A Timed Logic for Modeling and Reasoning about Security Protocols*

Xinfeng Lei[1] Rui Xue[1] and Ting Yu[2]

[1] State Key Laboratory of Information Security
Institute of Software, Chinese Academy of Sciences, Beijing, China
{leixinfeng, rxue}@is.iscas.ac.cn
[2] Department of Computer Science, North Carolina State University, USA
yu@csc.ncsu.edu

Abstract. Logical methods are usually considered suitable to express the static properties of security protocols while unsuitable to model dynamic processes or properties. However, a security protocol itself is in fact a dynamic process over time, and sometimes it is important to be able to express time-dependent security properties of protocols. In this paper, we present a new timed logic based on predicate modal logic, in which time is explicitly expressed in parameters of predicates or modal operators. This makes it possible to model an agent's actions, knowledge and beliefs at different and exact time points, which enables us to model both protocols and their properties, especially time-dependent properties. We formalize semantics of the presented logic, and prove its soundness.
We also present a modeling scheme for formalizing protocols and security properties of authentication and secrecy under the logic. The scheme provides a flexible and succinct framework to reason about security protocols, and essentially enhances the power of logical methods for protocol analysis. As a case study, we then analyze a timed-release protocol using this framework, and discover a new vulnerability that did not appear previously in the literature. We provide a further example to show additional advantages of the modeling scheme in the new logic.

1 Introduction

To formally analyze security protocols, a variety of methods, such as methods of logic [1–4], process calculus [5, 6], and strand spaces [7–10], have been proposed. Logical methods[1–4] are usually considered suitable to express properties rather than operational steps of a protocol. Process algebras is suitable to specify the operational behavior of protocols while unsuitable to express knowledge-related properties[11–13]. In this paper, we present a new logic for modeling and reasoning about security protocols. The proposed logic is suitable to model both a

* This work is partially supported by the Fund of the China Scholarship Council, NSFC grants (No. 60873260 and No. 60903210), the Natural Science Foundation of Jiangsu province of China (No.BK2008090), the 863 program (No. 2009AA01Z414), and the 973 Program (No. 2007CB311202).

security protocol itself and its properties, especially time-dependent properties. The key idea is to refine the logical method by explicitly using of time.

To see why time is necessary when modeling protocols, let us consider two simple scenarios. First, assume a logic without time involved is used to analyze a protocol. In this protocol, agent i receives message m. Then, the formula φ which means that i holds m is usually considered *false* before occurrence of the receiving action and *true* after that. It seems that both $\neg\varphi$ and φ are true in that logic, which makes the logic inconsistent. Where is the problem? Is there something being ignored? Obviously, it is time. One may argue that, without time involved, everything could be handled in a fixed time point by default. Let us see another scenario which takes the end of the protocol as the default time, when everything is discussed. Assume that agent i sends a message (in encrypted form) to j and hopes j to hold this message only after a specific time. Since everything is discussed in a fixed time, we even cannot express this property, let alone to analyze it.

Actually, in security protocols, time is an important factor related to their security. For one thing, each protocol is composed of a sequence of actions over time. Further, some protocols explicitly aim at a time-dependent requirement. For secrecy purposes, agents involved in protocols may require some messages be kept secret before a specific time [14],[15]. For authentication, agent's beliefs may change over time [16]. For fair non-repudiation, the lack of timeliness will lead to violation of fairness [17]. Therefore, it is necessary to take time into consideration when analyzing security protocols.

The logic provided in this paper is based on predicate modal logic. The time factor is expressed by parameters of predicates or modal operators. This makes it possible to model an agent's actions, knowledge and beliefs at different and exact time points, which enables us to describe the dynamic process of protocols and their properties, especially time-dependent properties. For example, in the scenarios mentioned above, we can use some formulas like $\neg\varphi(\tau)$ and $\varphi(\tau')$ instead of $\neg\varphi$ and φ, respectively, to avoid inconsistency and express similar assertions at different time τ and τ'. Moreover, since a security protocol is composed of a sequence of actions over time, to model a protocol, we can firstly model each action at a specific time, and then use a formula to reflect the time ordering. So, the proposed logic have advantages to model both protocols and their properties. Such advantages essentially enhance the expressiveness of the logical method in modeling and reasoning about security protocols.

One may wonder why a temporal logic (such as LTL or CTL) is not good enough for our purpose. In fact, a temporal logic is also more suitable to model properties, like safety or liveness. The time notion in it is expressed by temporal connectives such as F(some future state), G(all future state), U(until), etc. However, to model a security protocol, we often need to express a concrete process over a series of time, which is hard to be captured by today's temporal logics.

Related Work. Several authors [11–13] have recognized different natures between a protocol and its properties, and give some combined approaches to solving them. [11] aims at bridging the gap between operational and epistemic approaches to protocol analysis. This work proposes a combined framework which allows for modeling the behavior of a protocol in a process language and supports reasoning about properties expressed in a logic. [12] uses the set of traces generated by a process as models, and then defines a logic which has constructs for reasoning about the intruder's epistemic knowledge. [13] uses a hybrid method to verify security protocols, which firstly expresses the protocols in process calculus and then translates them into Horn clause in Horn logic. This method can take both advantages of process calculus and Horn logic. However, The translation from a process-style expression to a logic-style formula introduces some abstraction which makes some security protocols cannot be verified in it. For example, it fails to prove protocols that first need to keep some value secret and later reveal it[13]. Our work differs from the works mentioned above mainly in that the protocols and their security goals are all modeled in an unified logical framework instead of using two different frameworks to model protocols and properties respectively. Moreover, due to explicitly using of time, the proposed logic can capture time-dependent properties of a security protocol.

In logical method, after BAN logic, there have been a number of papers extending it which are called BAN-like logics ([2],[3],[4]). We can see that time is not used explicitly in these logics, so we cannot directly use them to analyze a time-dependent security protocol. Some logics have been proposed with time involved to express time-dependent properties. A logic proposed by Coffey and Saidha [18] which we refer to as the CS logic uses time explicitly. It originally aims at analysis of the public-key protocols, and mainly focuses on the authentication property of protocols. Moreover, it lacks of formal semantics and does not give a systematic approach to model protocols. Michiharu Kudo and Anish Mathuria[19] extend the CS logic to analyze a timed-release protocol. Their extended logic mainly aims at a specific protocol and also lacks in semantics. Another method to model time in security protocols is to use modal logic and temporal logic as well [20–22]. As be mentioned above, the time notion in temporal logic is abstract, and is unsuitable to model a protocol. Haack and Jeffrey [23] use timed Spi calculus to analyze protocols. Their intention is to model the time in cracking an encrypted message which is different from our goal. In the protocol composition logic (PCL) [24], a temporal ordering relation is used to strengthen the authentication properties by imposing ordering between actions of different participants. However, in PCL, time is not explicitly used, so it is not suitable to express a time-dependent properties. The increasing works about timed property ([25],[26],[27],[28],[29]) in recent years show importance of time in analyzing protocols.

In our own work [30], we have developed a time-dependent security protocol logic, which mainly focuses on expressing the time-dependent properties. This paper is an improved work of [30] in several aspects. The improvement includes putting off some symbols and adding new ones in syntax of the logic, renewing almost all groups of axioms. Also, a new method to model protocols, security goals, is provided. All these improvement and newly developed method make the logic more expressive and powerful in modeling the protocols and properties in reasoning about security protocols.

Other than a formal view[31] in analyzing security protocols, there is also a computational view[32, 33]. Each has its advantages and drawbacks. Some recent works focus on reconciling these two views [34–36] to provide a computational sound formal methods ([37][38][39]). Our work in this paper is within formal model. By elaborative definition, we expect it can provide computational soundness which is left to a further study.

Organization. The rest of the paper is organized as follows. Section 2 presents the syntax of the language and the axioms of the logic. In Section 4, a modeling scheme is provided to specify protocols and their security properties. An example is given in Section 5 to illustrate the process to reason about protocols. We conclude this paper in section 6 and discuss possible future directions following this. Due to space limit, we omit the formal semantics of proposed logic, the proof of soundness theorem, as well as further example to reason about protocols. A full version of the paper will be submitted to cryptology eprint archive at 'http://eprint.iacr.org/'.

2 The Proposed Logic

2.1 Syntax

To simplify the description, we use many-sorted variables in the proposed logic $\mathcal{L}$. There are four types of variables: $\mathbb{A}, \mathbb{T}, \mathbb{K}$ and $\mathbb{M}$, which, in intuitively, denote the agents, time, keys and messages respectively.

Definition 1 (Symbols in $\mathcal{L}$). *The language consists of the following symbols:*

1. Constants
   $\perp$ *contradiction;*
   e *constant of type* $\mathbb{A}$ *, a special agent which abstracts all the factors of environment including attackers;*
   τ_0 *constant of type* $\mathbb{T}$*, the starting time to run the protocol;*
   τ_c *constant of type* $\mathbb{T}$*, the time when the protocol completes.*
2. Variables
   $i, j, \ldots, i', j', \ldots$*: variables of type* $\mathbb{A}$*;*
   $k, k', k'', \ldots$*: variables of type* $\mathbb{K}$*;*
   $m, m', m'', \ldots$*: variables of type* $\mathbb{M}$*;*
   $\tau, \tau', \tau'', \ldots$*: variables of type* $\mathbb{T}$*.*
3. Functions
   $k^{-}_{(\cdot)} : \mathbb{A} \rightarrow \mathbb{K}$*, returns private key of an agent;*
   $k^{+}_{(\cdot)} : \mathbb{A} \rightarrow \mathbb{K}$*, returns public key of an agent;*
   $k_{(\cdot,\cdot)} : \mathbb{A} \times \mathbb{A} \rightarrow \mathbb{K}$*, returns shared key of two agents;*
   $[\cdot,\cdot] : \mathbb{M} \times \mathbb{M} \rightarrow \mathbb{M}$*, returns concatenation of two messages;*
   $[\cdot]_{(\cdot)} : \mathbb{M} \times \mathbb{K} \rightarrow \mathbb{M}$*, returns encrypted form of a message with a key;*
   $\tilde{(\cdot)} : \mathbb{K} \rightarrow \mathbb{K}$*, returns the reverse of a key;*

$+(\cdot,\cdot)$: $\mathbb{T} \times \mathbb{T} \to \mathbb{T}$, *returns sum of two time.* $+(\tau, \tau')$ *is often written as* $\tau + \tau'$.

4. Predicates
   The following are predicates and their informal meaning:
   - $<$: $\mathbb{T} \times \mathbb{T}$, *priority relation between two time points. For example,* $\tau < \tau'$ *means that the time* τ *is prior to time* τ'*;*
   - C : $\mathbb{M} \times \mathbb{M}$, *containing relation between two messages.* $C(m, m')$ *means that message* m *contains message* m'*;*
   - G : $\mathbb{A} \times \mathbb{T} \times \mathbb{M} \times \mathbb{M}$, *message getting.* $G(i, \tau, m, m')$ *means that agent* i *can get message* m' *from* m *at time* τ*;*
   - ν : $\mathbb{A} \times \mathbb{T} \times \mathbb{M}$, *message newly generating.* $\nu(i, \tau, m)$ *means that agent* i *newly generates a fresh message* m *at time* τ.
   - R : $\mathbb{A} \times \mathbb{T} \times \mathbb{M}$, *message receiving.* $R(i, \tau, m)$ *means that agents* i *receives message* m *at time* τ*;*
   - S : $\mathbb{A} \times \mathbb{T} \times \mathbb{M}$, *message sending.* $S(i, \tau, m)$ *means that agent* i *sends message* m *at time* τ*;*
   - H : $\mathbb{A} \times \mathbb{T} \times \mathbb{M}$, *message holding.* $H(i, \tau, m)$ *means that agent* i *holds message* m *at time* τ.
5. Equalities
   $=_\tau, =_a, =_k$ *and* $=_m$ *denote the equalities of times, agents, keys and messages respectively. Sometimes we use* $=$ *to include all the four cases.*
6. Quantifiers
   $\forall_\tau, \forall_a, \forall_k$ *and* $\forall_m$ *are the universal quantifiers of time, agent, key and message respectively. Sometimes we use* $\forall$ *to include all the four cases.*
7. Connectives
   $\neg$ *and* $\to$ *are negation and implication in first order logic.*
8. Modal operator
   $B_{i,\tau}$*: agent* i *believes something at time* τ.

The terms in the logic are introduced as follows.

Definition 2 (Terms). *Terms in* $\mathcal{L}$ *include* t_a, t_τ, t_k *and* t_m*, which denote the terms of type* $\mathbb{A}$*, type* $\mathbb{T}$*, type* $\mathbb{K}$ *and type* $\mathbb{M}$ *respectively, and they can be defined in Backus Naur form as follows:*

$$t_a ::= e \mid i \qquad\qquad t_\tau ::= \tau \mid \tau_0 \mid \tau_c \mid t_\tau + t_\tau$$
$$t_k ::= k \mid k^+_{t_a} \mid k^-_{t_a} \mid k_{t_a t_a} \mid \tilde{t_k} \qquad t_m ::= m \mid t_\tau \mid t_a \mid t_k \mid [t_m, t_m] \mid [t_m]_{t_k}$$

Now we look at formulas in $\mathcal{L}$.

Definition 3 (Formulas). *Formulas of* $\mathcal{L}$ *include atomic formula* φ_A *and formula* φ*. They are shown in Backus Naur form as follows:*

$$\begin{aligned}\varphi_A ::= {} & \bot \mid (t_\tau < t_\tau) \mid C(t_m, t_m) \mid G(t_a, t_\tau, t_m, t_m) \mid \nu(t_a, t_\tau, t_m) \mid S(t_a, t_\tau, t_m) \mid \\ & R(t_a, t_\tau, t_m) \mid H(t_a, t_\tau, t_m) \mid (t_a =_a t_a) \mid (t_\tau =_\tau t_\tau) \mid (t_k =_k t_k) \mid (t_m =_m t_m) \\ \varphi ::= {} & \varphi_A \mid \neg\varphi \mid \varphi \to \varphi \mid \forall_\tau \tau\, \varphi \mid \forall_a i\, \varphi \mid \forall_k k\, \varphi \mid \forall_m m\, \varphi \mid B_{i,\tau}\, \varphi\end{aligned}$$

To facilitate presentation, some notations are used as abbreviation in $\mathcal{L}$.

Definition 4 (Abbreviations).

1. *Messages Abbreviation:*
   - $[m_1, m_2, \ldots, m_\ell]$ *is an abbreviation of* $[m_1, [m_2, [\cdots, [m_{\ell-1}, m_\ell]] \cdots]$.
   - $[\![m]\!]$ *is an abbreviation of a message which contains* m. *More precisely, we have* $C([\![m]\!], m)$.
   - $[\![m]\!]_k$ *is an abbreviation of a encrypted message in which one can get* m *only through* $\tilde{k}$. *More precisely, we have* $C([\![m]\!]_k, m) \wedge \big(H(i, \tau, \tilde{k}) \leftrightarrow G(i, \tau, [\![m]\!]_k, m)\big)$.

   These notations allow us to focus on what we are interested in without listing all the other components.
2. *Abbreviation of the connective*
$$\varphi \vee \psi \triangleq \neg\varphi \to \psi \qquad\qquad \varphi \wedge \psi \triangleq \neg(\varphi \to \neg\psi)$$
$$\varphi \leftrightarrow \psi \triangleq (\varphi \to \psi) \wedge (\psi \to \varphi)$$
3. *Abbreviation of predicate*
$$\tau \leq \tau \triangleq (\tau < \tau') \vee (\tau =_\tau \tau')$$
$$\tau_1 *_1 \tau_2 *_2 \cdots \tau_{\ell-1} *_{\ell-1} \tau_\ell \triangleq (\tau_1 *_1 \tau_2) \wedge \cdots (\tau_{\ell-1} *_{\ell-1} \tau_\ell).$$
   where $*_i \in \{=_\tau, <, \leq\}$ *for* $i = 1, 2, \ldots, \ell-1$. *For example,* $\tau_1 =_\tau \tau_2 < \tau_3 \leq \tau_4$ *is an abbreviation of* $(\tau_1 =_\tau \tau_2) \wedge (\tau_2 < \tau_3) \wedge (\tau_3 \leq \tau_4)$.
4. *Abbreviation of quantifiers*
$$\exists x\, \varphi \triangleq \neg\forall x \neg\varphi$$

Also, we follow the conventional priority of all the connectives in a formula to evaluate.

2.2 Axioms and inference rules

The axioms of the logic consist of axiom schemas and inference rules. All the axioms include the instantiation and universal closure of the axiom schemas.

A 1 *All the axioms of predicate logic with many-sorted variables and equality.*

A 2 *Barcan formula*

$$\forall x B_{i,\tau}\varphi \leftrightarrow B_{i,\tau}\forall x\varphi \text{ (no free occurrence of } i, \tau \text{ in } \varphi)$$

where x *can be a variable of any type, and* $\forall$ *are the quantifiers corresponding to the type of* x.

The side condition assures the scopes of i and τ keep unchanged.

A 3 *Axioms of Monotonicity*

1. $H(i, \tau, m) \wedge (\tau < \tau') \rightarrow H(i, \tau', m)$
2. $G(i, \tau, m, m') \wedge (\tau < \tau') \rightarrow G(i, \tau', m, m')$
3. $B_{i,\tau}\, \varphi \wedge (\tau < \tau') \rightarrow B_{i,\tau'}\, \varphi$ *(no free occurrence of τ, τ' in φ)*

$A3$ states that knowledge, abilities, and beliefs of an agent will not be lost over time. One may argue $A3$,3) with the following example. If agent i believes that he does not hold some message, then he will believe his 'not holding' even when this message is sent to him. Of course this is not the case. Actually, considering that there is no free occurrence of τ in φ, if agent i believes that he does not hold message m at time τ, and he gets m later, say τ', then he still believes that he does not hold m at τ, but holds m at τ'.

A 4 *Axiom of time*

$$(\tau < \tau') \wedge (\tau' < \tau'') \rightarrow (\tau < \tau'')$$

A4 indicates that the relation $<$ is transitive.

A 5 *Axioms of key*

1. $H(i, \tau, k_i^-)$
2. $H(j, \tau, k_i^-) \rightarrow (j =_a i)$
3. $H(i, \tau, k_{ij}) \wedge H(j, \tau, k_{ij})$
4. $H(i', \tau, k_{ij}) \rightarrow (i' =_a i) \vee (i' =_a j)$
5. $\tilde{k}_i^+ =_k k_i^- \wedge \tilde{k}_i^- =_k k_i^+ \wedge \tilde{k}_{ij} =_a k_{ij}$

This group of axioms is about long-term keys, including symmetric key and asymmetric keys, which formulate the perfect encryption about symmetric and asymmetric encryption.

A 6 *Axioms of getting messages*

1. $G(i, \tau, m, m)$
2. $G(i, \tau, [m']_k, m) \leftrightarrow H(i, \tau, \tilde{k}) \wedge G(i, \tau, m', m)$
3. $G(i, \tau, [m, m'], m'') \leftrightarrow G(i, \tau, m, m'') \vee G(i, \tau, m', m'')$
4. $G(i, \tau, m, m') \wedge G(i, \tau, m', m'') \rightarrow G(i, \tau, m, m'')$

A 7 *Axioms of message containment*

1. $G(i, \tau, m, m') \rightarrow C(m, m')$
2. $C(m, m') \wedge C(m', m'') \rightarrow C(m, m'')$

The former one focuses on the relationship between message getting and message containing, and the later one indicates that C is transitive.

A 8 *Axioms of actions*

1. $\nu(i, \tau, m) \wedge \nu(i', \tau', m') \rightarrow m \neq_m m'$
2. $R(i, \tau, [\![m]\!]_{k_j^-}) \rightarrow \exists_\tau \tau' \big(\tau' < \tau \wedge H(j, \tau', m) \wedge S(j, \tau', [\![m]\!])\big)$

3. $\nu(i,\tau,m) \wedge S(i,\tau,[\![m']\!]_{k_j^+}) \wedge C(m',m) \wedge R(i,\tau',m'') \wedge G(i,\tau',m'',m) \rightarrow$
$\quad \exists_\tau \tau_1 \exists_\tau \tau_2 \big(\tau < \tau_1 \leq \tau_2 < \tau' \wedge R(j,\tau_1,[\![m']\!]_{k_j^+}) \wedge S(j,\tau_2,[\![m]\!])\big)$

4. $R(i',\tau,[\![m]\!]_{k_{ij}}) \rightarrow \exists_\tau \tau' \big(\tau' < \tau \wedge S(i,\tau',[\![m]\!]) \wedge H(i,\tau',m)\big)$
$\quad \vee \exists_\tau \tau'' \big(\tau'' < \tau \wedge S(j,\tau'',[\![m]\!]) \wedge H(j,\tau'',m)\big)$

5. $(i \neq_a j) \wedge \nu(i,\tau,m) \wedge H(j,\tau',[\![m]\!]) \rightarrow$
$\quad \tau < \tau' \wedge \exists_\tau \tau_1 \exists_\tau \tau_2 \big(\tau \leq \tau_1 < \tau_2 \leq \tau' \wedge S(i,\tau_1,[\![m]\!]) \wedge R(j,\tau_2,[\![m]\!])\big)$

This group of axioms describes the properties about actions, Often used for authentication or non-repudiation purposes. A8,1) indicates that a message comes from a *new* action is unique. A8,2) indicates that a message in private-key-encrypted form reveals its original sender. A8,3) implies that a message in public-key-encrypted form can designate its uniquely possible receiver. Furthermore, when a newly generated message is sent in such an encrypted form and sent back in a can-be-decrypted form later, one can assure that the message must have passed through that designated receiver(note that a newly generated message is unique). A8,4) states that a message in shared-key-encrypted form reveals its possible sources. A8,5) shows that a newly generated message will never be held by someone before the time, and if it is held by another agent later, then it must come from its generator.

A 9 *Axioms of holding messages*

1. $\nu(i,\tau,m) \vee S(i,\tau,m) \vee R(i,\tau,m) \rightarrow H(i,\tau,m)$
2. $H(i,\tau,m) \wedge G(i,\tau,m,m') \rightarrow H(i,\tau,m')$
3. $H(i,\tau,m) \wedge H(i,\tau,m') \leftrightarrow H(i,\tau,[m,m'])$
4. $H(i,\tau,m) \wedge H(i,\tau,k) \rightarrow H(i,\tau,[m]_k)$

A9 summarizes in what cases one holds a message.

A 10 *Axioms of belief*

1. $B_{i,\tau}(\varphi \rightarrow \psi) \wedge B_{i,\tau}\varphi \rightarrow B_{i,\tau}\psi$ *2.* $G(i,\tau,m,m') \rightarrow B_{i,\tau}G(i,\tau,m,m')$
3. $\nu(i,\tau,m) \rightarrow B_{i,\tau}\nu(i,\tau,m)$ *4.* $S(i,\tau,m) \rightarrow B_{i,\tau}S(i,\tau,m)$
5. $R(i,\tau,m) \rightarrow B_{i,\tau}R(i,\tau,m)$ *6.* $H(i,\tau,m) \rightarrow B_{i,\tau}H(i,\tau,m)$
7. $H(i,\tau,m) \wedge H(i,\tau,k) \rightarrow B_{i,\tau}C([m]_k,m)$

A10,1) is a variant of K axiom in modal logic. A10,2) means that an agent believes his ability to get messages. The followed four axioms mean that an agent believes what is done and what is held by himself. The last one implies that when an agent can construct an encrypted message with another message and key, then he believes the encrypted message contains the message (not including the encrypted key) used to construct it.

IR Inference Rule

$$1.\ \frac{\varphi \rightarrow \psi,\ \varphi}{\psi} \qquad\qquad 2.\ \frac{\vdash \varphi}{\vdash B_{i,\tau}\varphi}$$

where $\vdash \varphi$ is an abbreviation for $\vdash_{\mathcal{L}} \varphi$ which means that φ is a theorem in logic $\mathcal{L}$. When φ is deducible from a set of formulas Γ, we say that φ is a deduction consequence of Γ, written as $\Gamma \vdash \varphi$

3 SEMANTICS

To give the semantics of $\mathcal{L}$, we firstly define model in the logic. The model is based on a Kripke structures and is defined step by step with a series of definitions. The semantics of the formulas are given afterwards.

3.1 Model

Definition 5 (Domain). *We define the domain of the model as a set* $\mathcal{D} = \mathcal{A} \cup \mathcal{T} \cup \mathcal{K} \cup \mathcal{M}$, *where*

1. $\mathcal{A}$ *is the set of agents, i.e., the (possible) participants of the protocol;*
2. $\mathcal{T}$ *is a set of times. In our method, we use concrete time, or alternatively the integer to model time.* $<$ *is an order relation on* $\mathcal{T}$*;*
3. $\mathcal{K}$ *is a set of keys. Keys can be divided into public keys, private keys, shared keys and session keys, etc. They can be used to encrypt, decrypt, sign and verify some messages. In this model, we assume that a private key can only be known by its owner, and a shared key can only be known by those who share it. If a message encrypted by one key can be decrypted by another key, we say these two keys are reverse of each other. So, as usual, the public key and private key of the same agent are reverse, and a shared key is also considered as a reverse key of itself;*
4. $\mathcal{M}$ *is a set of messages including atomic messages, encrypted messages and combined messages. An atomic message is a message which has not be combined or encrypted, including agent, time, key, nonce, and single plaintext. Let* m, m' *be any two messages and* k *be any key, then encrypted messages are defined as the messages encrypted by* k, *written as* $[m]_k$, *and the combined messages are defined as the concatenation of* m *and* m', *written as* $[m, m']$.

Definition 6 (Message Extraction). *Given* $K \subseteq \mathcal{K}$ *and* $M \subseteq \mathcal{M}$, *we define* $\text{extract}_K(M)$ *as the set of messages which can be extracted from* M *with the help of* K, *and is recursively defined as follows:*

1. *If* $m \in M$, *then* $m \in \text{extract}_K(M)$*;*
2. *If* $[m, m'] \in \text{extract}_K(M)$, *then* $m \in \text{extract}_K(M)$ *and* $m' \in \text{extract}_K(M)$*;*
3. *If* $[m]_k \in \text{extract}_K(M)$, $\tilde{k} \in K$ *or* $\tilde{k} \in \text{extract}_K(M)$, *then* $m \in \text{extract}_K(M)$.

$\text{extract}_K(\{m\})$ is denoted as $\text{extract}_K(m)$. Message extraction can be used to interpret the predicate G.

Definition 7 (Message Construction). *Given* $K \subseteq \mathcal{K}$ *and* $M \subseteq \mathcal{M}$, *we define* $\text{construct}_K(M)$ *as the set of messages which can be constructed with the help of* M *and* K. *Formally, we recursively define* $\text{construct}_K(M)$ *as follows:*

1. *If* $m \in M$, *then* $m \in \text{construct}_K(M)$*;*
2. *If* $m, m' \in \text{construct}_K(M)$, *then* $[m, m'] \in \text{construct}_K(M)$*;*
3. *If* $m \in \text{construct}_K(M)$, *and* $k \in K$, *then* $[m]_k \in \text{construct}_K(M)$.

$\mathrm{construct}_K(\{m\})$ is denoted as $\mathrm{construct}_K(m)$. Message construction can be used to define message reconstruction and message containing.

Definition 8 (Message Reconstruction). *Given $K \subseteq \mathcal{K}$ and $M \subseteq \mathcal{M}$, we define* $\mathrm{reconstruct}_K(M)$ *as the set of messages which can be reconstructed with the help of M and K. Formally, we recursively define* $\mathrm{reconstruct}_K(M)$ *as follows:*

1. *If $m \in \mathbb{M}$, then $m \in \mathrm{reconstruct}_K(M)$;*
2. *If $m \in \mathrm{construct}_{K \cup key(extract_K(M))}(extract_K(M))$, then $m \in \mathrm{reconstruct}_K(M)$.*

$\mathrm{reconstruct}_K(\{m\})$ is denoted as $\mathrm{reconstruct}_K(m)$. Message reconstruction can be used to interpret the predicate H.

Definition 9 (Message Containment). *Given a message m, we call* $\mathrm{contain}(m)$ *as the set of messages which are contained in message m. That is, either someone can extract them from m, or someone can construct m with them. Formally, we define* $\mathrm{contain}(m)$ *as follows:*

1. *If there exists a key set K such that $m' \in \mathrm{extract}_K(m)$, then $m' \in \mathrm{contain}(m)$;*
2. *If there exists a key set K and a message m', such that $m \in \mathrm{construct}_K(m')$, then $m' \in \mathrm{contain}(m)$.*

Definition 10 (State machine of protocol). *A state machine of protocol is defined as a 4-tuple (Q, E, δ, q_0), in which*

1. **Q** *is the set of agents' states. The state of agent i at time τ, written as q_i^τ, is presented by the original messages and keys held by agent i till time τ, written as M_i^τ and K_i^τ respectively. Here, we use the* original message *to mean the message which keeps its form when it is received without any further reconstruction. So, we have $q_i^\tau = (M_i^\tau, K_i^\tau)$, especially, $q_i^{\tau_0} = (M_i^{\tau_0}, K_i^{\tau_0})$ is the initial state of agent i. q_0 is a vector of each agent's initial state, which is used to record the messages and keys initially held by each agent.*
2. **E** *is the set of actions. In a protocol, an agent can execute three types of actions:* $\mathrm{send}(m), \mathrm{receive}(m)$ *and* $\mathrm{new}(m)$, *which mean sending message , receiving message and newly generating a fresh value respectively. A fresh value is unique. That is, two newly generated messages will never be equal. To simplify our analysis, we assume that an agent can execute several actions with different types simultaneously; for example, an agent can execute a* receive *action and a* send *action at the same time, but is not allowed to execute several* receive *actions simultaneously. Generally $e, e_1, e_2, \ldots$ range over the single action, and α_i^τ denotes the set of actions agent i executes at time τ.*
   Furthermore, we can define the concept of action history. The action history of an agent is a finite sequence of action sets in which the agent has executed up to some moment in time. The set of action histories is denoted by Σ. Often, we use $\sigma_i^\tau =< \alpha_i^{\tau_0} \alpha_i^{\tau_1} \cdots \alpha_i^\tau >$ to denote the action history of agent i at time τ. An action e is said to appear in action history σ_i^τ, written as $e \in \sigma_i^\tau$, if there is a set of actions α in σ_i^τ and $e \in \alpha$.

With action history, we can define two cases of sending message, i.e., firstly send *and* forward. *An agent i is said to firstly send message m means that, there is a message $m' \in \mathcal{M}$ such that $m \in \text{contain}(m'), \text{send}(m') \in \sigma_i^\tau$, and for all $i' \in \mathcal{A}, \tau' \in \mathcal{T}, m'' \in \mathcal{M}$, if $m \in \text{contain}(m'')$ and $\text{send}(m'') \in \sigma_{i'}^{\tau'}$, then $\tau \leq \tau'$. Otherwise, we say that agent i only forwards message m;*

3. *$\delta : Q \times 2^E \to Q$, the function of state transition. More specifically, let $q_i^\tau = (M_i^\tau, K_i^\tau)$, and τ' be the next time after τ when some actions occur, we have*

$$q_i^{\tau'} = (M_i^{\tau'}, K_i^{\tau'}) = \delta(q_i^\tau, \alpha_i^{\tau'}) = \begin{cases} (M_i^\tau, K_i^\tau) & \text{if } \alpha_i^{\tau'} = \{\text{send}(m)\} \\ \big(M_i^\tau \cup \{m\}, V\big) & \text{if } \alpha_i^{\tau'} = \{\text{receive}(m)\} \\ \big(M_i^\tau \cup \{m\}, K_i^\tau \cup \text{key}(\{m\})\big) & \\ & \text{if } \alpha_i^{\tau'} = \{\text{new}(m)\} \end{cases}$$

where $V \triangleq K_i^\tau \cup \text{key}(\text{extract}_{K_i^\tau}(M_i^\tau \cup \{m\}))$ and $\text{key}(\cdot)$ is a function to select keys from a set of messages. Given a set of messages M, we say $k \in \text{key}(M)$ if and only if $k \in M$ and $k \in \mathcal{K}$. When agent i does more than one action at time τ, namely, $\alpha_i^\tau = \{e_1, e_2, \ldots, e_n\} (2 \leq n \leq 3)$, then, we define

$$\delta(q_i^\tau, \alpha_i^{\tau'}) = \cup_{e \in \alpha_i^{\tau'}} \delta(q_i^\tau, \{e\})$$

in which the union of the states is defined below. Let $q' = (M', K')$ and $q'' = (M'', K'')$, then

$$q' \cup q'' = (M' \cup M'', K' \cup K'')$$

Naturally, we can extend the state transition function to a similar function which can be applied to action history instead of action set, i.e., $\overline{\delta} : Q \times \Sigma \to Q$, then, we have,

$$q_i^\tau = \overline{\delta}(q_i^{\tau_0}, \sigma_i^\tau) = \delta(\cdots \delta(\delta(q_i^{\tau_0}, \alpha_i^{\tau_1}), \alpha_i^{\tau_2}), \cdots, \alpha_i^\tau)$$

From Definition 10, the state of an agent at a time tells only what is held by the agent at that time. But it cannot capture how this state is formed. Moreover, the execution of an action does not necessarily lead to the change of the state. To reflect the protocol precisely, besides the agent's local state, its action history is involved. This idea can help us define the concept of world.

Definition 11 (World). *We divide world into two local worlds and two global worlds.*

The local world of agent i at time τ is defined as $\omega_i^\tau = (q_i^{\tau_0}, \sigma_i^\tau)$, where $q_i^{\tau_0}$ is the initial state of the agent i, and σ_i^τ is the action history of agent i at time τ.

The local world of agent i is composed of the local worlds of agent i at all the time points in protocol, written as $\omega_i = (\omega_i^{\tau_0}, \omega_i^{\tau_1}, \cdots, \omega_i^{\tau_c})^T$.

The global world at a specific time τ is composed of local worlds of all the agents at time τ, written as $\omega^\tau = (\omega_i^\tau, \omega_j^\tau, \cdots, \omega_e^\tau)$.

The global world of a protocol (abbr. as world) is composed of the local worlds, written as ω, denoted as follows:

$$\begin{pmatrix} \omega_i^{\tau_0} & \omega_j^{\tau_0} & \cdots & \omega_e^{\tau_0} \\ \omega_i^{\tau_1} & \omega_j^{\tau_1} & \cdots & \omega_e^{\tau_1} \\ \vdots & \vdots & \vdots & \vdots \\ \omega_i^{\tau_c} & \omega_j^{\tau_c} & \cdots & \omega_e^{\tau_c} \end{pmatrix} = (\omega_i \ \omega_j \ \cdots \ \omega_e) = \begin{pmatrix} \omega^{\tau_0} \\ \omega^{\tau_1} \\ \vdots \\ \omega^{\tau_c} \end{pmatrix}$$

In the following, we use ω_i^τ to denote the local world of agent i at time τ, and $q_i^\tau(\omega)$ the local state of agent i at time τ in world ω. Similar understanding can be applied to $\alpha_i^\tau(\omega), \sigma_i^\tau(\omega), M_i^\tau(\omega)$ and $K_i^\tau(\omega)$.

Definition 12 (Possible World). *We define W as the set of all the possible worlds. A world ω is called a possible world, i.e. $\omega \in W$, if and only if it meets the following conditions:*

1. *If* $\text{new}(m) \in \alpha_i^\tau(\omega)$*, then*
   (a) m is atomic; and
   (b) for any $i' \neq i, \tau' \neq \tau$, we have $\text{new}(m) \notin \alpha_{i'}^{\tau'}(\omega)$*; and*
   (c) for any $i' \in \mathcal{A}, \tau' < \tau$, we have $m \notin \text{reconstruct}_{K_{i'}^{\tau'}(\omega)}(M_{i'}^{\tau'}(\omega))$.
2. *If* $\text{send}(m) \in \alpha_i^\tau(\omega)$*, then* $m \in \text{reconstruct}_{K_i^\tau}(M_i^\tau(\omega))$*. Especially, an atomic message m is firstly sent (see Definition 10 2)) by agent i at time τ, then* $\text{new}(m) \in \alpha_i^\tau(\omega)$.
3. *If* $\text{receive}(m) \in \alpha_i^\tau(\omega)$*, then, there must exist $i' \neq i, \tau' \leq \tau$, such that* $\text{send}(m) \in \alpha_{i'}^{\tau'}$*. Especially, if* $m' \in \text{contain}(m)$ *and* $\text{receive}(m) \in \alpha_i^\tau(\omega)$*, then, there must exist $i' \in A, \tau' < \tau$ and agent i' firstly send m in ω.*

Informally, 1a) means that a newly generated message must be an atomic message; 1b) means that a message can only be generated once; and 1c) means that no agent can hold a newly generated message before it is generated.

The item 2) means no holding, no sending. Also, if an agent has firstly sent an atomic message m at τ, he must generate this message at τ. It seems that m is not necessarily being generated at τ. Maybe, it can be generated before τ, but never be sent till τ. This is possible, but makes no difference in essence.

Intuitively, the item 3) means no sending, no receiving. It also implies that a received message may be forwarded many (but finite) times, but there must exist someone firstly sends it.

Definition 13 (Accessible Relationship). *Accessible relationship between possible worlds is written as $R_i^\tau \subseteq W \times W$. Let $\omega, \omega' \in W$, then*

$$\omega R_i^\tau \omega' \quad \textit{iff} \quad \omega_i^\tau = {\omega'}_i^\tau$$

We are finally ready to define the model.

Definition 14 (Model). *The model of $\mathcal{L}$ is defined as $\mathfrak{M} = (W, \{R_i^\tau\}_{i \in \mathcal{A}, \tau \in \mathcal{T}}, D, I)$ which is composed of the following components:*

1. *W, the set of possible worlds which is defined in Definition 12. The possible worlds are often written as $\omega', \omega'', \omega_1, \omega_2, \cdots$.*
2. *D, the domain of the proposed logic which is defined in Definition 5. Here we assume that $\mathcal{D} = \mathcal{A} \cup \mathcal{T} \cup \mathcal{K} \cup \mathcal{M}$ be the fixed domain of all the possible worlds.*
3. *R_i^τ, the accessible relationship which is defined in Definition 13. Different from the normal modal logic, R_i^τ is not a single relation, but a set of relations. That is, for any $i \in A, \tau \in T$, there exists a definition of accessible relationship.*
4. *I, a function of interpretation which maps the symbols of constant, function, predicate and modal operator into its counterpoints in model $\mathfrak{M}$.*
   - *For each constant c, $I(c)$ is an element of the corresponding domain, also denoted as $\overline{c}$;*
   - *For each function f, $I(f)$ is a function on the corresponding domain, also denoted as $\overline{f}$;*
   - *For each predicate P, $I(P)$ is a predicate on the corresponding domain (equality and modal operator can be considered as special predicate).*

For convenience, in the following, some syntax symbols will be self explained and conduct no confusions within the contexts to judge them to be in syntax form or semantic form.

3.2 Semantics

We are now in a position to compute a truth value for all formulas in $\mathcal{L}$.

Definition 15 (Assignment and Designation). *Assignment is defined as a mapping $s : \{v_1, v_2, \ldots\} \to \mathcal{D}$, where v_i is a variable, and is used to map variables of type $\mathbb{A}, \mathbb{T}, \mathbb{K}, \mathbb{M}$ to the sub-domain $\mathcal{A}, \mathcal{T}, \mathcal{K}, \mathcal{M}$, respectively.*

For such an s, we denote by $s[x \mapsto d]$ the mapping which maps x to d, and any other variable y to $s(y)$.

Designation of terms can be obtained by extending the assignment s. We define $\overline{s}$ as the mapping from the set of terms to its domain, then, to any term t, we have

$$\overline{s}(t) = \begin{cases} s(v) & \text{when } t \text{ is variable } v \\ \overline{c} & \text{when } t \text{ is constant } c \\ \overline{f}^{(n)}(\overline{s}(t_1), \ldots, \overline{s}(t_n)) & \text{when } t = f^{(n)}(t_1, \ldots, t_n) \end{cases}$$

We now define the semantics to the formulas. It is defined in conventional approach by computing a truth value.

Definition 16 (Semantics of the Formulas). *Let $\models^s_{\mathfrak{M},\omega} \varphi$ indicate that formula φ is true in the possible world ω of model $\mathfrak{M}$ under an assignment s. It will be abbreviated as $\models^s_\omega \varphi$ when $\mathfrak{M}$ defaulted. The semantics of the formulas are defined as follows:*

1. $\not\models^s_\omega \bot$.
2. $\models^s_\omega (t_\tau < t'_\tau)$ *iff* $\overline{s}(t_\tau) < \overline{s}(t'_\tau)$.
3. $\models^s_\omega C(t_m, t'_m)$ *iff* $\overline{s}(t'_m) \in \mathrm{contain}(\overline{s}(t_m))$.
4. $\models^s_\omega G(t_a, t_\tau, t_m, t'_m)$ *iff* $\overline{s}(t'_m) \in \mathrm{extract}_{K^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)}\big(\overline{s}(t_m)\big)$.
5. $\models^s_\omega H(t_a, t_\tau, t_m)$ *iff* $\overline{s}(t_m) \in \mathrm{reconstruct}_{K^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)}\big(M^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)\big)$.
6. $\models^s_\omega \nu(t_a, t_\tau, t_m)$ *iff* $\mathrm{new}(\overline{s}(t_m)) \in \alpha^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)$.
7. $\models^s_\omega S(t_a, t_\tau, t_m)$ *iff* $\mathrm{send}(\overline{s}(t_m)) \in \alpha^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)$.
8. $\models^s_\omega R(t_a, t_\tau, t_m)$ *iff* $\mathrm{receive}(\overline{s}(t_m)) \in \alpha^{\overline{s}(t_\tau)}_{\overline{s}(t_a)}(\omega)$.
9. $\models^s_\omega (t_a =_a t'_a)$ *iff* $\overline{s}(t_a) =_a \overline{s}(t'_a)$;
   $\models^s_\omega (t_\tau =_\tau t'_\tau)$ *iff* $\overline{s}(t_\tau) =_\tau \overline{s}(t'_\tau)$;
   $\models^s_\omega (t_k =_k t'_k)$ *iff* $\overline{s}(t_k) =_k \overline{s}(t'_k)$;
   $\models^s_\omega (t_m =_m t'_m)$ *iff* $\overline{s}(t_m) =_m \overline{s}(t'_m)$.
10. $\models^s_\omega \neg\varphi$ *iff* $\not\models^s_\omega \varphi$.
11. $\models^s_\omega (\varphi \to \psi)$ *iff* $\not\models^s_\omega \varphi$ *or* $\models^s_\omega \psi$.
12. $\models^s_\omega \forall_a i\varphi(i)$ *iff for all* $j \in \mathcal{A}$, $\models^{s[i\mapsto j]}_\omega \varphi[j]$;
    $\models^s_\omega \forall_\tau \tau\varphi(\tau)$ *iff for all* $\eta \in \mathcal{T}$, $\models^{s[\tau\mapsto\eta]}_\omega \varphi[\eta]$;
    $\models^s_\omega \forall_k k\varphi(k)$ *iff for all* $\kappa \in \mathcal{K}$, $\models^{s[k\mapsto\kappa]}_\omega \varphi[\kappa]$;
    $\models^s_\omega \forall_m m\varphi(m)$ *iff for all* $\mu \in \mathcal{M}$, $\models^{s[m\mapsto\mu]}_\omega \varphi[\mu]$.
13. $\models^s_\omega B_{i,\tau}\varphi$ *iff for all* $\omega' \in W$, *if* $\omega R^{\overline{s}(\tau)}_{\overline{s}(i)} \omega'$ *then* $\models^s_{\omega'} \varphi$.

3.3 Soundness

When a formula φ keeps true under all assignments and all possible worlds in model $\mathfrak{M}$, we say that φ is valid in $\mathfrak{M}$, written as $\models_{\mathfrak{M}} \varphi$, and $\models \varphi$ for short. $\Gamma \models \varphi$ means that when Γ evaluates to be true, φ evaluates to be true as well.

Theorem 1 (Soundness). *For any formula φ, and formula set Γ, we have*

1. *if* $\vdash \varphi$, *then* $\models \varphi$;
2. *if* $\Gamma \vdash \varphi$, *then* $\Gamma \models \varphi$.

Before giving the proof, we need some lemmas.

Lemma 1. *For any $\omega \in W$, $\tau_1, \tau_2 \in \mathcal{T}$, if $\tau_1 < \tau_2$, then*

$$M^{\tau_1}_i(\omega) \subseteq M^{\tau_2}_i(\omega) \tag{1}$$

$$K^{\tau_1}_i(\omega) \subseteq K^{\tau_2}_i(\omega). \tag{2}$$

Proof. From the state transition function in Definition 10 3), we can get this lemma directly.

Lemma 2. *For any set of messages M and set of keys K,*

$$\mathrm{extract}_K(M) \subseteq \mathrm{reconstruct}_K(M).$$

Proof. For any $m \in \text{extract}_K(M)$, we have $m \in \text{construct}_{K \cup key(extract_K(M))}(extract_K(M))$ from Definition 7 1); then from Definition 8 2), we have $m \in \text{reconstruct}_K(M)$. Therefore, $\text{extract}_K(M) \subseteq \text{reconstruct}_K(M)$.

Lemma 3. *For any atomic message m, a set of messages M and a set of keys K, if $m \in \text{reconstruct}_K(M)$, then $m \in \text{extract}(M)$.*

Proof. Since m is atomic, and $m \in \text{reconstruct}_K(M)$, from Definition 8 and Definition 6, the only case is $m \in \text{extract}(M)$.

Lemma 4. *For any message set M, M' and key set K, K', if $M' \subseteq M$ and $K' \subseteq K$, then*

$$\text{extract}_{K'}(M') \subseteq \text{extract}_K(M) \tag{3}$$

$$\text{reconstruct}_{K'}(M') \subseteq \text{reconstruct}_K(M) \tag{4}$$

Proof. A strict proof can be given by mathematical induction. We just give its main idea. Suppose $m \in \text{extract}_{K'}(M')$. By Definition 6, we know that m can be gained by (several times of) transformations (separation, or decryption). Let the number of times of transformations be n. We can use mathematical induction on n according to Definition 6 to get the first conclusion. The second one can be obtained with a similar approach using Definition 6, Definition 7 and Definition 8.

From Lemma 1 and Lemma 4, we can get the following corollary:

Corollary 1. *For any $\omega \in W, \tau_1, \tau_2 \in \mathcal{T}$, if $\tau_1 < \tau_2$, then*

$$\text{extract}_{K_i^{\tau_1}(\omega)}(M_i^{\tau_1}(\omega)) \subseteq \text{extract}_{K_i^{\tau_2}(\omega)}(M_i^{\tau_2}(\omega)) \tag{5}$$

$$\text{reconstruct}_{K_i^{\tau_1}(\omega)}(M_i^{\tau_1}(\omega)) \subseteq \text{reconstruct}_{K_i^{\tau_2}(\omega)}(M_i^{\tau_2}(\omega)) \tag{6}$$

Lemma 5. *For any $\omega \in W$, $i \in \mathcal{A}, \tau, \tau' \in \mathcal{T}$ with $\tau < \tau'$, then $\{\omega' \mid \omega R_i^{\tau'} \omega'\} \subseteq \{\omega'' \mid \omega R_i^{\tau} \omega''\}$.*

Proof. Suppose ω_1 satisfies $\omega R_i^{\tau'} \omega_1$ and by Definition 13. We have $\omega_i^{\tau'} = {\omega_1}_i^{\tau'}$, i.e. $(q_i^{\tau_0}(\omega), \sigma_i^{\tau'}(\omega)) = (q_i^{\tau_0}(\omega_1), \sigma_i^{\tau'}(\omega_1))$, and thus $\sigma_i^{\tau'}(\omega) = \sigma_i^{\tau'}(\omega_1)$. Considering that $\tau < \tau'$, by Definition 10 2) we have

$$\langle \alpha_i^{\tau_0}(\omega) \cdots \alpha_i^{\tau}(\omega) \cdots \alpha_i^{\tau'}(\omega) \rangle = \langle \alpha_i^{\tau_0}(\omega_1) \cdots \alpha_i^{\tau}(\omega_1) \cdots \alpha_i^{\tau'}(\omega_1) \rangle$$

which implies

$$\langle \alpha_i^{\tau_0}(\omega) \cdots \alpha_i^{\tau}(\omega) \rangle = \langle \alpha_i^{\tau_0}(\omega_1) \cdots \alpha_i^{\tau}(\omega_1) \rangle.$$

Specifically, $\sigma_i^{\tau}(\omega) = \sigma_i^{\tau}(\omega_1)$, and thus

$$(q_i^{\tau_0}(\omega), \sigma_i^{\tau}(\omega)) = (q_i^{\tau_0}(\omega_1), \sigma_i^{\tau}(\omega_1))$$

namely, $\omega_i^{\tau} = {\omega_1}_i^{\tau}$. From Definition 13, we have $\omega R_i^{\tau} \omega_1$. As desired.

Lemma 6. *Given $\omega \in W, \tau \in \mathcal{T}, i \in \mathcal{A}$, if agent i firstly sent message $[\![m]\!]_k$ at time τ, then $m \in \text{reconstruct}_{K_i^\tau}(\omega) M_i^\tau(\omega)$ and $k \in K_i^\tau(\omega)$.*

Proof. Suppose the contradictory that $m \notin \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$ or $k \notin K_i^\tau(\omega)$. Since i has sent $[\![m]\!]_k$, Definition 12, 2) tells us $[\![m]\!]_k \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$. From Definition 8 and Definition 10, also considering that $[\![m]\!]_k$ is not atomic, we know that agent i must have received the message $[\![m]\!]_k$ previously. This is in contradiction with the fact that agent i firstly sent the message $[\![m]\!]_k$ at time τ.

Lemma 7. *For any $\omega \in W$, if $\text{receive}([\![m]\!]_{k_j^-}) \in \sigma_i^\tau(\omega)$, then, there exists $\tau' < \tau$, so that agent j firstly sent message $[\![m]\!]_{k_j^-}$ at time τ'.*

Proof. Assume that $[\![m]\!]_{k_j^-}$ is not firstly sent by agent j, then, there must exist another agent who firstly sent it. Assume that i' firstly sent $[\![m]\!]_{k_j^-}$ at time τ', and $i' \neq_a j$. From Lemma 6, we have $k_j^- \in K_{i'}^{\tau'}(\omega)$. That is, i knows k_j^-. By Definition 5, 3), we have $i' =_a j$. This is in contradiction with our assumption, so $[\![m]\!]_{k_j^-}$ is firstly sent by agent j. Accordingly, by the definition of firstly sending in Definition 10 2), we have $\tau' < \tau$.

Lemma 8. *For any $\omega \in W$, if $\text{receive}([\![m]\!]_{k_{i,j}}) \in \sigma_{i'}^\tau(\omega)$, then there must exist time $\tau' < \tau$ so that agent i or j firstly sent the message $[\![m]\!]_{k_{ij}}$ at time τ'.*

Proof. It can be proved using an approach similar to that of Lemma 7.

Lemma 9. *For any $\omega \in W, i \in \mathcal{A}, \tau \in \mathcal{T}$, we have the following conclusions about $\text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$:*

1. *If m is an atomic message, then $m \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$ if and only if one of the following conditions holds:*
   (a) $m \in \text{reconstruct}_{K_i^{\tau_0}(\omega)} M_i^{\tau_0}(\omega)$;
   (b) $\text{new}(m) \in \sigma_i^\tau(\omega)$;
   (c) There exists m' so that $\text{receive}(m') \in \sigma_i^\tau(\omega)$ and $m \in \text{extract}_{K_i^\tau(\omega)} M_i^\tau(\omega)$.
2. *$[m]_k \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$ if and only if one of the following conditions holds:*
   (a) There exist message m' so that $\text{receive}(m') \in \sigma_i^\tau(\omega)$ and $[m]_k \in \text{extract}_{K_i^\tau(\omega)}(m')$;
   (b) $m, k \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$.
3. *$[m, m'] \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$ if and only if $m, m' \in \text{reconstruct}_{K_i^\tau(\omega)} M_i^\tau(\omega)$.*

Proof. The above conclusion can be drawn from Definition 6, 7,8, 10 3) and Definition 12. We omit the details here.

Lemma 10. *All the axioms in $\mathcal{L}$ are valid.*

Proof. Let us prove the validity of all the axioms in $\mathcal{L}$ one by one.

Given model $\mathfrak{M}$, for any possible world $\omega \in W$, any assignment s, to prove $\models \varphi$, we only need to show $\models_{\mathfrak{M},\omega}^s \varphi (\models_\omega^s \varphi$ for short).

*A*1 Validness of axiom *A*1 can be got by the soundness of predicate logic.

*A*2 Because the domain of our model is fixed, we can get the validity of *A*1 from the theory of predicate modal logic.

*A*3 1) Suppose $\models^s_\omega H(i,\tau,m) \wedge (\tau < \tau')$. From Definition 16, we know that $\overline{s} \in \text{reconstruct}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}}(M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega))$, and $\overline{s}(\tau) < \overline{s}(\tau')$. By Corollary 1 2), we have $\overline{s}(m) \in \text{reconstruct}_{K^{\overline{s}(\tau')}_{\overline{s}(i)}(\omega)}(M^{\overline{s}(\tau')}_{\overline{s}(i)}(\omega))$. So, from Definition 16 5), $\models^s_\omega$ $H(i,\tau',m)$ holds.
2) Suppose $\models^s_\omega G(i,\tau,m,m') \wedge (\tau < \tau')$. From Definition 16, we have $\overline{s}(m') \in \text{extract}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}}(\overline{s}(m))$ and $\overline{s}(\tau) < \overline{s}(\tau')$. By Corollary 1, 1), we can get it.
3) Suppose $\models^s_\omega B_{i,\tau}\varphi \wedge (\tau < \tau')$ From Definition 16, for any $\omega' \in W$, if $\omega R^{\overline{s}(\tau)}_{\overline{s}(i)}\omega'$, then $\models^s_{\omega'} \varphi$. That is, for any possible world ω' in set $\{\omega' \mid \omega R^{\overline{s}(\tau)}_{\overline{s}(i)}\omega'\}$, we have $\models^s_{\omega'} \varphi$. Since we have $\overline{s}(\tau) < \overline{s}(\tau')$, with Lemma 5, we know that $\{\omega'' \mid \omega R^{\overline{s}(\tau')}_{\overline{s}(i)}\omega''\} \subseteq \{\omega' \mid \omega R^{\overline{s}(\tau)}_{\overline{s}(i)}\omega'\}$. So, for all ω'' satisfying $\omega R^{\overline{s}(\tau')}_{\overline{s}(i)}\omega''\}$, we also have $\models^s_{\omega''} \varphi$. This means that $\models^s_\omega B_{i,\tau'}\varphi$ by Definition 16 13).

*A*4 From Definition 5 2), we know that $<$ is an ordering relation on $\mathcal{T}$, so, it is transitive.

*A*5 The validity of axioms *A*5 is obtained by Definition 5 3).

*A*6 *A*6 is obtained by Definition 6, Definition 8, Definition 16,4),5) and Lemma 3.

*A*7 1) $\models^s_\omega G(i,\tau,m,m')$. From Definition 16,4), we have $\overline{s}(m') \in \text{extract}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega)}(\overline{s}(m))$. Then from Definition 9,1), $\overline{s}(m') \in \text{contain}(\overline{s}(m))$. By Definition 16,3), we have $\models^s_\omega C(m,m')$.
2) can be proved by Definition 6 7, 8, 9 and Definition 16 3).

*A*8 1) The validity of this axiom is from Definition 10 2).
2)Suppose $\models^s_\omega R(i,\tau,[\![m]\!]_{k_j^-})$. From Definition 16 8) and Lemma 7, we know that agent $\overline{s}(j)$ firstly sent message $\overline{s}([\![m]\!]_{k_j^-})$ before time $\overline{s}(\tau)$. Then, by the definition of firstly sending in Definition 10 2), we have

$$\models^s_\omega \exists_\tau \tau' \exists_m m''((\tau' < \tau) \wedge S(j,\tau',[\![m]\!]_{k_j^-})).$$

Also, by Lemma 6, we have $\overline{s}(m) \in \text{reconstruct}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}}(M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega))$. i.e. $\models^s_\omega$ $H(i,\tau,m)$.
2) We just provide a sketch of the proof. When $\overline{s}(i)$ sends $\overline{s}([\![m']\!]_{k_j^+}))$ with $\overline{s}(m')$ containing a newly generated message $\overline{s}(m)$, if there is an agent receiving this message, the only receiver must be $\overline{s}(j)$, because only $\overline{s}(j)$ has the decrypted key. This can be assured by the fact that $\overline{s}(i)$ receives $\overline{s}(m'')$ at $\overline{s}(\tau)$, from which $\overline{s}(i)$ can get $\overline{s}(m)$. Then, $\overline{s}(j)$ must have received and then sent a message containing $\overline{s}(m)$. Since $\overline{s}(i)$ newly generates $\overline{s}(m)$ at $\overline{s}(\tau)$ and receives message containing $\overline{s}(m)$ at $\overline{s}(\tau')$, so the only possible time for $\overline{s}(j)$ to receive and send such a message is between $\overline{s}(\tau)$ and $\overline{s}(\tau')$.

3) and 4) can be proved in a similar way with the help of Lemma 8 and Lemma 9.

$A9$ can be proved by definitions as follows:
1) by Definition 16,5),6),7),8), and Definition 10 3);
2) by Definition 16,4),5), and Lemma 2;
3) and 4) by Definition 16,5) and Definition 7.

$A10$ 1) is the K axiom in modal logic and its validity can be obtained by the Definition 16,13) and the corresponding conclusion in modal logic.
Other proofs of axioms A10 can proceed in a similar way. So, we just give the proof of 4)and 7).
4) Suppose $\models^s_\omega S(i,\tau,m)$. We have send$(m) \in \alpha^{\overline{s}(\tau)}_{\overline{s}(i)}$. Let ω' be any possible world so that $\omega R^{\overline{s}(\tau)}_{\overline{s}(i)} \omega'$, then, we can get $\omega^{\overline{s}(\tau)}_{\overline{s}(i)} = \omega'^{\overline{s}(\tau)}_{\overline{s}(i)}$, i.e.

$$(q^{\overline{s}(\tau_0)}_{\overline{s}(i)}(\omega), \sigma^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega)) = (q^{\overline{s}(\tau_0)}_{\overline{s}(i)}(\omega'), \sigma^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')).$$

Accordingly, $\sigma^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega) = \sigma^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')$, thus $\alpha^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega) = \alpha^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')$. Since send$(m) \in \alpha^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega)$, hence send$(m) \in \alpha^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')$, i.e. $\models^s_{\omega'} S(i,\tau,m)$. From Definition 16,13), we get the conclusion $\models^s_\omega B_{i,\tau} S(i,\tau,m)$.
7) Suppose $\models^s_\omega H(i,\tau,m) \wedge H(i,\tau,k)$. From Definition 16,5), we get

$$\overline{s}(m), \overline{s}(k) \in \text{reconstruct}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega)}(M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega))\,.$$

Together with Definition 8, we know that there exists a key set $\{\overline{s}(k)\}$, and a message $\overline{s}(m)$ such that $[\overline{s}(m)]_{\overline{s}(k)} \in \text{reconstruct}_{\overline{s}(k)}(\overline{s}(m))$. According to Definition 9,2), we have $\overline{s}(m) \in \text{contain}([\overline{s}(m)]_{\overline{s}(k)})$, i.e. $\models^s_\omega C([m]_k, m)$. For any $\omega' \in W$, if $\omega R^{\overline{s}(\tau)}_{\overline{s}(i)} \omega'$, then, by Definition 13, we have $\omega^{\overline{s}(\tau)}_{\overline{s}(i)} = \omega'^{\overline{s}(\tau)}_{\overline{s}(i)}$. Thus, $M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega) = M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')$, and $K^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega) =^{\overline{s}(\tau)}_{\overline{s}(i)} (\omega')$. Therefore, $\overline{s}(m), \overline{s}(k) \in \text{reconstruct}_{K^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega')}(M^{\overline{s}(\tau)}_{\overline{s}(i)}(\omega'))$. As the above, we obtain $\models^s_{\omega'} C([m]_k, m)$. Thus, from Definition 16,13), $\models^s_{\omega'} B_{i,\tau} C([m]_k, m)$.

Lemma 11. *If the premise of the inference rule is valid, then the conclusion will keep valid.*

Proof. We will prove two inference rules **IR** 1), 2), respectively.

1) Suppose $\models^s_\omega (\varphi \to \psi)$. From Definition 16,11), we have $\not\models^s_\omega \varphi$ or $\models^s_\omega \psi$. By $\models^s_\omega \varphi$, hence $\models^s_\omega \psi$.
2) Since $\vdash \varphi$ shows that φ is a theorem, there must exist a proof sequence about it. We will use mathematical induction over size n of such a sequence.
Base case: If $n = 1$, φ must be an axiom. From Lemma 10, we have $\models \varphi$, namely, in model $\mathfrak{M}$, for any possible world $\omega \in W$ and assignment s, we have $\models^s_\omega \varphi$. Thus, for any $i \in \mathcal{A}, \tau \in \mathcal{T}$, and any possible world ω' which satisfies $\omega R^\tau_i \omega'$, we have $\models^s_\omega \varphi$. From Definition 16,13), $\models^s_\omega B_{i,\tau}\varphi$ holds.

Induction step: Assume $\models^s_\omega B_{i,\tau}\varphi$ when $n \leq k$, we will prove that when $n = k{+}1$, we also have $\models^s_\omega B_{i,\tau}\varphi$. If φ is an axiom, then, from the proof of the basic case, we have $\models^s_\omega B_{i,\tau}\varphi$. If φ is deduced from $\psi, \psi \to \varphi$ via **IR** 1), then, from the assumption of induction, we have $\models^s_\omega B_{i,\tau}\psi$ and $\models^s_\omega B_{i,\tau}(\psi \to \varphi)$. Thus, with the validity of $A10, 1)$, we get $\models^s_\omega B_{i,\tau}\varphi$.

Theorem 1 (Soundness): For any formula φ

- If $\vdash \varphi$, then $\models \varphi$.
- If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.

Proof. Since 1) is a special case of 2), we only give the proof of 2).

If $\Gamma \vdash \varphi$, there must exist a sequence of formulas $\varphi_1\varphi_2 \cdots \varphi_n$ such that $\varphi_n = \varphi$ in which either φ_i is an axiom, or $\varphi_i \in \Gamma$, or φ is deduced from previous φ_j's via inference rules. Also, we use mathematical induction over the size n of this sequence.

Base case: If $n = 1$, φ is an axiom or in Γ. From Lemma 10, we see that φ is valid in the former case. Then in the later case, if all the formulas in Γ is valid, it's natural that φ is valid. So, in both cases we can get $\Gamma \models \varphi$.

Induction step: Assume that the proposition holds when $n \leq k$. We show that it holds when $n = k + 1$. If φ is an axiom or in Γ, from the proof of the base case, we have $\Gamma \models \varphi$. If ϕ is deduced from $\varphi_i, \varphi_j (i, j \leq k)$ via inference rules, by the assumption of induction, we have $\Gamma \models \varphi_i, \Gamma \models \varphi_j$. By Lemma 11, we conclude that φ is valid, i.e. $\Gamma \models \varphi$.

4 Modeling

Before analyzing protocols, we must model the protocol and its security goals. In this section, we will give such a modeling scheme.

4.1 Protocol

Usually, a protocol is a sequence of protocol statements. A protocol statement is often in the form of '$i \rightarrowtail j \,:\, \hat{m}$', meaning that agent i sends message $\hat{m}$ to agent j. In our modeling scheme, the original protocol (written as $\hat{P}$) is modeled as a set of formal formulas, which is called a formalized protocol, written as P.

Since some information in $\hat{P}$ is often expressed implicitly, to model $\hat{P}$ accurately, we need to make these information explicit.

Implicit information Usually, given a protocol $\hat{P}$, there are three types of information being implicitly expressed.

The first type of information is the identity of agent i in message $[m]_{k_i^-}$ is usually implicitly expressed. Recall that a message in private-key-encrypted form reveal its original sender. To make the identity information in a message explicit, in our scheme, such message m in form of $[m']_{k_i^-}$ will be replaced by $[i, m']_{k_i^-}$.

The second type is the running order of the protocol. In general, the running order of a protocol is implicitly expressed by the order in which all the protocol statements in it are arranged. Fortunately, logic $\mathcal{L}$ has an advantage to express these orders explicitly. Firstly, we need to attach two time labels on each protocol statement, and then, use some formulas to express time order according to the running order implied by $\hat{P}$.

The last type is that message newly generated is expressed implicitly in $\hat{P}$. Certainly, we can assume some messages are generated before running protocol and thus are initially held by some agents. But this assumption does not apply to a fresh value such as a nonce, a random number or a timestamp. So, when a fresh value first appears in '$i \rightarrowtail j \,:\, \hat{m}$' of $\hat{P}$, we can make it explicit that agent i newly generates this value at the time he sends $\hat{m}$.

Protocol Formalization In our modeling scheme, we will model a protocol $\hat{P}$ by a set of formulas, writhen as P.

Usually, a protocol $\hat{P}$ defines the actions of every agent participating in the protocol. This gives us a global view of $\hat{P}$. However, in each agent's view, what he can confirm is what actions(receiving, sending, and generating messages) he does himself. Therefore, we first model protocol by each agent's local view which is expressed by a subset of P. Formally, a local view of agent i about protocol $\hat{P}$ is written as P_i.

Let $\hat{P}$ be the original protocol including n protocol statements, $\hat{P}_r$ be the r-th ($1 \leq r \leq n$) statement in $\hat{P}$ with the form of '$i' \rightarrowtail j' \,:\, \hat{m}_r$', i be any agent of $\hat{P}$. The following procedure is used to formalize the protocol by constructing the local view of i about $\hat{P}$, say P_i, which is initialized by an empty set:

1. *Message transformation.* For each $\hat{m}_r$ of $\hat{P}_r$, if there exists a message m' and a key k_i^- such that $[m']_{k_i^-}$ is contained in $\hat{m}_r$, then transform $\hat{m}_r$ into m_r by
$$m_r = \begin{cases} \hat{m}_r\left([m', i]_{k_i^-}/[m']_{k_i^-}\right) & \text{if } \neg C(m', i) \\ \hat{m}_r & \text{if } C(m', i) \end{cases}$$
in which $m(y/x)$ means replacing all x contained in m with y.
2. *Time association.* Associate two time labels on each $\hat{P}_r$, and get a timed protocol statement in the form of
$$\tau_r(i' \rightarrowtail j' \,:\, m_r)\tau'_r$$
3. *Message generation.* For each timed protocol statement '$\tau_r(i' \rightarrowtail j' \,:\, m_r)\tau'_r$', if $i =_a i'$, and there is a newly generated message m contained in m_r, then[3],
$$P_i := P_i \cup \{\nu(i, \tau_r, m)\}.$$
4. *Statement transformation.* For each timed protocol statement '$\tau_r(i' \rightarrowtail j' \,:\, m_r)\tau'_r$'[4],
$$\text{if } i =_a i',\ \text{then, } P_i := P_i \cup \{S(i, \tau_r, m_r)\};$$

[3] Here, we use ':=' to denote evaluating, and thus, '$P_i := P_i \cup \varphi$' means that, P_i is evaluated by adding an additional formula φ.

[4] When the timed protocol statement is in form of 'τ_r(if φ, $i \rightarrowtail j \,:\, m_r)\tau'_r$', we also need to add φ to P_i.

$$\text{if } i =_a j', \text{ then, } P_i := P_i \cup \{R(i, \tau_r', m_r)\}.$$

5. *Time ordering.* Let the number of i's actions in the protocol be ℓ, and the x'th$(1 \leq x \leq \ell)$ action of agent i is associated with time τ_{i_x}, then, add the formula about time ordering into P_i:

$$P_i := P_i \cup \{\tau_0 < \tau_{i_1} < \tau_{i_2} < \cdots < \tau_{i_\ell} < \tau_c\}.$$

Let us give further explanation about this procedure.

– Step 1 is used to explicitly express the identity message in a private-key-encrypted message. An immediate question is that, whether $[m]_{k_j^+}$ can be replaced by $[m, j]_{k_j^+}$. The answer is *yes*, but doing so is meaningless, because a private key can keep the integrity of a message while a public key cannot.
– Step 2 is used to explicitly express the time factor. In '$\tau_r(i' \rightarrowtail j' : m_r)\tau_r'$', τ_r indicates the time of i''s sending action and τ_r' indicates the time of j''s receiving action. Moreover, recall that τ_0 and τ_c are the beginning time and end time of a protocol. Thus, if $\tau_r(i' \rightarrowtail j' : m_r)\tau_r'$ is the first protocol statement, we have $\tau_r =_\tau \tau_0$, and if $\tau_r(i' \rightarrowtail j' : m_r)\tau_r'$ is the last protocol statement, we have $\tau_r' =_\tau \tau_c$.
– In step 3, we use predicate ν to capture freshness of a message, like a nonce, a random number, a timestamp etc. Obviously, freshness is time-dependent, so, the predicate ν can naturally be used to model it. Note that some other messages which are not fresh are often considered as the initial knowledge of an agent.
– In step 4, the sending or receiving action is modeled and the time when such action takes place is specified.
– Step 5 explicitly expresses the time ordering in locally running the protocol by an agent i.

Given the local view P_i of agent i about protocol $\hat{P}$, we can get the global view $P := \bigcup_i \{P_i\}$

Unlike the idealization process of BAN logic, the protocol formalization in our method is unambiguous. Although we provide 5 steps in the procedure, it is in fact very succinct and, given $\hat{P}$, one can even write down each P_i and P directly.

Initial Assumptions. Before modeling goals of the protocol, we need to talk about initial assumptions. Initial assumptions can provide a more idealized environment, but often can also make things unpractical. So, we only give as few assumptions as possible. The following form of assumptions is often used in our reasoning:

1. $H(i, \tau, k_j^+) \wedge B_{i',\tau'} H(i, \tau, k_j^+)$
2. $(x \neq y) \wedge B_{i,\tau}(x \neq y)$
3. $\neg C(m, m')$

The first assumption is about the public key. That is, public keys are well-known. The second one makes sense when some terms are obviously unequal. For example, an atomic message is assumed never equal to an encrypted one; a protocol with identical initiator and responder is also considered meaningless. The third one is often used in proof of a secrecy goal. If a message m' is obviously not contained in message m, no one can get m' from m. To avoid such vain attempt, we can explicitly announce it. Other initial assumptions may depend on a specific protocol. The initial assumption is expressed by a set of formulas, written as S.

4.2 Security goals

Security goals is central to the analysis of security protocols. In this subsection, we will show how to model authentication, secrecy as well as a time-dependent property.

Authentication Intuitively, authentication in a protocol between two agents means that an agent believes another agent is just who he claimed to be. There are three implications about this intuition:

- Each agent has claimed his identity to the other agent;
- Each agent needs to provide the proof of his own identity;
- Each agent believes the other agent's identity.

Such things can also be modeled in $\mathcal{L}$. Given a formal protocol P which is intended to complete the authentication between agent i and j, authenticating j by i at time τ can be modeled as one of the following expressions:

$$P_i \cup S \vdash B_{i,\tau} \exists_\tau \tau' \big(\tau_0 \leq \tau' \wedge R(j, \tau', [\![i]\!]) \wedge H(j, \tau', i)\big)$$
$$P_i \cup S \vdash B_{i,\tau} \exists_\tau \tau' \big(\tau_0 \leq \tau' \wedge S(j, \tau', [\![i]\!]) \wedge H(j, \tau', i)\big).$$

Note that the appearance of i at a message field just indicates an identity of agent i. In the former expression, "$\tau_0 \leq \tau' \wedge R(j, \tau', [\![i]\!])$" assures that the message containing i's identity received by j is a new message; "$H(j, \tau', i)$" assures that j holds i's identity in current run. The most important thing is that, all these must be believed by i in i's local view. Someone may say that only "$B_{i,\tau} \exists_\tau \tau' \big(\tau_0 \leq \tau' \wedge R(j, \tau', [\![i]\!])$" can model the authentication of j by i. Consider that a received message may be in an encrypted form and thus the receiver may not get the identity information from it. Still, one may say that only "$B_{i,\tau} \exists_\tau \tau' \big(\tau_0 \leq \tau' \wedge H(j, \tau', i)\big)$" can model the authentication of j by i. Consider that j received i's identity before τ_0, j will also hold i's identity at τ', but nothing can be authenticated from such 'holding'.

In this modeling, when i believes someone has recently received the message containing his identity and held this identity, the identity of i must have been claimed to j, and the proof to show j's identity must have been provided to i. Therefore, all the three aspects of the intuition of authentication above are captured.

We provide two expressions of authentication. In general, the former is suitable for a protocol using the public key, and the latter is suitable for a protocol using the private key. Both can be used in a protocol with the shared key.

Secrecy Intuitively, secrecy of a protocol means to keep a specific message in this protocol secret. That is, after running a protocol, the secret message can never be got from the messages transmitted in the protocol by the agents other than its intended holders. This intuition can be easily modeled in $\mathcal{L}$. Given a formal protocol P and its initial assumptions S, let m be that needs to be kept secret, agents i and j be the intended holders of m, τ be the specific time before which m needs to be kept secret, M_{τ_c} be all the messages transmitted in $\hat{P}$. Then, the secrecy property can be modeled as follows:

$$P \cup S \vdash i' \neq_a i \wedge i' \neq_a j \rightarrow \neg\, G(i', \tau_c, M_{\tau_c}, m)$$
$$P \cup S \vdash H(i, \tau_c, m) \wedge H(j, \tau_c, m)$$

Additionally, for compatibility of message types, we define M_{τ_r} as a combined message instead of a message set. More formally, using the notation in protocol formalization, we have

$$M_{\tau_r} \triangleq [m_1, m_2, \ldots, m_r].$$

Note that, in modeling secrecy, we use a global view P instead of a local view P_i. The reason is that, after showing the authentication of a protocol, an agent can confirm the identity of the agents who communicate with him. In another words, the authentication extend the local view to a global view. Without authentication, such secrecy is in a passive setting.

Time-dependent properties Sometimes we also need a time-dependent property. A time-dependent authentication may need an agent to believe something at a specific time other than the time after running a protocol. A time-dependent secrecy may need some message to be kept secret before a specific time other than during the whole protocol. To express such properties, we only need to replace the τ_c in corresponding goals mentioned above by a specific time τ.

Time-dependent properties also have other advantages. Sometimes, we need to know what will happen when the protocol is stopped in the middle of its run. For example, in a fair exchange protocol, an agent may take advantage of a partially executed protocol. This situation can be easily modeled by using a time-dependent property. Specifically, if we need to know whether an agent i holds a message m when a running protocol stops at time τ, we just need to check whether $H(i, \tau, m)$ holds.

5 Reasoning

In general, reasoning about a protocol $\hat{P}$ requires the following steps:

1. formalize Protocol (P);
2. give the initial assumptions(S);

3. list the protocol goals(G);
4. prove the protocol goals G in the proposed logic.

In the following, we will give two examples to illustrate how to reason about security protocols under our modeling scheme using the proposed logic.

5.1 Timed-release protocol

This protocol is proposed by Michiharu Kudo and Anish Mathuria [19]. The protocol aims to provide some time-dependent security properties using the notion of timed-release crypto described in Rivest, Shamir and Wagner [14]. In this protocol, A wants to send a confidential message X_a(in encrypted form) to B, but hopes B to hold X_a only after a specific time τ_s. To complete this goal, a trusted agent T is used. Firstly, A send τ_s to T, requiring T to generate a time-key pair for the future time τ_s. After generating a time-key pair k_{τ_s+} and k_{τ_s-}, T will reply a signed message containing the encryption key k_{τ_s+} to A, and holds the decryption key k_{τ_s-} until time τ_s. Then, after initializing a communication with B, A encrypts X_a with the encryption key k_{τ_s+}, and sends it to B. Upon receiving the encrypted message, B acknowledges the receipt of it, and then requests the decryption key k_{τ_s-} from T. If the current time is greater or equal to τ_s, T will return the decryption key to B; otherwise he does not respond. The detailed protocol is described in fig.1, in which R_a and N_b are used to denote the random number and the nonce generated by A and B respectively.

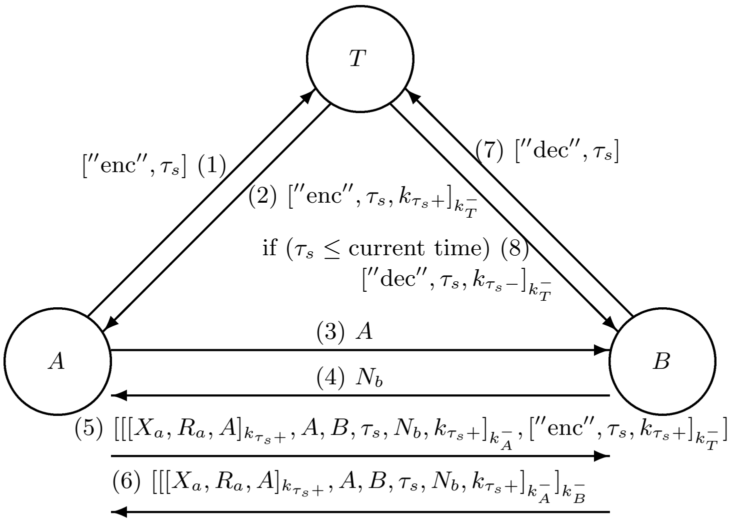
T
A
B
$[''\text{enc}'', \tau_s]$ (1)
(2) $[''\text{enc}'', \tau_s, k_{\tau_s+}]_{k_T^-}$
(7) $[''\text{dec}'', \tau_s]$
if ($\tau_s \leq$ current time) (8)
$[''\text{dec}'', \tau_s, k_{\tau_s-}]_{k_T^-}$
(3) A
(4) N_b
(5) $[[[X_a, R_a, A]_{k_{\tau_s+}}, A, B, \tau_s, N_b, k_{\tau_s+}]_{k_A^-}, [''\text{enc}'', \tau_s, k_{\tau_s+}]_{k_T^-}]$
(6) $[[[X_a, R_a, A]_{k_{\tau_s+}}, A, B, \tau_s, N_b, k_{\tau_s+}]_{k_A^-}]_{k_B^-}$

Fig. 1. Timed-Release Protocol

In [19], the authors give three goals of the protocol. Informally, the first goal says that any agent except A and T cannot hold X_a before time τ_s. The second goal says that B holds X_a after τ_s. The third goal says that B can authenticate A. In fact, we find that another goal must be specified. That is, A can also authenticate B. Because if A cannot authenticate B, she may send the message containing X_a to an agent other than B and thus B cannot hold X_a. We then show the reasoning process about timed-release protocol. **Protocol formalization** According to the fist two steps in formalizing the protocol, we get the protocol as follows:

τ_0 $(A \rightarrowtail T : [''\text{enc}'', \tau_s])$ τ_1

τ_1 $(T \rightarrowtail A : [''\text{enc}'', T, \tau_s, k_{\tau_s}+]_{k_T^-})$ τ_2

τ_2 $(A \rightarrowtail B : A)$ τ_3

τ_3 $(B \rightarrowtail A : N_b)$ τ_4

τ_4 $(A \rightarrowtail B : [[[X_a, R_a, A]_{k_{\tau_s+}}, A, B, \tau_s, N_b, k_{\tau_s}+]_{k_A^-}, [''\text{enc}'', T, \tau_s, k_{\tau_s}+]_{k_T^-}])$ τ_5

τ_5 $(B \rightarrowtail A : [B, [[X_a, R_a, A]_{k_{\tau_s+}}, A, B, \tau_s, N_b, k_{\tau_s}+]_{k_A^-}]_{k_B^-})$ τ_6

τ_7 $(B \rightarrowtail T : [''\text{dec}'', \tau_s])$ τ_8

τ_9 (if ($\tau_s \leq$ current time), $T \rightarrowtail B : [''\text{dec}'', T, \tau_s, k_{\tau_s}-]_{k_T^-}$) τ_c

Here, for simplification, we ignored message processing time when an agent sends a message directly after when the same agent receives a message. the same agent's receiving a message. So, we distinguish τ_6 and τ_7 simply because they specify the time to the actions of different agents. Note that the duration between τ_8 and τ_9 cannot be ignored, because T sends the message at 8-th step only when *current time* is greater than or equal to τ_s.

According to the protocol formalizing scheme, we can get P_A, P_B and P_T which denote the local views of agent A, B and T respectively as following:

P_A :	P_B :	P_T :
$\{\nu(A, \tau_4, R_a)$	$\{\nu(B, \tau_3, N_b)$,	$\{\nu(T, \tau_1, k_{\tau_s+})$,
$S(A, \tau_0, m_1)$,	$R(B, \tau_3, m_3)$,	$\nu(T, \tau_1, k_{\tau_s-})$,
$R(A, \tau_2, m_2)$,	$S(B, \tau_3, m_4)$,	$R(T, \tau_1, m_1)$,
$S(A, \tau_2, m_3)$,	$R(B, \tau_5, m_5)$,	$S(T, \tau_1, m_2)$,
$R(A, \tau_4, m_4)$,	$S(B, \tau_5, m_6)$,	$R(T, \tau_8, m_7)$,
$S(A, \tau_4, m_5)$,	$S(B, \tau_7, m_7)$,	$S(T, \tau_9, m_8)$,
$R(A, \tau_6, m_6)$,	$R(B, \tau_c, m_8)$,	$\tau_0 < \tau_1 < \tau_8 < \tau_9 < \tau_c$,
$\tau_0 < \tau_2 < \tau_4 < \tau_6 < \tau_c\}$	$\tau_0 < \tau_3 < \tau_5 < \tau_7 < \tau_c\}$	$\tau_s \leq \tau_9\}$

Taking all the local views together, we can get the global view of the protocol:

$$P := P_A \cup P_B \cup P_T.$$

Initial assumption(S)

$S_1 : H(i, \tau, k_j^+) \wedge B_{i', \tau'} H(i, \tau, k_j^+)$ $S_2 : \neg C([m_1, m_2, m_3, m_4, m_7, m_8], X_a)$

$S_3 : k_{\tau_s+} =_k \tilde{k}_{\tau_s-}$ $S_4 : (\tau < \tau_9) \rightarrow \neg S(T, \tau, [\![k_{\tau_s-}]\!])$

$S_5 : H(A, \tau_0, [X_a, A])$ $S_6 : A \neq_a B$

Authentication goals

$$G_1 \quad P_B \cup S \vdash B_{B,\tau_c}\exists_\tau \tau(\tau_0 \leq \tau \wedge S(A,\tau,\llbracket B \rrbracket) \wedge H(A,\tau,B))$$
$$G_2 \quad P_A \cup S \vdash B_{A,\tau_c}\exists_\tau \tau(\tau_0 \leq \tau \wedge S(B,\tau,\llbracket A \rrbracket) \wedge H(B,\tau,A))$$

Proof. First, we prove G_1. Let $m' \triangleq [[X_a, R_a, A]_{\tau_s+}, A, B, \tau_s, N_b, k_{\tau_s+}]$. G_1 is proved as follows:

1.	$B_{B,\tau_5}R(B,\tau_5,m_5)$	$P_B, A10\,5)$
2.	$B_{B,\tau_5}G(B,\tau_5,m_5,[m']_{k_A^-})$	$S_1, A6, A10\,2)$
3.	$B_{B,\tau_5}G(B,\tau_5,[m']_{k_A^-},m')$	$S_1, A6, A10\,2)$
4.	$B_{B,\tau_5}R(B,\tau_5,\llbracket m' \rrbracket_{k_A^-})$	$1,2,3, A6\,4), A10\,1)$
5.	$B_{B,\tau_5}\exists_\tau \tau((\tau < \tau_5) \wedge H(A,\tau,m') \wedge S(A,\tau,\llbracket m' \rrbracket))$	$4, A8\,2), IR\,2)$
6.	$B_{B,\tau_5}((\tau' < \tau_5) \wedge H(A,\tau',m') \wedge S(A,\tau,\llbracket m' \rrbracket))$	5, assumption
7.	$B_{B,\tau_5}(G(A,\tau',m',B) \wedge G(A,\tau',m',N_b))$	$A6\, A10\,2)$
8.	$B_{B,\tau_5}(H(A,\tau',B) \wedge H(A,\tau',N_b))$	$6,7, A9\,2), IR\,2), A1$
9.	$B_{B,\tau_5}\nu(B,\tau_3,N_b)$	$P_B, A10\,3), A3\,3)$
10.	$B_{B,\tau_5}(\tau_3 < \tau')$	$9, S_6, 8, A8\,5), A10\,1)$
11.	$B_{B,\tau_5}C(m',B)$	$7, A7\,1), IR\,2)$
12.	$B_{B,\tau_5}\exists_\tau \tau(\tau_3 < \tau \wedge S(A,\tau,\llbracket B \rrbracket) \wedge H(A,\tau,B))$	$6,8,10,11, A10\,1), A1$
13.	$B_{B,\tau_c}\exists_\tau \tau(\tau_0 \leq \tau \wedge S(A,\tau,\llbracket B \rrbracket) \wedge H(A,\tau,B))$	$12, P_B, A4, A3\,3)$.

Let $m'' \triangleq [[X_a, R_a, A]_{\tau_s+}, A, B, \tau_s, N_b, k_{\tau_s+}]_{k_A^-}$. The proof of G_2 proceeds as follows:

1.	$B_{A,\tau_6}R(A,\tau_6,[m'']_{k_B^-})$	$P_A, A10\,5)$
2.	$G(A,\tau_6,[m'']_{k_B^-},m'')$	$S_1, A5\,5), A6$
3.	$B_{A,\tau_6}C([m'']_{k_B^-},m'')$	$2, A10\,2), A7\,1), IR\,2)$
4.	$B_{A,\tau_6}R(A,\tau_6,\llbracket m'' \rrbracket_{k_B^-})$	$1,2,3$, Def. $4\,1)$
5.	$B_{A,\tau_6}\exists_\tau \tau((\tau < \tau_6) \wedge S(B,\tau,\llbracket m'' \rrbracket) \wedge H(B,\tau,m''))$	$4, A8\,2), IR\,2), A10\,1)$
6.	$B_{A,\tau_6}(S(B,\tau',\llbracket m'' \rrbracket) \wedge H(B,\tau',m'')$	5, assumption
7.	$B_{A,\tau_6}(H(B,\tau',k_A^+) \wedge H(A,\tau_6,K_A^+))$	S_1
8.	$B_{A,\tau_6}(G(A,\tau_6,m'',A) \wedge C(m'',A)$ $\wedge C(m'',[X_a,R_a,A]_{k_{\tau_s+}}))$	$7, A6, A10\,2), A7\,1)$
9.	$B_{A,\tau_6}S(B,\tau',\llbracket A \rrbracket) \wedge H(B,\tau',A)$	$6,8, A7\,2)$, Def. $4\,1)$
10.	$H(A,\tau_6,K_{\tau_s+})$	$P_2, A9\,1), S_1, A6, A3\,1)$
11.	$H(A,\tau_6,R_a)$	$P_A, A9\,1), A3\,1)$

12. $H(A, \tau_6, [X_a, R_a, A])$	$11, S_5, A3\,1), A9\,3)$
13. $B_{A,\tau_6} C([X_a, R_a, A]_{k_{\tau_s+}}, [X_a, R_a, A])$	$10, 12, A10\,7)$
14. $B_{A,\tau_6} C([X_a, R_a, A]_{k_{\tau_s+}}, R_a)$	$13, A7, IR\,2)$
15. $B_{A,\tau_6} C(m'', R_a)$	$8, 14, A7\,2), IR\,2)$
16. $B_{A,\tau_6} S(B, \tau', \llbracket R_a \rrbracket)$	$6, 15, \text{Def. } 4\,1)$
17. $B_{A,\tau_6} H(B, \tau', \llbracket R_a \rrbracket)$	$16, A9\,1), IR\,2)$
18. $B_{A,\tau_6} \nu(A, \tau_4, R_a)$	$P_A, A10\,3), A3, 3)$
19. $B_{A,\tau_6} (\tau_4 < \tau')$	$S_6, 17, 18, IR\,2)$
20. $B_{A,\tau_6} \exists_\tau \tau((\tau_4 \leq \tau) \wedge S(B, \tau, \llbracket A \rrbracket) \wedge H(B, \tau, A))$	$9, 19, A1, A10\,1)$
21. $B_{A,\tau_c} \exists_\tau \tau(\tau_0 \leq \tau \wedge S(B, \tau, \llbracket A \rrbracket) \wedge H(B, \tau, A))$	$20, P_A, A4, A3\,3), A10\,1)$

Time-dependent authentication goals It is not hard to see that, in G1 and G2, A can authenticate B, and B can authenticate B only when the protocol completes. However, in this protocol, what we really need is a stronger authentication, i.e., the time-dependent authentication goals. Specifically, the latest time when A needs to authenticate B is the time when she sends the message containing X_a, say τ_4, and the latest time when B needs to authenticate A is the time when he acknowledges a receipt, say τ_5. Such authentication can not be expressed by a logic without time, but can be easily modeled in $\mathcal{L}$ as follows:

$$\begin{array}{ll} G'_1 & P_B \cup S \vdash B_{B,\tau_5} \exists_\tau \tau(\tau_0 < \tau \wedge S(A, \tau, \llbracket B \rrbracket) \wedge H(A, \tau, B)) \\ G'_2 & P_A \cup S \vdash B_{A,\tau_4} \exists_\tau \tau(\tau_0 < \tau \wedge S(B, \tau, \llbracket A \rrbracket) \wedge H(B, \tau, A)) \end{array}$$

Similar to the proof of G_1, we can easily prove that G'_1 holds. However, to prove G'_2, we need a proof to show that, at τ_4, A believes that B can receive message 5. Since at τ_4, A cannot confirm whether B will reply a receipt, G_2 cannot be satisfied. This indicates that, before A can authenticate B, she has sent the message containing X_a to B. What if an adversary C imitates B, and selects not to acknowledge the receipt? The result is that C can also get k_{τ_s-} from T after τ_s. Thus, an attack appears. We illustrate such an attack as follows:

$$\begin{array}{rl} A \rightarrowtail T & : [''\text{enc}'', \tau_s] \\ T \rightarrowtail A & : [''\text{enc}'', \tau_s, k_{\tau_s+}]_{k_T^-} \\ A \rightarrowtail B(C) & : A \\ B(C) \rightarrowtail A & : N_c \\ A \rightarrowtail B(C) & : [[[X_a, R_a, A]_{k_{\tau_s+}}, A, B, \tau_s, N_c, k_{\tau_s+}]_{k_A^-}, [''\text{enc}'', \tau_s, k_{\tau_s+}]_{k_T^-}] \\ B(C) \rightarrowtail T & : [''\text{dec}'', \tau_s] \\ \text{if } (\tau_s \leq \text{ current time}), & \\ T \rightarrowtail B(C) & : [''\text{dec}'', \tau_s, k_{\tau_s-}]_{k_T^-} \end{array}$$

We can see that, the explicit support of time in the proposed logic helps us find the above vulnerability. To our knowledge, this vulnerability has not been discovered before in the literature.

Time-dependent secrecy goals

$$\begin{array}{ll} G_3 & P \cup S \vdash (i \neq_a A) \wedge (i \neq_a T) \rightarrow \neg\, G(i, \tau_s, M_{\tau_s}, X_a) \\ & \text{in which } M_{\tau_s} \triangleq [m_1, m_2, m_3, m_4, m_5, m_6, m_7] \\ G_4 & P \cup S \vdash H(B, \tau_c, X_a) \end{array}$$

Given a global view P, we can easily prove G_3 and G_4. We prove G_3 by contradiction as follows:

1.	$(i \neq_a A) \wedge (i \neq_a T)$	premise
2.	$G(i, \tau_s, M_{\tau_s}, X_a)$	assumption
3.	$\tau_s \leq \tau_9$	$P, A1$
4.	$G(i, \tau_9, M_{\tau_s}, X_a)$	$2, 3, A3\,2)$
5.	$G(i, \tau_9, M_{\tau_9}, M_{\tau_s})$	$3, A6\,3), A6\,1)$
6.	$G(i, \tau_9, M_{\tau_9}, X_a)$	$4, 5, A6\,4)$
7.	$\neg\, G(i, \tau_9, [m_1, m_2, m_3, m_4, m_7, m_8], X_a)$	$S_2, A7\,1), A1$
8.	$G(i, \tau_9, m_5, X_a) \vee G(i, \tau_9, m_6, X_a)$	$6, 7, A6\,3), A1$
9.	$G(i, \tau_9, m_5, X_a)$	$8,$ assumption
10.	$H(i, \tau_9, k_A^+) \wedge H(i, \tau_9, k_{\tau_s -})$	$9, A6, A5\,5), S_3, A1$
11.	$G(i, \tau_9, m_5, X_a) \rightarrow H(i, \tau_9, k_{\tau_s -})$	$9, 10, A1$
12.	$G(i, \tau_9, m_6, X_a)$	$8,$ assumption
13.	$H(i, \tau_9, k_B^+) \wedge H(i, \tau_9, k_A^+) \wedge H(i, \tau_9, k_{\tau_s -})$	$12, A6\,2), A6\,1), A1$
14.	$G(i, \tau_9, m_6, X_a) \rightarrow H(i, \tau_9, k_{\tau_s -})$	$12, 13, A1$
15.	$H(i, \tau_9, k_{\tau_s -})$	$8, 11, 14, A1$
16.	$\exists_\tau \tau((\tau_1 < \tau < \tau_9) \wedge S(T, \tau, \llbracket k_{\tau_s -} \rrbracket)$	$1, P, 15, A8\,5)$
17.	$\bot$(under assumption)	$16, S_4, A1$
18.	$\neg\, G(i, \tau_s, M_{\tau_s}, X_a)$	$2, 17, A1$
19.	$(i \neq_a A) \wedge (i \neq_a T) \rightarrow \neg\, G(i, \tau_s, M_{\tau_s}, X_a)$	$1, 18, A1.$

We now prove $G_4 \,:\, H(B, \tau_c, X_a)$.

1.	$H(B, \tau_5, m_5)$	$P, A9\,1)$
2.	$H(B, \tau_5, k_A^+)$	S_1
3.	$G(B, \tau_5, m_5, [X_a, R_a, A]_{k_{\tau_s +}})$	$2, A6, A5\,5)$
4.	$H(B, \tau_5, [X_a, R_a, A]_{k_{\tau_s +}})$	$1, 3, A9\,2)$
5.	$H(B, \tau_c, m_8)$	$P_8, A9\,1)$
6.	$G(B, \tau_c, m_8, k_{\tau_s -})$	$5, S_3, A6$
7.	$H(B, \tau_c, k_{\tau_s -})$	$5, 6, A9\,2)$
8.	$H(B, \tau_c, [X_a, R_a, A]_{k_{\tau_s +}})$	$4, P, A4, A3\,1)$
9.	$H(B, \tau_c, X_a)$	$7, 8, A6, A9\,3), A9\,2)$

However, since G_2 is not satisfied, a local view cannot be extended to a global view P. So, the premise in G_3 and G_4 is over-used. In fact, when a local view is used, G_3 can also be proved, while G_4 cannot be proved in the local view of A. Intuitively, from the vulnerability mentioned above, we can see that if an adversary C imitates B, B may know nothing about X_a. Moreover, in this situation, G_3 will become a trivial goal. That is, although X_a is kept secret before time τ_s, B may learn nothing about X_a after τ_s.

5.2 Needham-Schroeder (NS) public key protocol

This is a well-known protocol[40] which is often used as an example in reasoning about protocol. We use this example to show two things:

- $\mathcal{L}$ can be used not only to a timed protocol, but also to a general protocol.
- The modeling scheme can accurately capture some security goals which may be neglected by other modeling scheme.

Assume the readers are familiar with NS protocol. Here, as usual, we only use its three core steps. We just give the specification (with time attached) of it without further explanation.

$$\begin{array}{lll}
\tau_0 & (A \rightarrowtail B\ : [N_a, A]_{k_B^+}) & \tau_1 \\
\tau_1 & (B \rightarrowtail A\ : [N_a, N_b]_{k_A^+}) & \tau_2 \\
\tau_2 & (A \rightarrowtail B\ : [N_b]_{k_B^+}) & \tau_c
\end{array}$$

Protocol formalization

$$\begin{array}{ll}
P_A : & P_B : \\
\quad \{\nu(A, \tau_0, N_a), & \quad \{\nu(B, \tau_1, N_b), \\
\quad\ S(A, \tau_0, [N_a, A]_{k_B^+}), & \quad\ R(B, \tau_1, [N_a, A]_{k_B^+}), \\
\quad\ R(A, \tau_2, [N_a, N_b]_{k_A^+}), & \quad\ S(B, \tau_1, [N_a, N_b]_{k_A^+}), \\
\quad\ S(A, \tau_2, [N_b]_{k_B^+}), & \quad\ R(B, \tau_3, [N_b]_{k_B^+}), \\
\quad\ \tau_0 < \tau_2 < \tau_c\} & \quad\ \tau_0 < \tau_1 < \tau_c\}
\end{array}$$

Initial assumption

$$\begin{aligned}
S_1 &: H(i, \tau, k_j^+) \wedge B_{i', \tau'} H(i, \tau, k_j^+) \\
S_2 &: H(i, \tau_0, i)
\end{aligned}$$

Protocol goals

$$\begin{aligned}
&G_1\ P_A \cup S \vdash B_{A,\tau_c} \exists_\tau \tau(\tau_0 \le \tau \wedge R(B, \tau, [\![A]\!]) \wedge H(B, \tau, A)) \\
&G_2\ P_B \cup S \vdash B_{B,\tau_c} \exists_\tau \tau(\tau_0 \le \tau \wedge R(A, \tau, [\![B]\!]) \wedge H(A, \tau, B)).
\end{aligned}$$

Since the public key is used, we express the goals with predicate R instead of S.

Proof. Prove G_1 as follows:

1.	$B_{A,\tau_0} S(A, \tau_0, [N_a, A]_{k_B^+})$	$P_A, A10\,4)$
2.	$H(A, \tau_0, [N_a, A]) \wedge H(A, \tau_0, k_B^+)$	$N_1, A9\,1), S_1, S_2$
3.	$B_{A,\tau_0} C([N_a, A]_{k_B^+}, [N_a, A])$	$2, A10\,7)$
4.	$B_{A,\tau_2} S(A, \tau_0, \llbracket N_a, A \rrbracket_{k_B^+})$	$1, 3, \text{Definition}\,4\,1), A3\,3)$
5.	$B_{A,\tau_2} C([N_a, A], N_a)$	$A6, A10\,2), A7\,1), IR\,2)$
6.	$B_{A,\tau_2} R(A, \tau_2, [N_a, N_b]_{k_A^+})$	$P_A, A10\,5)$
7.	$B_{A,\tau_2} G(A, \tau_2, [N_a, N_b]_{k_A^+}, N_a)$	$S_1, A6, A10\,2)$
8.	$B_{A,\tau_2} \nu(A, \tau_0, N_a)$	$N_1, A10\,3), A3\,3)$
9.	$B_{A,\tau_2} \exists_\tau \tau(\tau_0 \leq \tau < \tau_2 \wedge R(B, \tau, \llbracket N_a, A \rrbracket_{k_B^+}))$	$4, 5, 6, 7, 8, A8\,3), IR\,2)$
10.	$B_{A,\tau_2} \exists_\tau \tau(\tau_0 \leq \tau < \tau_2 \wedge R(B, \tau, \llbracket A \rrbracket_{k_B^+}) \wedge H(B, \tau, A))$	$9, S_1, A9, A6, A7, IR\,2)$
11.	$B_{A,\tau_c} \exists_\tau \tau(\tau_0 \leq \tau \wedge R(B, \tau, \llbracket A \rrbracket) \wedge H(B, \tau, A))$	$P_A, A10\,7), A3\,3)$

Then, let's turn to G_2. Similar to the proof of $G1$, we can prove the following conclusion:

$$B_{B,\tau_c} \exists_\tau \tau(\tau_0 \leq \tau \wedge R(A, \tau, \llbracket N_b \rrbracket) \wedge H(A, \tau, N_b)) \tag{7}$$

Unfortunately, this is not our goal, and no further conclusion about this goal can be draw. This implies that B cannot assure whether A knows his identity. As known, this is just the vulnerability of the NS protocol. [6] gives the details on this vulnerability and provides an improved protocol (NSL) which adds B to the second message. We can prove G_2 in NSL with the similar proof of G_1 in NS.

Often, to model the authentication goal, a variant of formula (7) is frequently used. From the above example, we can see that such modeling may leave the vulnerability neglected. So, this example shows that the modeling scheme in $\mathcal{L}$ can capture the goals of the protocols more accurately.

6 Conclusion

We present a timed logic for modeling and reasoning about security protocols. In this logic, time is expressed explicitly, which makes it possible to model agents' actions, knowledge or beliefs at different time points. Under this logic, a modeling scheme is provided to formalize protocols and their security properties, especially time-dependent properties. The logic and the modeling scheme can be used to reason about protocols flexibly in several angles. We summery them follows:

– In the angle of time, the following things can be modeled: 1) the running order of protocols, 2) the time-dependent security properties of protocols, 3) fresh values(nonce, random number, and timestamp, etc.), and 4) the properties of partially executed protocols.

– In the angle of space, there exist two views about a protocol: 1) the global view of the protocols and 2) the local view of the protocols.

The result of our case study shows that the logic can accurately model protocols, capture the goals of protocols and is helpful in finding vulnerability.

Further work can be done to give the application of $\mathcal{L}$ in analyzing various security protocols. It would be interesting to know whether the logic can be used to analyze the non-repudiation and fairness of the protocols, especially the fairness with time limit. Still another work can be done to show whether the logic is computationally sound.

References

1. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems **8**(1) (February 1990) 18–36
2. Gong, L., Needham, R., Yahalom, R.: Reasoning about belief in cryptographic protocols. In Cooper, D., Lunt, T., eds.: Proceedings 1990 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society (1990) 234–248
3. Abadi, M., Tuttle, M.R.: A semantics for a logic of authentication (extended abstract). In: PODC. (1991) 201–216
4. Syverson, P., van Oorschot, P.C.: On unifying some cryptographic protocol logics. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, IEEE Computer Society Press (1994) 14–28
5. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The spi calculus. Information and Computation **148**(1) (10 January 1999) 1–70
6. Lowe, G.: Breaking and fixing the needham schroeder public-key protocol using FDR. In: Proceedings of TACAS. Volume 1055., Springer Verlag (1996) 147–166
7. Fabrega, F.J.T., Herzog, J.C., Guttman, J.D.: Strand spaces: Why is a security protocol correct. In: Proceedings of the 1998 Conference on Security and Privacy (S&P-98), IEEE Press (May 1998) 160–171
8. Guttman, J.D., Thayer, F.J.: Authentication tests and the structure of bundles. Theor. Comput. Sci **283**(2) (2002) 333–380
9. Doghmi, S.F., Guttman, J.D., Thayer, F.J.: Searching for shapes in cryptographic protocols. In Grumberg, O., Huth, M., eds.: TACAS. Volume 4424 of Lecture Notes in Computer Science., Springer (2007) 523–537
10. Guttman, J.: Security theorems via model theory. CoRR **abs/0911.2036** (2009)
11. Dechesne, F., Mousavi, M.R., Orzan, S.: Operational and epistemic approaches to protocol analysis: Bridging the gap. In Dershowitz, N., Voronkov, A., eds.: LPAR. Volume 4790 of Lecture Notes in Computer Science., Springer (2007) 226–241
12. Chadha, R., Delaune, S., Kremer, S.: Epistemic logic for the applied pi calculus. In Lee, D., Lopes, A., Poetzsch-Heffter, A., eds.: FMOODS/FORTE. Volume 5522 of Lecture Notes in Computer Science., Springer (2009) 182–197
13. Blanchet, B.: Automatic verification of correspondences for security protocols. Journal of Computer Security **17**(4) (2009) 363–434
14. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684 (February 1996)
15. Takács, P.: The additional examination of the kudo-mathuria time-release protocol. J. UCS **12**(9) (2006)

16. Zhang, Y., Varadharajan, V.: A logic for modeling the dynamics of beliefs in cryptographic protocols, IEEE Computer Society (2001) 215–222
17. Kim, K., Park, S., Baek, J.: Improving fairness and privacy of zhou-gollmann's fair non-repudiation protocol. In: Proceedings of the 1999 International Conference on Parallel Processing (ICPP '99). (1999) 140–145
18. Coffey, Saidha: Logic for verifying public-key cryptographic protocols. IEEPCDT: IEE Proceedings on Computers and Digital Techniques **144** (1997)
19. Kudo, M., Mathuria, A.: An extended logic for analyzing timed-release public-key protocols. In Varadharajan, V., Mu, Y., eds.: ICICS. Volume 1726 of Lecture Notes in Computer Science., Springer (1999) 183–198
20. Syverson, P.F.: Adding time to a logic of authentication. In: ACM Conference on Computer and Communications Security. (1993) 97–101
21. Dixon, C., Gago, M.C.F., Fisher, M., van der Hoek, W.: Temporal logics of knowledge and their applications in security. Electr. Notes Theor. Comput. Sci **186** (2007) 27–42
22. Kramer, S.: Cryptographic protocol logic: Satisfaction for (timed) dolev-yao cryptography. J. Log. Algebr. Program **77**(1-2) (2008) 60–91
23. Haack, C., Jeffrey, A.: Timed spi-calculus with types for secrecy and authenticity. In Abadi, M., de Alfaro, L., eds.: CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA, August 23-26, 2005, Proceedings. Volume 3653 of Lecture Notes in Computer Science., Springer (2005) 202–216
24. Datta, A., Derek, A., Mitchell, J.C., Roy, A.: Protocol composition logic (PCL). Electr. Notes Theor. Comput. Sci **172** (2007) 311–358
25. Cathalo, J., Libert, B., Quisquater, J.J.: Efficient and non-interactive timed-release encryption. In Qing, S., Mao, W., Lopez, J., Wang, G., eds.: Information and Communications Security, 7th International Conference, ICICS 2005, Beijing, China, December 10-13, 2005, Proceedings. Volume 3783 of Lecture Notes in Computer Science., Springer (2005) 291–303
26. Jakubowska, G., Penczek, W.: Modelling and checking timed authentication of security protocols. Fundam. Inform **79**(3-4) (2007) 363–378
27. Corin, R., Etalle, S., Hartel, P.H., Mader, A.: Timed analysis of security protocols. Journal of Computer Security **15**(6) (2007) 619C645
28. Schaller, P., Schmidt, B., Basin, D.A., Capkun, S.: Modeling and verifying physical properties of security protocols for wireless networks. In: CSF, IEEE Computer Society (2009) 109–123
29. Ölveczky, P.C., Grimeland, M.: Formal analysis of time-dependent cryptographic protocols in real-time maude. In: IPDPS, IEEE (2007) 1–8
30. Xinfeng, L., jun, L., Junmo, X.: Time-dependent cryptographic protocol logic and its formal semantics. Journal of software **To appear** (2010)
31. Dolev, D., Yao, A.C.: On the security of public-key protocols. IEEE Transactions on Information Theory **30**(2) (1983) 198–208
32. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Computing **13**(4) (November 1984) 850–863
33. Goldwasser, S., Micali, S.: Probabilistic encryption. JCSS **28**(2) (April 1984) 270–299
34. Abadi, Rogaway: Reconciling two views of cryptography (the computational soundness of formal encryption). JCRYPTOL: Journal of Cryptology **15** (2002)
35. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In Naor, M., ed.: Theory of Cryptography, First Theory

of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings. Volume 2951 of Lecture Notes in Computer Science., Springer (2004) 133–151
36. Micciancio, D., Warinschi, B.: Completeness theorems for the Abadi-Rogaway language of encrypted expressions. Journal of Computer Security **12**(1) (2004) 99–130
37. Impagliazzo, R., Kapron, B.M.: Logics for reasoning about cryptographic constructions. J. Comput. Syst. Sci **72**(2) (2006) 286–320
38. Blanchet, B.: A computationally sound mechanized prover for security protocols. IEEE Trans. Dependable Sec. Comput **5**(4) (2008) 193–207
39. Barthe, G., Daubignard, M., Kapron, B., Lakhnech, Y.: Computational frames functional semantics and logics. Technical report (2010)
40. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Communications of the ACM **21**(12) (December 1978) 993–999