission has two cases, *Success* and *Failure*. In the case of Success , the protocol continues to run, otherwise it stops. We specify KB_1 as follows:

$$\begin{aligned} KB_1 \equiv_{\text{def}} \exists\, t_0 \in Nat \bullet\ & send(C, AS, M_1) \wedge T = t_0\ ; \\ & receive(AS, C, M_1)\ ; \\ & (T \leq t_0 + d_1 \wedge write(AS, C, M_1)) \vee \\ & halt(T > t_0 + d_1) \end{aligned}$$

In KB_1, $K_{C,AS}$ is derived from the password introduced by user in the current session and C introduces this password to AS. Not affecting generality, we can assume that both C and AS believe that $K_{AS,C}$ is only known by them.

It is also necessary to guarantee that message must be received within the specified window. We use d_1 as the time period of the allowed receiving window. When the message is received, it must be verified whether it is still valid, i.e., the time delay is within the window d_1. Otherwise, the transmission is not sucessful, the protocol will halt, which can be specified as $halt\ (T > t_0 + d_1)$.

For the other three transmissions, we can similarly get :

$$\begin{aligned} KB_2 \equiv_{\text{def}} \exists\, t_0 \in Nat \bullet\ & send(AS, C, M_2) \wedge T = t_0\ ; \\ & receive(C, AS, M_2)\ ; \\ & (T \leq t_0 + d_2 \wedge write(C, AS, M_2)) \vee \\ & halt(T > t_0 + d_2) \end{aligned}$$

$$\begin{aligned} KB_3 \equiv_{\text{def}} \exists\, t_0 \in Nat \bullet\ & send(C, S, M_3) \wedge T = t_0\ ; \\ & receive(S, C, M_3)\ ; \\ & (T \leq t_0 + d_3 \wedge write(S, C, M_3)) \vee \\ & halt(T > t_0 + d_3) \end{aligned}$$

$$\begin{aligned} KB_4 \equiv_{\text{def}} \exists\, t_0 \in Nat \bullet\ & send(S, C, M_4) \wedge T = t_0\ ; \\ & receive(C, S, M_4)\ ; \\ & (T \leq t_0 + d_4 \wedge write(C, S, M_4)) \vee \\ & halt(T > t_0 + d_4) \end{aligned}$$

where d_2, d_3 and d_4 are the corresponding windows for the other three transmissions.

Therefore, the *TicketRequest* and *Authenticate* can be specified as follows.

$$TicketRequest \equiv_{\text{def}} KB_1\ ;\ Unchanged(\omega - \{T\})\ ;\ KB_2$$

$$Authenticate \equiv_{\text{def}} KB_3\ ;\ Unchanged(\omega - \{T\})\ ;\ KB_4$$

TicketValid specifies that the ticket only can be used within its *lifetime*. Whenever the ticket is invalid, a new ticket must be requested from *AS*.

The lifetime of the session key $K_{C,S}$ is part of $TK_{C,S}$ (See Section 2) is the time duration in the original *Kerberos* specification. In our time model we use a natural number l to represent it.

Therefore, the definition of *TicketValid* is given:

$$\begin{aligned}TicketValid \equiv_{\text{def}} & KB_1 ; \exists\, t_0 \in Nat \bullet \\ & (send(AS, C, M_2) \wedge T = t_0 \text{ ; true}) \wedge \\ & fin(T \leq t_0 + l)\end{aligned}$$

The predicate $send(AS, C, M_2)$ in *TicketRequest* is the place where the ticket appears in the protocol for the first time.

6 INFERENCE RULES ABOUT TIMED BELIEF

The following symbolic notation is similar to that in the BAN Logic [6]:

$X \mid\equiv P$: X believes P or is entitled to believe P. X may act as though P is true.

To express the belief evolved by time, the *Knowledge Set* (KS) introduced in [7] is used. However, we think the message sequence indicator M_i in the definition of KS makes generality lost. To adapt to ITL, its definition is modified as:

The knowledge set (KS) of a message content $M \in Msg$ is defined as $X \in$ KS (a) if a is recognizable by X at this moment. Here $M \in Msg$ and $X \in Principal$.

Comparing with the definition in [7], the major difference is that the message sequence indicator M_i is not used. ITL operators can express the change of KS in time.

If both X and Y believe that the session key K is shared with the another principal and trust all other principals which may have information about the key to the knowledge of the session key K. Thus we have

$$X \mid\equiv \{X, Y\} \in \text{KS}(K)$$
$$Y \mid\equiv \{X, Y\} \in \text{KS}(K)$$

For the simplicity, only the first-level belief [7] is involved in this paper. Now we give several inference rules about timed belief:

I_1: **Message meaning rule**

$$Y \mid\equiv \{X\} \in \text{KS}(K) \wedge write(X, Y, \{M\}_K) \supset$$
$$Y \mid\equiv \bigcirc\text{KS}(M) = \text{KS}(M) \cup \{X\}$$

That is, X already believes that Y knows their session key and obtains the message encrypted by this key, and after obtaining the encrypted message, X believes that Y knows the message.

I_2: **Belief Inclusion (1)**

$$write(X, Y, M) \supset X \mid\equiv \bigcirc\text{KS}(M) = \text{KS}(M) \cup \{Y, X\} \wedge$$
$$Y \mid\equiv \bigcirc\text{KS}(M) = \text{KS}(M) \cup \{X\}$$

That is , if X obtains a message from Y successfully, X believes that X and Y know the message and Y believes that X knows the message.

I_3: **Belief Inclusion (2)**

$$send(X, Y, M) \supset X \mid\equiv \bigcirc\, \text{KS}(M) = \text{KS}(M) \cup \{X\}$$

That is, X believes that X knows all messages sent by X.

I_4: **Knowledge Set Induction**

$$Y \mid\equiv \{X\} \in \text{KS}(\{M\}_K) \wedge Y \mid\equiv \{X\} \in \text{KS}(K) \supset$$
$$Y \mid\equiv \{X\} \in \text{KS}(M)$$

If Y believes that X knows the encrypted message and encryption key, then Y believes that X knows the decrypted one.

7 PROPERTY OF KERBEROS

In this section, the important property of *Kerberos* is given. We first describe them informally and then specify them formally.

Assumption S believes that the shared key between AS and S is already established, and C also believes that the shared key between AS and C is already established (Although $K_{C,AS}$ is derived from the password introduced by user in the current session a, AS can definitely know $K_{C,AS}$ by the password given by C and the specified algorithm). From the design of the *Kerberos*, *AS* manages the secrecy of all the shared key used and has the jurisdiction over those keys.

The formalized specification of *Assumption* is :

$$
\begin{aligned}
&Assumption \equiv_{def} \Box\,(\\
&\qquad S, AS \mid\equiv \\
&\qquad (\ \{AS,S\} \in \mathrm{KS}(K_{AS,S}) \wedge \\
&\qquad \{AS,S\} \in \mathrm{KS}(K_{AS,S})\ ) \wedge \\
&\qquad C, AS \mid\equiv \\
&\qquad (\{AS,C\} \in \mathrm{KS}(K_{AS,C}) \wedge \\
&\qquad \{AS,C\} \in \mathrm{KS}(K_{AS,C})\ )\ )
\end{aligned}
$$

Property about Timed Belief Client C and Server S are able to establish their session shared key if any pair of *send* and *receive* actions in the step 1 to 4 of the protocol is able to occur within the specified receiving window.

The formalized specification of the property is :

$$
\begin{aligned}
&Assumption \wedge KB \wedge \\
&(\forall X, Y \in Principal, \forall M \in Msg, \exists t_0 \in Nat \bullet \\
&send(X, Y, M) \wedge T = t_0\ ; \\
&receive(Y, X, M_3)\ ; (T \le t_0 + d\ \wedge write(Y, X, M))) \supset \\
&fin(\ C \mid\equiv \{C,S\} \in \mathrm{KS}\ (K_{C,S}) \wedge S \mid\equiv \{C,S\} \in \mathrm{KS}\ (K_{C,S})\ ))
\end{aligned}
$$

The predicate about *send*, *write* and *receive* has same significance with those used in the definitions of KB_{1-4}.

The predicate about *send*, *write* and *receive is used to guarantee the communication actions in the protocol execute successfully.* Similarly, d is the length of the receiving window of Y.

$fin(\ C \mid\equiv \{C,S\} \in \mathrm{KS}\ (K_{C,S}) \wedge S \mid\equiv \{C,S\} \in \mathrm{KS}\ (K_{C,S})\ )\ )$ means finally C and S believe that each other shares the session key $K_{C,S}$.

8 CONCLUSION

In this paper, by specifying *Kerberos*, we specify cryptographic protocol *Kerberos* using ITL. We also develop the model of specification and the base of verification by adopting some useful notations from other existing logics. Obviously it takes advantage of the specification in temporal characteristics of the protocol. Additionally, we can describe the behavior of protocol in a clear, provable and strict logic. Using the inference rules about timed belief, we can verify the property without much difficulty in our formal framework, which are omitted in this paper.

REFERENCE

[1] B. Moszkowski. *Executing Temporal Logic Program*, CAMBRIDGE UNIVERSITY PRESS, 1986.

[2] Nevin Heintze and J. D. Tygar. *Timed Models for Protocol Security*, CMU-CS-92-100, School of Computer, Carnegie Mellon University, 1992

[3] B. Clifford Neuman and Theodore Ts'o. *Kerberos: An Authentication Service for Computer Networks*, USC/ISI Technical Report number ISI/RS-94-399, 1994.

[4] John T. Kohl, B. Clifford Neuman an Theodore Y. Ts'o. *The Evolution of the Kerberos Authentication Service*, presented in the Spring 1991 European Conference, in Tromso, Norway.

[5] Jennifer G. Steiner, Clifford Neuman and Jeffrey I. Schiller. *Kerberos: An Authentication Service for Open Network System*, in the proceeding of the winter 1988 Usenix Conference, pp. 191-201, February 1988.

[6] Michael Burrows, Martin Abadi and Roger Needham. *A Logic of Authentication*, DEC System Research Center report No. 39, 1989

[7] Rajashekar Kailar and Virgil D.Gligor. *On Belief Evolution in Authentication Protocols*. In Proceedings of the Computer Security Foundations Workshop IV, pages 103-116. IEEE Computer Society press, Los Alamitos, California, 1991