Automatic Verification of Complex Security Protocols With an Unbounded Number of Sessions

Kaile Su, Weiya Yue and Qingliang Chen
Department of Computer Science, Sun Yat-sen University Guangzhou, P.R. China

Abdul Sattar
Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, Australia

Mehmet A Orgun
Department of Computing, Macquarie University, Sydney, Australia

Abstract: We present a new protocol logic, called Logic of Local Sessions (LLS), which is based on a quite natural semantic model called Instantiation Space. Given a protocol and a log file for a principal's message data flow, which is formalized in Cryptographical Message Exchange model (CME), the notion of Instantiation Space is used to identify all the protocol's local runs carried out by the principal. LLS is implemented and resulted in a robust Security Protocol Verifier (SPV), which is particularly suitable for automatic verification of complex protocols with an unbounded number of sessions. Due to the flexibility of LLS, the current version of SPV can deal with complex message formats with arbitrarily nested encryptions by public, private, shared and hash keys as well as freshly generated keys. Also, SPV can be used to verify complex security properties such as "Alice observes (knows) Bob observes (knows) Alice said something". Most importantly, it has been applied to automatically verify a lot of interesting and important properties for quite complex security protocols like Kerberos V5 and the SET purchase phase protocol.

Contents

1 Introduction

Security protocols are communication protocols which are used to perform some information exchanges in the presence of malicious intruders over a public network. An intruder can listen to the messages being sent and also block or forge messages as defined in the *Dolev-Yao model* [19]. Although security protocols typically use the mechanism of encrypted communication for ensuring security, they may not be able to ensure the desired security goals due the *logical flaws* in the design of the protocol instead of the defects and deficiencies of the underlying cryptographic algorithms (see [28] for an illuminating account). There are two main approaches to analyze logical flaws in security protocols: the falsification-oriented approach and the justification-oriented approach. The falsification-oriented approach aims at finding bugs or possible attack traces with a protocol and can not guarantee the correctness of a protocol, while the justification-oriented approach is dedicated to verifying some important security properties of a protocol and guaranteeing the correctness of the protocol. There are many efficient tools in the falsification-oriented approach, while the justification-based approach seems difficult to automate.

Recently, there have been a number of influential tools and approaches for producing proofs of correctness in the Dolev-Yao model, such as Durgin's protocol compositional logic [20], Paulson's inductive method [33], and Blanchet's ProVerif [9]. However, most of them are either not fully automated or not suitable for analyzing large-scale and complex protocols like the SET purchase phase protocol. Another limitation is that they can not deal with epistemic protocol goals like "Alice knows Bob knows something", which are considered seriously in the earlier protocol logics like the BAN logic [11]. Our work is along the justification-oriented approach and aims at addressing both of the limitations discussed above in that it is fully automated and can be used for verifying interesting and important properties for complex security protocols.

In this paper, we first present a new protocol logic, called Logic of Local Sessions (LLS) and a new model of security protocol logics, which is called an Instantiation Space. The intuition behind this model is that the message terms in a protocol are essentially message variables and for a principal to execute a particular (local) session of the protocol is just to instantiate those message variables. Given a protocol and a log file for a principal's message data flow, it is interesting to mine all or some protocol's local runs from the principal's log file. In our approach, principals' log files are formalized as Cryptographical Message Exchange Model (CME); the notion of Instantiation Space is used to identify the protocol's local runs (complete or incomplete) in such a log file.

A set of interesting axioms (security properties) are proved to be sound with respect to the proposed Instantiation Space model. Some of them provide new methodologies for reasoning about security protocols. For example, the axioms of $(n, 1)$-Secrecy Properties are beyond the capabilities of the predicate `Source` in compositional logic [20] and the *challenge-response axiom schema* and even the *send-receive axiom schema* in Cervesato *et al* [14] (see Remarks 5.7 and 5.8 in Section 4 for further details).

We have also implemented a new Security Protocol Verifier (SPV) based on SAT solvers. With SPV, we are able to automatically verify some interesting properties for complex security protocols. Even for the complex SET purchase phase protocol [8], SPV takes only several hours to generate eighty thousand rules and then verify many concerned specifications in several seconds (see the experimental results reported in Appendix B). Although the SET purchase phase protocol can be handled by semi-automatic verifiers like theorem prover Isabelle [8], to the best of our knowledge, its security properties have not been proved by the fully automatic approaches other than SPV.

To prove multi-level epistemic goals like "Alice observes (knows) Bob observes (knows) Alice said something", we explicitly indicate, for each principal, the *observability* of each variable appearing in those axioms. Then, we propose a key notion of *principal's observation*, which is somewhat similar to a principal's belief or knowledge in earlier approaches based on logics of knowledge. However, a principal's observation in our approach is simply and directly defined by the set of axioms and the set of variables observable to the principal. In contrast, many previous approaches based on logics of knowledge (or belief) to the verification of security protocols, such as the BAN logic [11] and many of its various alternatives and variants [23, 2, 13, 40, 37], use logics of knowledge mainly as a specification language and do not explain from where and how a principal's knowledge exactly comes.

The general protocol analysis problem is known to be undecidable [3]. As a result, our solution only captures a fragment of this undecidable problem. It may fail to prove a valid security property for a given protocol. However, we have verified a lot of important complex security protocols and have not yet experienced such a case.

The structure of this paper is as follows. In the next section, we propose the notion of an observation theory, and show how the verification problem for epistemic goals can be reduced into the satisfiability problem by using this methodology. In Section 3, we present Cryptographical Message Exchange (CME) model, with which we define the notion of *Instantiation Spaces* in Section 4. Section 5 proposes a Logic of Local Sessions (LLS) with respect to the Instantiation Space model. Some important axioms like $(n, 1)$-Secrecy Properties are presented and proved to be sound. Section 6 demonstrates applications of LLS in automatic verification of security goals like SECRET and PRECEDES in CAPSL [30], as well as multi-level epistemic goals based on the notion of an observation theory. We also exemplify our approach by verifying the Revised Needham-Schroeder protocol and the Kerberos V5 protocol. Experimental results are summarized in Section 7. Finally, we discuss related work and conclude this paper.

2 Reasoning about Principals' Observations

To verify more interesting and complex security properties such as "Alice observes Bob observes Alice said something", we propose the notion of an observation theory, and show how the verification problem for security properties can be reduced into the satisfiability problem.

2.1 Observation Theory, Observable Formulas and Observations

We define an *observation theory* Γ (with n principals) as a tuple of the form $(V, \Theta, O_1, \ldots, O_n)$, where V is a finite set of propositional variables, Θ a set of Boolean formulas over V and, for each i, O_i a subset of V. Moreover, we say variables in V are *system variables* of the observation theory, and those variables in O_i *observable variables* to principal i. We call Θ the *common knowledge base*; intuitively, it is a list of common assumptions about the principals and their communication environment. For convenience, we use also Θ to denote the conjunction of all formulas in Θ.

We define principals' observation as follows:

Definition 2.1 Suppose we are given an observation theory $(V, \Theta, O_1, \ldots, O_n)$ and a formula φ. For every $i \leq n$, we call φ i-observable if φ is a formula over O_i. Let F_i be the set of all i-observable formulas.

Definition 2.2 Given an observation theory $(V, \Theta, O_1, \ldots, O_n)$ and a formula φ. For every $i \leq n$, we define $K_i\varphi$ to be the following formula:

$$\bigvee_{\alpha \in F_i,\ \Theta \vdash \alpha \Rightarrow \varphi} \alpha$$

By our definition, $K_i\varphi$ holds iff there is an i-observable formula α such that α holds and α logically implies φ under the common knowledge base Θ. Intuitively, $K_i\varphi$ means that principal i observes that φ holds (or principal i observes φ, for short).

The notion of *principal's observation*, is somewhat similar to a principal's belief or knowledge in previous approaches based on logics of knowledge. However, a principal's observation in our approach is directly defined just by the set of axioms and the set of variables observable to the principal.

Proposition 2.3 *Given an observation theory $(V, \Theta, O_1, \ldots, O_n)$ and formulas φ and ψ, we have that*

1. $\Theta \vdash K_i\varphi \Rightarrow \varphi$.
2. *If φ is an i-observational formula, then* $\Theta \vdash K_i\varphi \Leftrightarrow \varphi$.
3. *If $\Theta \vdash \varphi \Rightarrow \psi$, then* $\Theta \vdash K_i\varphi \Rightarrow K_i\psi$

Proof: Straightforward by the definition. ∎

Example 2.4 Let us consider the scenario that Alice sends Bob a message and Bob sends Alice an acknowledgement upon receiving the message. We assume Alice and Bob commonly have the following background knowledge base Θ_{AB}:

$$\begin{array}{l} \mathit{Bob\_recv\_msg} \Rightarrow \mathit{Alice\_send\_msg} \\ \mathit{Bob\_send\_ack} \Rightarrow \mathit{Bob\_recv\_msg} \\ \mathit{Alice\_recv\_ack} \Rightarrow \mathit{Bob\_send\_ack} \end{array}$$

where *Bob_recv_msg* and *Bob_send_ack* are *observable* variables to Bob, while *Alice_send_msg* and *Alice_recv_ack* are *observable* to Alice. Then, we have the observation theory

$$\Gamma_{AB} = (V_{AB}, \Theta_{AB}, O_A, O_B)$$

where

$$\begin{array}{ll} O_A & = \{\mathit{Alice\_send\_msg}, \mathit{Alice\_recv\_ack}\} \\ O_B & = \{\mathit{Bob\_recv\_msg}, \mathit{Bob\_send\_ack}\} \\ V_{AB} & = O_A \cup O_B. \end{array}$$

It is not difficult to establish that $\Theta_{AB} \vdash \mathit{Alice\_recv\_ack} \Rightarrow K_A K_B \mathit{Alice\_send\_msg}$, which means that if Alice receives the acknowledgement from Bob, then Alice observes that Bob observes that Alice sent the message. However, we can not establish that $\Theta_{AB} \vdash \mathit{Alice\_recv\_ack} \Rightarrow K_A K_B \mathit{Alice\_recv\_ack}$; in other words, even if Alice receives the acknowledgement from Bob, Alice may not observe that Bob observes that Alice receives the acknowledgement from Bob.

2.2 Observations and Satisfiability

Given an observation theory $(V, \Theta, O_1, \ldots, O_n)$, let O_i denote the conjunction of all formulas in O_i. Given an assignment or a subset σ of O_i and a formula α, let $\alpha \cdot \sigma$ be the formula resulting from α by substituting *true* for those variables in σ and *false* for those in $O_i - \sigma$.

Proposition 2.5 *For an arbitrary formula φ on V and an i-observable formula α, $\Theta \vdash \alpha \Rightarrow K_i\varphi$ iff $\Theta \wedge \alpha \wedge \neg\varphi$ is unsatisfiable.*

Proof: If $\Theta \wedge \alpha \wedge \neg\varphi \in SAT$, then there exists an assignment σ such that $\Theta \cdot \sigma = \mathbf{true}$, $\alpha \cdot \sigma = \mathbf{true}$ and $\varphi \cdot \sigma = \mathbf{false}$. Under assignment σ, for each $\beta \in F_i$ such that $\Theta \vdash \beta \Rightarrow \varphi$, we have $\beta \cdot \sigma = \mathbf{false}$. Therefore, $K_i\varphi \cdot \sigma = \mathbf{false}$. Thus, $\Theta \models \alpha \Rightarrow K_i\varphi$ is not valid under this assignment σ. So, we have $\Theta \nvdash \alpha \Rightarrow K_i\varphi$

On the other hand, if $\Theta \wedge \alpha \wedge \neg\varphi$ is unsatisfiable, then $\Theta \vdash \alpha \Rightarrow \varphi$. We immediately have $\vdash \alpha \Rightarrow K_i\varphi$ since α is an i-observable formula ∎

By the above proposition, the verification problem of the specification $\alpha \Rightarrow K_i\varphi$ can be reduced into the satisfiability problem of $\Theta \wedge \alpha \wedge \neg\varphi$.

Proposition 2.6 *For an arbitrary formula φ on V and an i-observable formula α, $\Theta \vdash \alpha \Rightarrow K_iK_j\varphi$ iff for every assignment σ of O_j, if $\Theta \cdot \sigma \wedge \alpha \cdot \sigma$ is satisfiable, then $(\Theta \wedge \neg\varphi) \cdot \sigma$ is unsatisfiable.*

Proof: Suppose for some assignment σ of O_j, $\Theta \cdot \sigma \wedge \alpha \cdot \sigma$ is satisfiable and so is $(\Theta \wedge \neg\varphi) \cdot \sigma$. We will see that $\Theta \nvdash \alpha \Rightarrow K_iK_j\varphi$

There must exist an assignment δ for $V - O_j$ such that $\Theta \cdot \sigma \cdot \delta = \mathbf{true}$ and $\varphi \cdot \sigma \cdot \delta = \mathbf{false}$. Then under the assignments δ and σ, every $\beta \in F_j$ with $\Theta \vdash \beta \Rightarrow \varphi$ must be false. Therefore, we have $K_j\varphi \cdot \delta \cdot \sigma = \mathbf{false}$. Based on $K_j\varphi \cdot \delta \cdot \sigma = \mathbf{false}$, the similar argument will lead to $K_iK_j\varphi \cdot \delta \cdot \sigma = \mathbf{false}$. Hence, $\Theta \nvdash \alpha \Rightarrow K_iK_j\varphi$.

From the other direction, suppose for every assignment σ of O_j, if $\Theta \cdot \sigma \wedge \alpha \cdot \sigma$ is satisfiable, then $(\Theta \wedge \neg\varphi) \cdot \sigma$ is unsatisfiable. To show $\Theta \vdash \alpha \Rightarrow K_iK_j\varphi$ holds, by Proposition 2.5, we need only to show $\Theta \wedge \alpha \wedge \neg K_j\varphi$ is unsatisfiable. For this purpose, suppose that $\Theta \wedge \alpha \wedge \neg K_j\varphi$ is satisfied by an assignment δ for V.

Let σ_0 be the assignment of O_j that consistent with δ. We have that $\Theta \cdot \sigma_0 \wedge \alpha \cdot \sigma_0$ is satisfiable. On the other hand, let $\gamma_0 \in F_j$ be the such that $\gamma_0 \cdot \sigma_0$ is **true**. Then $\gamma_0 \cdot \delta$ is **true**. By the assumption that $\neg K_j\varphi$ is satisfied by δ, we conclude that $\Theta \nvdash \gamma_0 \Rightarrow \varphi$. Thus, there is an assignment δ' by which $\Theta \wedge \gamma_0 \wedge \neg\varphi$ is satisfied. Assignment δ' must be consistent with σ_0 because γ_0 is satisfied by δ'. Thus, $\Theta \cdot \sigma_0 \wedge \neg\varphi \cdot \sigma_0$ is also satisfiable. Therefore, we find an assignment σ_0 of O_j such that both $\Theta \cdot \sigma_0 \wedge \alpha \cdot \sigma_0$ and $(\Theta \wedge \neg\varphi) \cdot \sigma_0$ is satisfiable. This is a contradiction. ∎

By the above result, when there are not too many assignments σ of O_j such that $\Theta \cdot \sigma \wedge \alpha \cdot \sigma$ is satisfiable, we can check those specifications of the form $\alpha \Rightarrow K_iK_j\varphi$ by a SAT solver.

3 Cryptographical Message Exchange Model

In this section, we give some preliminary definitions on message algebra and introduce the notion of Cryptographical Message Exchange (CME) model, which is used in defining our semantic model: Instantiation Space.

3.1 Message Algebra

In our model, there are three types of messages: plain text, concatenation and cryptical messages. Keys are plain text messages. A message algebra is a pair $(\mathcal{M}, \mathcal{K})$, where $\mathcal{K} \subseteq \mathcal{M}$ contains a special key $hash$, having the following operations:

- Inverse key: If $k \in \mathcal{K}$ and $k \neq hash$, then $k^{-1} \in \mathcal{K}$.
- Concatenation: If $m_1, \cdots, m_n \in \mathcal{M}$ then $[m_1, \cdots, m_n] \in \mathcal{M}$
- Decomposition: If $[m_1, \cdots, m_n] \in \mathcal{M}$ then $m_1, \cdots, m_n \in \mathcal{M}$.
- Encryption: If $k \in \mathcal{K}$ and $m \in \mathcal{M}$ and m does not have the form $\{m'\}_{k^{-1}}$, then $\{m\}_k \in \mathcal{M}$.
- Decryption: If $k^{-1} \in \mathcal{K}$ and $\{m\}_k \in \mathcal{K}$, then $m \in \mathcal{M}$.

We assume that

1. if $\{m_1\}_{k_1} = \{m_2\}_{k_2}$ then $m_1 = m_2$ and $k_1 = k_2$; and
2. if $[m_1, \cdots, m_n] = [m'_1, \cdots, m'_{n'}]$ then $n = n'$ and $m_i = m'_i$ for $0 < i \leq n$[1].

Let $\mathcal{M}'$ be a set of messages, $\mathcal{K}'$ a set of keys, and m_0 a message. We introduce the following notations. We omit detailed the definitions, which can be carried out by routine structural induction on messages.

1. $cl(\mathcal{M}')$ is the set of messages obtained from $\mathcal{M}'$ by encryption, decryption, decomposition and concatenation.
2. $cl^-(\mathcal{M}')$ is the set of messages obtained from $\mathcal{M}'$ by decryption and decomposition.
3. $cl^+(\mathcal{M}')$ is the set of messages obtained from $\mathcal{M}'$ by encryption and concatenation.
4. $\mathbf{Sub}(m_0, \mathcal{K}')$ is the set of submessages of m_0 obtained by decomposition and by decryption with keys in $\mathcal{K}'$.

   Similarly, we define $\mathbf{Sub}(\mathcal{M}', \mathcal{K}') = \bigcup_{m \in \mathcal{M}'} \mathbf{Sub}(m, \mathcal{K}')$.

We have the following lemma.

Lemma 3.1 *For a set of message $\mathcal{M}'$,*

$$cl(\mathcal{M}') = cl^+(cl^-(\mathcal{M}'))$$

Proof: It suffices to show that $cl^+(cl^-(\mathcal{M}'))$ is closed under decryption and decomposition. This can be done if prove that for every message $m \in cl^+(cl^-(\mathcal{M}'))$, the following claim holds:

If $m = [m_1, \cdots, m_h]$, then $m_i \in cl^+(cl^-(\mathcal{M}'))$, for $i = 1, \cdots, h$; and if $m = \{m'\}_k$, then $m' \in cl^+(cl^-(\mathcal{M}'))$.

But by the definition of $m \in cl^+(cl^-(\mathcal{M}'))$, there are only the following three cases:

1. $m \in cl^-(\mathcal{M}')$
2. $m = [m_1, \cdots, m_h]$ and $m_i \in cl^+(cl^-(\mathcal{M}'))$
3. $m = \{m'\}_k$ and $m', k \in cl^+(cl^-(\mathcal{M}'))$.

In all the above case, the claim for m is clearly true. ∎

[1] To deal with type-flaw attacks, SPV permits, for instance, $[m_1, m_2, m_3] = [m_1, [m_2, m_3]]$.

3.2 The Notion of CME Model

We introduce the notion of a Cryptographical Message Exchange model (or a CME model, for short), which is a tuple $\Sigma = (\mathcal{M}, \mathcal{K}, e, Ag, \mathbf{Pk}, \mathbf{Sk}, \mathbf{Label}, \mathbf{Nonce}, \mathbf{Init}, \mathbf{Recv}, \mathbf{Sent})$, where

- $\mathcal{M}$ is a set of messages.
- $\mathcal{K}$, a subset of $\mathcal{M}$, is a set of keys.
- e is the communication environment possibly under the control of attackers.
- Ag is a set of agents or principals. For convenience, we assume that $Ag \subseteq \mathcal{M}$.
- **Pk** consists of those tuples $\langle a, k, k^{-1} \rangle$, which indicates that k is role a's public key and k^{-1} the corresponding private key.
- **Sk** consists of those tuples $\langle a_1, a_2, m \rangle$, which indicates that m is the shared secret of principals a_1 and a_2. Usually $m \in \mathcal{K}$ and in this case, m is the shared key of principals a_1 and a_2.
- **Label** is a set of messages representing timestamps, session ID numbers, or nonces. Messages in **Label** are called *label messages*.
- **Nonce** is a function that associates each principal $a \in Ag \cup \{e\}$ with a set of nonces or challenge numbers generated by a. We have that $\mathbf{Nonce}(a) \subseteq \mathbf{Label}$ for all $a \in Ag \cup \{e\}$.
- **Init** is a function that associates every $a \in Ag \cup \{e\}$ with a set of plain text messages that a initially possesses [2].
- **Recv** is a function that associates each principal $a \in Ag \cup \{e\}$ and each natural number (time point) t with a set of messages, which expresses the set of messages that principal a receives at time t.
- **Sent** is a function that associates each principal $a \in Ag \cup \{e\}$ and each natural number (time point) t with a set of messages, which expresses the set of messages sent by a at time t.

Given a principal $\rho \in Ag$ and a nonce or a timestamp or session ID number c, we assume there is at most one ρ's local run of protocol $\mathcal{P}$ using message c. This assumption is reasonable for an honest principal. Thus message c can be regarded as a label of local session of a protocol run. This is why we call nonces, timestamps and session ID numbers *label messages*. Label messages can be divided two in two classes: unguessable ones like nonces, and guessable ones like timestamps and session ID numbers.

We define functions **Recvs** and **Sents** by $\mathbf{Recvs}(a, t) = \bigcup_{t' \leq t} \mathbf{Recv}(a, t')$ and $\mathbf{Sents}(a, t) = \bigcup_{t' \leq t} \mathbf{Sent}(a, t')$.

For every $a \in Ag$, let $\mathcal{K}_a = \mathbf{Init}(a) \cap \mathcal{K}$. Clearly, $\mathcal{K}_a$ is the set of keys that a initially possesses. To express the set of those keys that e initially possesses, let $\mathcal{K}_e = \mathcal{K} - (\mathbf{PrK} \cup \mathbf{ShK})$, where

$$\mathbf{PrK} = \{k^{-1} \mid \langle a, k, k^{-1} \rangle \in \mathbf{Pk} \text{ for some } a \in Ag\}$$
$$\mathbf{ShK} = \{k \mid \langle a_1, a_2, k \rangle \in \mathbf{Sk} \text{ for some } a_1, a_2 \in Ag\}.$$

Here, we assume that environment e has all keys except principals' private keys and shared keys among principals in Ag.

Let **Poss** be the function such that for every $a \in Ag \cup \{e\}$ and every natural number (time point) t, $\mathbf{Poss}(a, t) = cl(\mathbf{Init}(a) \cup \mathbf{Recvs}(a, t))$. We make the following assumptions about CME models:

[2] $\mathbf{Init}(a)$ can be infinite and include $\mathbf{Nonce}(a)$; without any loss of generality, we assume that a principal generates all his nonces (including dynamically generated session keys) initially but never used until a particular session.

1. For every $a \in Ag$ and $m \in M$, we have that
   (a) if $m \in \mathbf{Sents}(a,t)$, then $m \in \mathbf{Recvs}(e,t+1)$;
   (b) if $m \in \mathbf{Recvs}(a,t)$, then $m \in \mathbf{Sents}(e,t-1)$.

   This assumption indicates that every message sent by a principal is received by the environment and every message received by a principal is sent by the environment.

2. If $m \in \mathbf{Recvs}(e,t)$, then $m \in \mathbf{Sents}(a,t-1)$ for some $a \in Ag$.

   This assumption implies that the environment includes every principal other than those in Ag.

3. For every $a \in Ag \cup \{e\}$, $\mathbf{Sents}(a,t) \subseteq \mathbf{Poss}(a,t-1)$.

4. For every $a \in Ag \cup \{e\}$, and $t' < t$, $\mathbf{Sents}(a,t') \subseteq \mathbf{Sents}(a,t)$ and $\mathbf{Recvs}(a,t') \subseteq \mathbf{Recvs}(a,t)$.

5. (a) For $a \in Ag$ and $k \in \mathcal{K}$, if $\langle a, k, k^{-1}\rangle \in \mathbf{Pk}$, then $k^{-1} \notin \mathbf{Init}(a')$ for all $a' \in Ag \cup \{e\}$ other than a.
   (b) For $a_1, a_2 \in Ag$, and $m \in \mathcal{M}$, if $\langle a_1, a_2, m\rangle \in \mathbf{Sk}$, then $m \in \mathbf{Init}(a_1)$ and $m \in \mathbf{Init}(a_2)$ but $m \notin \mathbf{Init}(a')$ for all $a' \in Ag \cup \{e\}$ other than a_1 and a_2.
   (c) For $a \in Ag \cup \{e\}$ and $m \in \mathbf{Nonce}(a)$, we have that $m \in \mathbf{Init}(a)$ but $m \notin \mathbf{Init}(a')$ for every $a' \in Ag \cup \{e\}$ other than a.

   The above assumption says that the environment or a principal does not initially possess a key that the principal should not have, nor a challenge number generated by others.

To explore the properties of CME model, we introduce some more notations. We define $\mathbf{Keys}(a,t) = \mathbf{Poss}(a,t) \cap \mathcal{K}$, and $\mathbf{Sees}(a,t) = \mathbf{Sub}(\mathbf{Recvs}(a,t), \mathbf{Keys}(a,t))$.

Recalling that $\mathbf{Poss}(a,t) = cl(\mathbf{Init}(a) \cup \mathbf{Recvs}(a,t))$, we have $\mathbf{Poss}(a,t) = cl^{+}(cl^{-}(\mathbf{Init}(a) \cup \mathbf{Recvs}(a,t)))$. However, we have that

$$\begin{aligned} & cl^{-}(\mathbf{Init}(a) \cup \mathbf{Recvs}(a,t)) \\ = \; & \mathbf{Sub}(\mathbf{Init}(a) \cup \mathbf{Recvs}(a,t), \mathbf{Keys}(a,t)) \\ = \; & \mathbf{Sub}(\mathbf{Init}(a), \mathbf{Keys}(a,t)) \cup \mathbf{Sub}(\mathbf{Recvs}(a,t), \mathbf{Keys}(a,t)) \\ = \; & \mathbf{Init}(a) \cup \mathbf{Sees}(a,t). \end{aligned}$$

Therefore, we get $\mathbf{Poss}(a,t) = cl^{+}(\mathbf{Init}(a) \cup \mathbf{Sees}(a,t))$.

Also, we define $\mathbf{Said}(a,t)$ to be the least set $\mathcal{M}'$ such that

1. $\mathbf{Sents}(a,t) \subseteq \mathcal{M}'$;
2. if $[m_1, \cdots, m_n] \in \mathcal{M}'$ then $m_1, \cdots, m_n \in \mathcal{M}'$; and
3. if $\{m\}_k \in \mathcal{M}'$ and $m, k \in \mathbf{Poss}(a,t)$, then $m \in \mathcal{M}'$.

We set

$$\mathbf{Encr}(a,t) = \{\{m\}_k \,|k \in \mathbf{Poss}(a,t) \text{ and } m \in \mathbf{Poss}(a,t)\}$$
$$\mathbf{Recg}(a,t) = \left\{\{m\}_k \,\middle|\, k^{-1} \in \mathbf{Poss}(a,t) \text{ or both } k \in \mathbf{Poss}(a,t) \text{ and } m \in \mathbf{Poss}(a,t)\right\}.$$

We notice that $\mathbf{Encr}(a,t)$ is the set of those messages that a can construct by encryption, and $\mathbf{Recg}(a,t)$ the set of those messages recognizable to a.

We define $\mathbf{Fresh}(a, m_0, t)$ as follows.

1. $m_0 \in \mathbf{Fresh}(a, m_0, t)$;

2. $[m_1, \cdots, m_n] \in \mathbf{Fresh}(a, m_0, t)$, if for some i $(1 \leq i \leq n)$ we have that $m_i \in \mathbf{Fresh}(a, m_0, t)$;

3. $\{m\}_k \in \mathbf{Fresh}(a, m_0, t)$, if $\{m\}_k \in \mathbf{Recg}(a, t)$, and m or k is in $\mathbf{Fresh}(a, m_0, t)$.

The intuition behind $\mathbf{Fresh}(a, m_0, t)$ is that if m_0 is fresh, then so are messages in $\mathbf{Fresh}(a, m_0, t)$ from the principal a's point of view.

3.3 Some Important Properties about CME Models

In this subsection, we give some important properties about CME models. Just by those properties, we are able to prove the validity of some interesting axioms like $(n, 1)$-Secrecy Properties, which turn out to be very useful for verifying complex security protocols.

Given $\mathcal{A}' \subseteq Ag \cup \{e\}$, we also use $\mathbf{Keys}(\mathcal{A}', t)$ to denote $\bigcup_{a \in \mathcal{A}'} \mathbf{Keys}(a, t)$. The notations such as $\mathbf{Poss}(\mathcal{A}', t)$, $\mathbf{Sents}(\mathcal{A}', t)$, $\mathbf{Recvs}(\mathcal{A}', t)$, $\mathbf{Sees}(\mathcal{A}', t)$, $\mathbf{Said}(\mathcal{A}', t)$, $\mathbf{Init}(\mathcal{A}')$, $\mathbf{Nonce}(\mathcal{A}')$, and $\mathcal{K}_{\mathcal{A}'}$ etc can be introduced in the same way. We define $\mathbf{Encr}(\mathcal{A}', t)$ as $\{\{m\}_k \mid k \in \mathbf{Poss}(\mathcal{A}', t) \text{ and } m \in \mathbf{Poss}(\mathcal{A}', t)\}$.

We notice that $\mathcal{K}_{\mathcal{A}'}$ is the set of all keys that group $\mathcal{A}'$ initially possesses, while $\mathbf{Keys}(\mathcal{A}', t)$ is the set of those keys that possessed by someone $\mathcal{A}'$ at time t.

We use $t^{\mathbf{Poss}}(a, m)$ to denote time point t such that $m \in \mathbf{Poss}(a, t)$ but not $m \in \mathbf{Poss}(a, t')$ for any $t' < t$. If there is not such a time point, we set $t^{\mathbf{Poss}}(a, m) = -1$. The notations $t^{\mathbf{Sents}}(a, m)$, $t^{\mathbf{Recvs}}(a, m)$, $t^{\mathbf{Said}}(a, m)$, $t^{\mathbf{Sees}}(a, m)$ can be introduced in the same way. We can also introduce $t^{\mathbf{Poss}}(\mathcal{A}', m)$, $t^{\mathbf{Recvs}}(\mathcal{A}', m)$, $t^{\mathbf{Sees}}(\mathcal{A}', m)$, $t^{\mathbf{Sents}}(\mathcal{A}', m)$ and $t^{\mathbf{Said}}(\mathcal{A}', m)$ for a set $\mathcal{A}' \subseteq Ag \cup \{e\}$.

We say that *a sees m from m′ at time t*, if $m' \in \mathbf{Sees}(a, t)$, and $m \in \mathbf{Sub}(m', \mathbf{Keys}(a, t))$. Given $\mathcal{A} \subseteq Ag \cup \{e\}$, we say that $\mathcal{A}$ *sees m from m′* , if there is a message sequence $m_0, \cdots, m_{l+1}$ with $m_0 = m$, $m' = m_{l+1}$, a time sequence $t_0 < \cdots < t_l$, and principal sequence $a_0, \cdots, a_l$ with $a_i \in \mathcal{A}$ $(0 \leq i \leq l)$ such that, for every $0 \leq i \leq l$, a_i sees m_i from m_{i+1} at time t_i. We say that $\mathcal{A}$ *sees m from m′ at time t*, if $\mathcal{A}$ sees m from m' and $m' \in \mathbf{Sees}(\mathcal{A}, t)$.

The following lemma implies that if a principal possesses a message that can not be constructed by the principal, then the principal must see the message.

Lemma 3.2 *For every $a \in Ag \cup \{e\}$,*

1. *if $m \in \mathbf{Poss}(a, t) \cap \mathbf{Init}(Ag \cup \{e\})$, then either $m \in \mathbf{Init}(a)$ or $m \in \mathbf{Sees}(a, t)$; and*

2. *if $\{m\}_k \in \mathbf{Poss}(a, t) - \mathbf{Encr}(a, t)$ holds, then $\{m\}_k \in \mathbf{Sees}(a, t)$.*

Proof: Notice that $\mathbf{Poss}(a, t) = cl^+(\mathbf{Init}(a) \cup \mathbf{Sees}(a, t))$. If $m \in \mathbf{Init}(Ag \cup \{e\})$, then m is a plain text message. So, $m \in \mathbf{Init}(a) \cup \mathbf{Sees}(a, t)$. As for the second item, suppose $\{m\}_k \in \mathbf{Poss}(a, t) - \mathbf{Encr}(a, t)$. Because $\{m\}_k$ is not of the concatenation form, and it is impossible that both $m \in \mathbf{Poss}(a, t)$ and $k \in \mathbf{Poss}(a, t)$ hold, we have that $\{m\}_k \in \mathbf{Init}(a) \cup \mathbf{Sees}(a, t))$. Thus, $\{m\}_k \in \mathbf{Sees}(a, t)$. ∎

Lemma 3.3 *For every $a \in Ag \cup \{e\}$, if $m \in \mathbf{Sees}(a, t)$ holds, then $m \in \mathbf{Said}(a', t')$ for some $a' \in Ag \cup \{e\}$ and $t' < t$.*

Proof: We prove this lemma by showing that for every $m' \in \mathbf{Sub}(\mathbf{Recvs}(Ag \cup \{e\}, t), \mathcal{K})$, there are an $a' \in Ag \cup \{e\}$ and $t' < t$ such that $m' \in \mathbf{Said}(a', t')$. This can be worked out by induction on m' and by using Lemma 3.2 ∎

The next lemma says that if an encrypted message is seen, then it must be said and constructed by someone before.

Lemma 3.4 *For every $a \in Ag \cup \{e\}$, if $\{m'\}_k \in \mathbf{Sees}(a,t)$ holds, then $\{m'\}_k \in \mathbf{Said}(a',t') \cap \mathbf{Encr}(a',t')$ for some $a' \in Ag \cup \{e\}$ and $t' < t$.*

Proof: Suppose $\{m'\}_k \in \mathbf{Sees}(a,t)$. By Lemma 3.3, $\{m'\}_k \in \mathbf{Said}(a',t')$ for some $a' \in Ag \cup \{e\}$ and $t' < t$. If $\{m'\}_k \notin \mathbf{Encr}(a',t')$, then by Lemmas 3.2 and 3.3, there are $a'' \in Ag \cup \{e\}$ and $t'' < t'$ such that $\{m'\}_k \in \mathbf{Said}(a'',t'')$. Thus, we will have an infinite sequence of time $t' > t'' > \cdots$, which is impossible. ∎

Lemma 3.5 *Given $\mathcal{A}' \subseteq Ag \cup \{e\}$ and message X such that*

1. *$X \in \mathbf{Nonce}(Ag \cup \{e\} - \mathcal{A}')$; or*
2. *X is a shared secret of two principals in $(Ag \cup \{e\} - \mathcal{A}')$; or*
3. *X is of the form $\{X'\}_k$, where X' or k is not in $\mathbf{Poss}(\mathcal{A}',t')$ (i.e., $X \notin \mathbf{Encr}(\mathcal{A}',t')$)*

Suppose that m' is a message such that $\mathcal{A}'$ sees X from m' and $m' \in \mathbf{Poss}(\mathcal{A}',t')$. Then, there is a $m'' \in \mathbf{Sub}(m',\mathcal{K})$ such that $\mathcal{A}'$ sees X from m'' at time t'.

Proof: Suppose $\mathcal{A}'$ sees X from m' and $m' \in \mathbf{Poss}(\mathcal{A}',t')$. First, by the supposition that $\mathcal{A}'$ sees X from m', we have a message sequence $\{m_i\}_{i \le l}$ such that

1. $m_0 = m'$ and $m_l = X$; and
2. for every $i < l$, $\mathcal{A}'$ sees m_i from m_{i+1}, and $m_i = \{m_{i+1}\}_k$ for some k, or m_i is of the concatenation form and m_{i+1} is one of its components.

Since $m' \in \mathbf{Poss}(\mathcal{A}',t')$, there is a greatest number $i_0 \le l$ with $m_{i_0} \in \mathbf{Poss}(\mathcal{A}',t')$.

If $i_0 = l$, then $X \in \mathbf{Poss}(\mathcal{A}',t')$. By the supposition about X and Lemma 3.2, we have that $X \in \mathbf{Sees}(\mathcal{A}',t')$. Thus, $\mathcal{A}'$ sees X from X at time t'.

Let us consider the case of $i_0 < l$. In this case, we have $m_{i_0} = \{m_{i_0+1}\}_k$ for some k and $m_{i_0} \notin \mathbf{Encr}(\mathcal{A}',t')$. Thus, by Lemma 3.2, we have that $m_{i_0} \in \mathbf{Sees}(\mathcal{A}',t')$. Therefore, we have that $\mathcal{A}'$ sees X from m_{i_0} at time t'. ∎

The following lemma says that if someone other than a possesses a message from which a message encrypted with a's private key can be obtained, then a must have said the message encrypted with a's private key.

Lemma 3.6 *Given $a \in Ag$, $\langle a,k,k^{-1}\rangle \in \mathbf{Pk}$ and $\{m'\}_{k^{-1}} \in \mathcal{M}$, if there is an $m'' \in \mathbf{Poss}((Ag \cup \{e\} - \{a\}),t)$ with $\{m'\}_{k^{-1}} \in \mathbf{Sub}(m'',\mathcal{K})$. Then $\{m'\}_{k^{-1}} \in \mathbf{Said}(a,t)$.*

Proof: Suppose that $m' \in \mathbf{Poss}((Ag \cup \{e\} - \{a\}),t)$ and $\{m''\}_{k^{-1}} \in \mathbf{Sub}(m',\mathcal{K})$. Let $\{m_i\}_{i \le l}$ be a sequence of messages such that

1. $m_0 = m'$ and $m_l = m''$; and
2. for every $i < l$, $m_i = \{m_{i+1}\}_k$ for some k, or m_i is of the concatenation form and m_{i+1} is one of its components.

Assume that i_0 is the greatest i such that $m_{i_0} \in \mathbf{Poss}((Ag \cup \{e\} - \{a\}), t)$. If $i_0 = l$, then $\{m''\}_{k^{-1}} \in \mathbf{Poss}(a', t)$ for some $a' \in (Ag \cup \{e\} - \{a\})$. Notice that $\{m''\}_{k^{-1}} \in \mathbf{Encr}(a', t)$ for all $a' \neq a$. By Lemmas 3.2 and 3.3, there are $a'' \in Ag \cup \{e\}$ and $t'' < t$ such that $\{m''\}_{k^{-1}} \in \mathbf{Poss}(a'', t'')$. Thus, if $\{m''\}_{k^{-1}} \notin \mathbf{Poss}(a, t')$ for some $t' < t$, we would have $a'' \in (Ag \cup \{e\} - \{a\})$. Continuing the same argument, there would be an infinite descent sequence of time points, which is impossible.

Consider the case of $i_0 < l$. In this case, m_{i_0} is not of the concatenation form, otherwise $m_{i_0+1} \in \mathbf{Poss}((Ag \cup \{e\} - \{a\}), t)$. Thus, $m_{i_0} = \{m_{i_0+1}\}_k$ for some k. Let $m_{i_0} \in \mathbf{Poss}(a', t)$ for some $a' \in Ag \cup \{e\} - \{a\}$. Then $m_{i_0} \notin \mathbf{Encr}(a', t)$ for $m_{i_0+1} \notin \mathbf{Poss}((Ag \cup \{e\} - \{a\}), t)$. By Lemmas 3.2 and 3.3 and $m_{i_0+1} \notin \mathbf{Poss}((Ag \cup \{e\} - \{a\}), t)$, we conclude that $m_{i_0} \in \mathbf{Said}(a, t')$ for some $t' < t$. Using the similar argument, we can further prove, by induction on i, that for every $i_0 \leq i \leq l$, $m_i \in \mathbf{Said}(a, t'')$ for some $t'' < t$. This completes the proof of the lemma. ∎

Intuition behind the following lemma is that an encrypted message with a shared key should be said by one of the two principals who have the shared key, if the encrypted message is possessed by someone other than the two principals.

Lemma 3.7 *Given $a_1, a_2 \in Ag$, $\langle a_1, a_2, k\rangle \in \mathbf{Sk}$ and $\{m_1\}_k \in \mathcal{M}$, if there is an $m' \in \mathbf{Poss}((Ag \cup \{e\} - \{a_1, a_2\}), t)$ with $\{m_1\}_k \in \mathbf{Sub}(m', \mathcal{K})$. Then $\{m_1\}_k \in \mathbf{Said}(a_1, t)$, or $\{m_1\}_k \in \mathbf{Said}(a_2, t)$.*

Proof: By using the similar argument as in the proof for Lemma 3.6. ∎

Lemma 3.8 *Let $\mathcal{A} \subseteq Ag$, $\mathcal{A}' = (Ag - \mathcal{A}) \cup \{e\}$, and $X \in \mathbf{Poss}(\mathcal{A}', t)$. If $X \in \mathbf{Nonce}(\mathcal{A})$ or X is a shared secret of two principals in $\mathcal{A}$, then there are $t' < t$ and $m' \in \mathbf{Sub}(\mathbf{Sents}(\mathcal{A}, t'), \mathcal{K})$ such that $\mathcal{A}'$ sees X from m' and there is no m'' such that $\mathcal{A}'$ sees X from m'' and $t^{\mathbf{Sees}}(\mathcal{A}', m'') < t^{\mathbf{Sees}}(\mathcal{A}', m')$.*

Proof: Suppose that X is given as in the lemma. By Lemma 3.2, we have that $X \in \mathbf{Sees}(\mathcal{A}', t')$ for some $t' < t$. Thus $\mathcal{A}'$ sees X from m' at time t'. Let t_0 be the least t'' satisfying that, for some m', $\mathcal{A}'$ sees X from m' at time t''.

We need only to prove that $m' \in \mathbf{Sub}(\mathbf{Sents}(\mathcal{A}, t'), \mathcal{K})$. Since $m' \in \mathbf{Sees}(\mathcal{A}', t')$, there are some $a' \in \mathcal{A}'$ and $a \in Ag \cup \{e\}$ such that $m' \in \mathbf{Sub}(m_s, \mathbf{Keys}(a', t'))$, where $m_s \in \mathbf{Sents}(a, t'')$ for some $t'' < t'$. If $a \in \mathcal{A}'$, then $m_s \in \mathbf{Poss}(\mathcal{A}', t'')$. By Lemma 3.5, we thus have that $\mathcal{A}'$ sees X from m_s at time $t'' < t'$. This is a contradiction. Therefore, $a \in \mathcal{A}$, and $m' \in \mathbf{Sub}(\mathbf{Sents}(\mathcal{A}, t'), \mathcal{K})$. ∎

The following lemma indicates that if group $\mathcal{A}'$ possesses but can not construct message $\{m_1\}_k$, then there is someone other than principals in $\mathcal{A}'$ who first reveals $\{m_1\}_k$ to group $\mathcal{A}'$.

Lemma 3.9 *Let $\mathcal{A} \subseteq Ag$, $\mathcal{A}' = (Ag - \mathcal{A}) \cup \{e\}$ and $\{m_1\}_k \in \mathbf{Poss}(\mathcal{A}', t') - \mathbf{Encr}(\mathcal{A}', t')$. Then, there is an $m_2 \in \mathbf{Sub}(\mathbf{Sents}(\mathcal{A}, t'), \mathcal{K})$ such that $\mathcal{A}'$ sees $\{m_1\}_k$ from m_2 and there is no m_3 such that $\mathcal{A}'$ sees $\{m_1\}$ from m_3 and $t^{\mathbf{Sees}}(\mathcal{A}', m_3) < t^{\mathbf{Sees}}(\mathcal{A}', m_2)$.*

Proof: Similar to the proof for Lemma 3.8. ∎

4 Instantiation Space with a Protocol

In order to give a formal soundness proof for the axioms of our protocol logic, we introduce a new semantics model: Instantiation Space. The intuition behind this model is that the message terms in a protocol are essentially message variables and for a principal to execute a particular (local) session of the protocol is just to instantiate those message variables.

4.1 Protocol and Local Protocol

To describe a protocol, we first identify five sets of symbols or strings: $\mathbf{T_A}$, $\mathbf{T_K}$, $\mathbf{T_N}$, $\mathbf{T_L}$ and $\mathbf{T_G}$. The strings in $\mathbf{T_A}$, $\mathbf{T_K}$, $\mathbf{T_N}$, $\mathbf{T_L}$ and $\mathbf{T_G}$ are called *agent terms , key terms , nonce terms, label message terms and general plain text terms*, respectively. These strings are also called *atomic message terms* or *plain text terms*. A *message term* is either an atomic message term, or of the form $[M_1, \cdots, M_n]$, where every M_i $(0 < i \leq n)$ is a message term, or of the form $\{M'\}_K$, where M' is a message term and K is a key term.

We define a security protocol $\mathcal{P}$ as a tuple $(AG, PK, SK, INIT, BODY)$, where

- AG is a set of agent terms, which are called *roles* that agents or principals play in the protocol;
- PK consists of those tuples $\langle P, K, K^{-1}\rangle$, which indicates that key term K is role P's public key and K^{-1} the corresponding private key;
- SK consists of those tuples $\langle P, Q, M\rangle$, which indicates that term M is a shared secret of P and Q. Usually, M is a key term and in this case M is the shared key of P and Q.
- $INIT$ is a function that associates each role $P \in AG$ with a set of atomic message terms; and finally,
- $BODY$ is the body of the protocol, which is a finite sequence of those tuples $\langle P, Q, M\rangle$, where P and Q are roles in AG, M is a message term constructed by the atomic message terms in $\bigcup_{P\in AG} INIT(P)$ and keys appearing in PK and SK.

For convenience, let $NONCE$ be a function such that for each role $P \in AG$, $NONCE(P) = INIT(P) \cap \mathbf{T_N}$. In other words, $NONCE$ is a function that associates each role $P \in AG$ with a set of nonce terms, which expresses nonces generated by P;

In the rest of this paper, letters P, Q, R stand for agent terms, K for key terms, N for nonce or time stamp terms and M for message terms. Notice that letter P is not used to express an agent or principal, but stands for a string in $\mathbf{T_A}$. In contrast, we use symbols ρ, τ to denote agents or principals.

We use the notation $MSG_{\mathcal{P}}$ to denote the set of all message terms appearing in protocol $\mathcal{P}$, which is defined as the smallest set of message terms such that: (1) $\bigcup_{R\in AG} INIT(R) \subseteq MSG_{\mathcal{P}}$; (2) if $\langle P, R, M\rangle$ is in $BODY$ for some $P, R \in AG$, then $M \in MSG_{\mathcal{P}}$; (3) if an *encryption* term $\{M\}_K \in MSG_{\mathcal{P}}$, then both M and K is in $MSG_{\mathcal{P}}$; and (4) if a *concatenation* term $[M_1, \ldots, M_n]$ is in $MSG_{\mathcal{P}}$, then $M_i \in MSG_{\mathcal{P}}$ for all $i = 1, \ldots, n$.

We define a role Q's local protocol, denoted by $\mathcal{P}(Q)$, as a local history of sent and received messages described by $BODY$ in the protocol $\mathcal{P}$. It can be characterized as a sequence of events $\{E_j\}_{j<l_Q}$, where l_Q is the length of Q's local protocol, and each E_j is either $send(M)$ or $recv(M)$ for some $M \in MSG_{\mathcal{P}}$ such that $\langle Q, P, M\rangle$ or $\langle R, Q, M\rangle$ is an element of the sequence $BODY$.

4.2 Some More Notations

We introduce some auxiliary notations. Given a protocol $\mathcal{P}$ and a role Q, for each $l \leq l_Q$, we consider the initial segment $\{E_j\}_{j<l}$ of $\{E_j\}_{j<l_Q}$. We denote by $Sents(Q, l)$ the set of those messages that role Q sent before step l according to protocol $\mathcal{P}$. We also introduce further notations of $Recvs(Q, l)$, $Poss(Q, l)$, $Said(Q, l)$ and $Sees(Q, l)$. Formal definitions of these notations are as follows:

- $Sents(Q, l)$: the set of those M such that $send(M) = E_j$ for some $j < l$. Clearly, if $l = 0$, then $Sents(Q, l)$ is an empty set.

- $Recvs(Q,l)$: the set of those M such that $recv(M) = E_j$ for some $j < l$. Notice that $Recvs(Q,0)$ is empty.
- $Poss(Q,l)$: the least set $\Delta \subseteq MSG_{\mathcal{P}}$ such that: (1) $INIT(Q) \cup Recvs(Q,l) \subseteq \Delta$; (2) if $[M_1,\ldots,M_n] \in \Delta$, then $M_i \in \Delta$, for $i = 1,\ldots,n$; and (3) if $\{M\}_K \in \Delta$ and $K^{-1} \in \Delta$ then $M \in \Delta$.
- $Said(Q,l)$: the least set Δ such that: (1) $Sents(Q,l) \subseteq \Delta$; (2) if $[M_1,\ldots,M_n] \in \Delta$, then $M_i \in \Delta$, for $i = 1,\ldots,n$; (3) if $\{M\}_K \in \Delta$, $K \in Poss(Q,l)$ and $M \in Poss(Q,l)$, then $M \in \Delta$.
- $Sees(Q,l)$: the least set Δ such that: (1) $Recvs(Q,l) \subseteq \Delta$; (2) if $[M_1,\ldots,M_n] \in \Delta$, then $M_i \in \Delta$, for $i = 1,\ldots,n$; (3) if $\{M\}_K \in \Delta$ and $K^{-1} \in Poss(Q,l)$, then $M \in \Delta$.
- $Recg(Q,l)$: the set of those encrypted messages $\{M\}_K \in MSG_{\mathcal{P}}$ such that $\{M,K\} \subseteq Poss(Q,l)$ or $\{\{M\}_K, K^{-1}\} \subseteq Poss(Q,l)$, and those messages $M_1 \oplus M_2$ such that $M_1 \in Poss(Q,l)$ and $M_2 \in Poss(Q,l)$.

  Intuitively, if $\{M\}_K \in Recg(Q,l)$, then role Q recognizes $\{M\}_K$ as a message encrypted with key K.

4.3 Instantiation Functions and Spaces

Imagine Alice and Bob have run an E-commence protocol; however Alice likes to deny that she has run any session of this protocol. Suppose we can obtain the log file that records her message data flow. Thus, it is important to judge whether she has run a session of this protocol based on such a log file. The notion of instantiation functions can be used for this purpose. If we find out, from Alice's log file, an instantiation function related to the protocol, then we can say that Alice has indeed run a session of the protocol.

In order to formally define instantiation functions and instantiation spaces, we assume that we are given a protocol $\mathcal{P}$ and a CME model

$$\Sigma = (\mathcal{M}, \mathcal{K}, e, Ag, \mathbf{Pk}, \mathbf{Sk}, \mathbf{Label}, \mathbf{Nonce}, \mathbf{Init}, \mathbf{Recvs}, \mathbf{Sents}),$$

which can be regarded as a collection of principals' log files of message data flows.

4.3.1 Instantiation Function

As mentioned earlier, given a principal $\rho \in Ag$ and label message c, we assume there is at most one ρ's local run of protocol $\mathcal{P}$ using message c. Therefore, we use the pair (ρ, c) to denote ρ's local run of protocol $\mathcal{P}$ with label message c (if any). For convenience, we use variables μ and ν to range over pairs of principals and label messages, and c to range over label messages.

For each local run $\mu = (\rho, c)$, we get a message value function $f : MSG_{\mathcal{P}} \to \mathcal{M}$ such that for each $M \in MSG_{\mathcal{P}}$, $f(M)$ is the message playing the role of M in local run μ of $\mathcal{P}$. The following definition of a (ρ, c, P, l)-function captures some important properties of such functions.

Definition 4.1 Let $\rho \in Ag$, c a label message, P a role and $l \leq l_P$. We say a message value function $f : MSG_{\mathcal{P}} \to \mathcal{M}$ is a (ρ, c, P, l)-function if

1. $f(M)$ is of the type indicated by M. Particularly, if $A \in AG$, then $f(A) \in Ag$; and for every label message term $T \in MSG_{\mathcal{P}}$, $f(T) \in \mathbf{Label}$.
2. (a) $f(P) = \rho$.

(b) For every $N \in NONCE(P)$, $f(N) \in \mathbf{Nonce}(\rho)$.

(c) For $M \in INIT(P)$, $f(M) \in \mathbf{Init}(\rho)$.

(d) For every $l' \leq l$ and $M \in Poss(P, l') - Recg(P, l')$, there is a t such that $f(M) \in \mathbf{Poss}(\rho, t) - \mathbf{Recg}(\rho, t)$.

3. For some label message term $M \in Poss(P, l)$, $f(M) = c$.

4. (a) $f([M_1, \ldots, M_n]) = [f(M_1), \ldots, f(M_n)]$.

   (b) $f(K^{-1})$ is the inverse key of $f(K)$.

   (c) $f(\{M\}_K) = \{f(M)\}_{f(K)}$, if $\{M\}_K \notin Poss(P, l)$ or $\{M\}_K \in Recg(P, l)$.

5. (a) There is a sequence of time points $t_0 < \cdots < t_{l-1}$ such that, for every $j < l$, if $E_j = send(M)$ for some $M \in MSG_{\mathcal{P}}$, then $f(M) \in \mathbf{Sent}(\rho, t_j)$; and if $E_j = recv(M)$ for some $M \in MSG_{\mathcal{P}}$, then $f(M) \in \mathbf{Recv}(\rho, t_j)$.

   (b) Suppose $m \in \mathbf{Fresh}(\rho, f(L), t)$, where L is a label message term in the protocol and t is a time point. We have that if $m \in \mathbf{Sent}(\rho, t)$, then there is an $E_j = send(M)$ such that $m = f(M)$; moreover, if $m \in \mathbf{Recv}(\rho, t)$, then there is an $E_j = recv(M)$ in the initial segment $\{E_j\}_{j<l}$ of local protocol $\mathcal{P}(P)$ such that $m = f(M)$.

Intuitively speaking, if there is a (ρ, c, P, l)-function, then we can say principal ρ have run a session of the protocol with role P and complete the first l steps of the session locally.

An *instantiation function* is a (ρ, c, P, l)-function for some role P and $l \leq l_P$.

For each local run $\mu = (\rho, c)$, if ρ plays role P and has finished exactly l-steps of local steps in $\mathcal{P}(P)$, then the message value function related to local run μ can be defined as a (ρ, c, P, l)-function.

Notice that if f is a (ρ, c, P, l)-function, and $f(L) = c'$ for some label message term L in the protocol, then f is also a (ρ, c', P, l)-function.

4.3.2 Instantiation Space

The notion of an instantiation space imposes some constraints on the behaviors of those principals in Ag, who may run protocol $\mathcal{P}$ with an unbounded number of local sessions.

Definition 4.2 An instantiation space is a tuple $(\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S})$,

1. Σ is a CME model.

2. $\mathcal{L}$ is a set of label messages related to protocol $\mathcal{P}$.

3. $\mathcal{F}$ is a set of instantiation functions such that

   (a) there is at most one (ρ, c, P, l)-function for given $\rho \in Ag$ and $c \in \mathcal{L}$; and

   (b) for a (ρ, c, P, l)-function f and a label message term M, we have that $f(M) \in \mathcal{L}$.

4. $\mathcal{S}$ is a set of *assumptions* about the behaviors of principals in Ag; each of them is of one of the following forms: $Supp_1^{\rho}$, $Supp_2^{\rho,Q}$, $Supp_3^{\rho,Q}$ and $Supp_4^{\rho}$.

   (a) If $Supp_1^{\rho} \in \mathcal{S}$, then for an arbitrary label message c and time t, we have that $c \in \mathbf{Said}(\rho, t)$ or $c \in \mathbf{Sees}(\rho, t)$ implies that, for some role P and number $l \leq l_P$, there is a (ρ, c, P, l)-function in $\mathcal{F}$.

(b) If $Supp_2^{\rho,P} \in \mathcal{S}$, then for an arbitrary label message c and time t, we have that $c \in \mathbf{Said}(\rho, t)$ or $c \in \mathbf{Sees}(\rho, t)$ implies that, for some number $l \leq l_P$, there is a (ρ, c, P, l)-function in $\mathcal{F}$.

(c) If $Supp_3^{\rho,P} \in \mathcal{S}$, then principal ρ should announce to another principal if there exists a ρ's uncompleted local run with role P in protocol $\mathcal{P}$. So, it is commonly known whether such a local run exists.

(d) If $Supp_4^{\rho} \in \mathcal{S}$, then if $said(\rho, \{[m_1, \cdots, \{m'\}_{k'}, \cdots, m_h]\}_k)$, where $k \in \mathbf{PrK}(\rho)$, k' is a public key or hash key, then $poss(\rho, m')$.

Due to Definition 4.2 (3a), we can use the notation $f^{\rho,c}$ to denote the (ρ, c, P, l)-function (if any). And for any $M \in MSG_{\mathcal{P}}$, we denote $f^{\rho,c}(M)$ by $\overline{M}^{\rho,c}$.

Intuitively, $Supp_1^{\rho}$ indicates that principal ρ runs protocol $\mathcal{P}$ with label message c if it says or sees label message c. And $Supp_2^{\rho,Q}$ says that principal ρ only runs protocols $\mathcal{P}$ and plays role Q in the protocol. For example, we can assume that a fixed communication channel of a bank always plays the role of *bank* in some fixed E-Commerce protocol. Assumption $Supp_3^{\rho,P}$ is helpful to build multi-level and common knowledge. Assumption $Supp_4^{\rho}$ requires that principal ρ must not sign a sequence of messages with one of them is unrecognizable to ρ. As a result, if ρ signs the hash of a message, then ρ should possess the message (otherwise, the hash of a message must be unrecognizable to ρ.)

Remark 4.3 Our requirements for instantiation functions and spaces in Definitions 4.1 and 4.2 are rational in many situations and closely related to those assumptions in many existing approaches in security protocol verifications, such as the *honesty rule* [20], faithfulness assumption [38], and the *fail-stop* property [24]. The *honesty rule* only assumes that every party follows the prescribed steps of the protocol correctly. The faithfulness assumption just assumes that participants should only proceed if they have seen the message they were to have received most recently. The *fail-stop* property indicates that a principal will stop running the protocol and automatically halt when the principal finds any abnormality or signs of attack in the run.

Remark 4.4 In our approach, runs of protocol $\mathcal{P}$ could be concurrent, and the number of runs is not limited. Moreover, we allow principals to run other protocols; nevertheless, by Definition 4.1(5b) the fresh messages in a principal's local run of a protocol should not appear again in the principal's local runs of other protocols.

5 A Logic of Local Sessions

We now propose a Logic of Local Sessions (LLS), which is a first-order theory. Since a principal's local session is uniquely determined by a label message, our object variables v, $v_0, v_1, \cdots$, and v' in LLS ranging over label messages, are called label message variables.

Below, we present a set $\Theta_{\mathcal{P}}$ of axioms for LLS.

5.1 Atomic Formulas

For convenience, let X and Y denote messages of the form $\overline{M}^{\rho,v}$ as well as principals and label message variables. The set of primitive formulas in LLS includes:

- $X = Y$: Specifically, $\overline{P}^{\rho,v} = \rho$ means that ρ plays the role of P in local run (ρ, v) of $\mathcal{P}$. We use $role(\rho, v, P)$ to denote $\overline{P}^{\rho,v} = \rho$, which indicates that ρ plays the role of P in the local run with label message c.

- $reach(\rho, v, P, l)$: ρ plays the role of P and reaches step $(l-1)$ of local run (ρ, v) of $\mathcal{P}$. In particular, $reach(\rho, v, P, 0)$ is equivalent to $role(\rho, v, P)$. Moreover, let l_P be the length of role P's local protocol of $\mathcal{P}$, $reach(\rho, v, P, l_P)$ indicates that ρ completes role P in local run (ρ, v) of $\mathcal{P}$. For convenience, we use $rch(\rho, v, P, l)$ to denote $reach(\rho, v, P, l) \wedge \neg reach(\rho, v, P, l+1)$.

- $norm(\rho, v, P)$: If ρ plays the role of P in local run (ρ, v) of protocol $\mathcal{P}$, then he/she reaches the end. We note that $norm(\rho, v, P)$ is logically equivalent to formula $role(\rho, v, P) \Rightarrow reach(\rho, v, P, l_P)$; nevertheless, it is observable to all principals in case of $Supp_3^{\rho,P} \in \mathcal{S}$.

- $poss(\rho, X)$: ρ possesses X or ρ has X.

- $said(\rho, X)$: ρ has said X by sending some message containing X.

- $sees(\rho, X)$: ρ can get X from a received message.

- $fresh(\rho, v, X)$: $fresh(\rho, v, X)$ means that ρ is able to check that message X belongs to a (local) session (ρ, v) by checking the value of a label message.

- $secr(\rho_1, \cdots, \rho_n, X)$: X is a secrecy among principals $\rho_1, \cdots, \rho_n \in Ag$.

Because there are only finitely many principals and message terms, the above primitive formulas can be regarded as a finite set of unary and binary predicates on label messages. For example, $\overline{P}^{\rho,v} = \rho$ represents an unary predicate (on v), and $\overline{P}^{\rho,v} = \overline{P}^{\tau,v'}$ represents a binary predicate (on v and v').

We now define the semantics of the primitive formulas in LLS with respect to an instantiation space $\Omega = (\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S})$. We use $\Omega \models \psi$ to denote the satisfaction relation that ψ is true with respect to model Ω [3].

- $\Omega \models X = Y$ iff X and Y represent the same message in $\mathcal{M}$ and, in the case X or Y is of form $\overline{M}^{\rho,v}$, there is a (ρ, v, P, l)-function in $\mathcal{F}$ for some P and l.

- $\Omega \models reach(\rho, v, P, l)$ iff there is a (ρ, v, P, l)-function in $\mathcal{F}$.

- $\Omega \models norm(\rho, v, P)$ iff there is no (ρ, v, P, l)-function in $\mathcal{F}$ with $l < l_P$.

- $\Omega \models poss(\rho, X)$ iff for some time t, $X \in \mathbf{Poss}(\rho, t)$.

- $\Omega \models sees(\rho, X)$ iff for some time t, $X \in \mathbf{Sees}(\rho, t)$.

- $\Omega \models said(\rho, X)$ iff for some time t, $X \in \mathbf{Said}(\rho, t)$.

- $\Omega \models fresh(\rho, v, X)$ iff for some time t, $X \in \mathbf{Fresh}(\rho, f(N), t)$, where f is a (ρ, v, P, l)-function in $\mathcal{F}$ for some role P and $l < l_P$, and N is a label message term in the protocol.

- $\Omega \models secr(\rho_1, \cdots, \rho_n, X)$ iff there is no $\rho' \in Ag \cup \{e\} - \{\rho_1, \cdots, \rho_n\}$ such that for some time t, $X \in \mathbf{Poss}(\rho', t)$.

[3] For convenience, we omit the assignment function for label message variables v's, not distinguishing v's themselves from those label message values they represent.

5.2 Some Important Axioms

Given $\mathcal{S}$, we use $\models_{\mathcal{S}} \psi$ (or $\models \psi$ if $\mathcal{S}$ is not emphasized) to denote that $(\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S}) \models \psi$ holds for all Σ, $\mathcal{L}$ and $\mathcal{F}$ such that $(\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S})$ forms an instantiation space. We say a property ψ of protocol is valid if $\models_{\mathcal{S}} \psi$.

Proposition 5.1 (Label Message Existence)

$$\models \forall v \exists v' \left(\bigvee_{P \in AG,\ N \in Poss(P, l_P)} role(\rho, v, P) \Rightarrow (\overline{N}^{\rho, v} = v') \right)$$

where N is a label message term.

To remove the quantifiers in the above formulas, we introduce Skolem functions g_ρ^N in LLS, and have that

$$\models \bigvee_{P \in AG,\ N \in Poss(P, l_P)} role(\rho, v, P) \Rightarrow (\overline{N}^{\rho, v} = g_\rho^N(v))$$

Proposition 5.2 (Role Assumption)

1. *If $Supp_1^\tau \in \mathcal{S}$, then* $\models_{\mathcal{S}} (said(\tau, v) \vee sees(\tau, v)) \Rightarrow \bigvee_{Q \in AG} role(\tau, v, Q)$.
2. *If $Supp_2^{\tau, Q} \in \mathcal{S}$, then* $\models_{\mathcal{S}} (said(\tau, v) \vee sees(\tau, v)) \Rightarrow role(\tau, v, Q)$.

Proof: Straightforward by the definition of the assumptions $Supp_1^\tau$ and $Supp_2^{\tau, Q}$. ∎

We notice that, if both of the assumptions $Supp_1^\tau$ and $Supp_2^{\tau, Q}$ do not hold in the instantiation space, we may add a role assumption of the form $RA_{\mathcal{P}}^{\tau, c, Q} \Rightarrow role(\tau, c, Q)$, where the formula $RA_{\mathcal{P}}^{\tau, c, Q}$ indicates the evidence that principal τ intends to play the role of Q in local run (τ, c) of protocol $\mathcal{P}$. However, we can not give the formula $RA_{\mathcal{P}}^{\tau, c, Q}$ automatically from the protocol; we can usually obtain such a formula by hand before analyzing a protocol according to the local history of an agent's behaviors in the protocol as done by Thayer *et al* [39].

Proposition 5.3 (Signature Properties:) *Given P, ρ, Q, τ, c, l. Suppose K is a public key or hash term, $M_i = \{M'\}_K \in Recg(Q, l)$ and $M = \{[M_1, \cdots, M_i, \cdots, M_h]\}_{K_P^{-1}} \in Poss(Q, l)$. Then, if $supp_4^\rho \in \mathcal{S}$, we have that*

$$\models reach(\tau, c, l) \wedge \overline{P}^{\tau, c} = \rho \wedge said(\rho, \overline{M}^{\tau, c}) \Rightarrow poss(\rho, \overline{M'}^{\tau, c})$$

Proof: Straightforward by the definition of the assumptions $Supp_4^\rho$. ∎

Proposition 5.4 (Secrecy Derivation:) *Given P_i, ρ_i, and l_i $(i = 0, \ldots, n)$. Suppose $M \in Poss(P_0, l_0)$. Then, we have that*

$$\models \begin{array}{l} reach(\rho_0, v_0, P_0, l_0) \wedge \bigwedge_{0 < i \leq n} rch(\rho_i, v_i, P_i, l_i) \Rightarrow \\ secr(\rho_1, \cdots, \rho_n, \overline{M}^{\rho_0, v_0}) \vee GS(P_0, \rho_0, v_0, P_1, \rho_1, v_1, l_1 \ldots, P_n, \rho_n, v_n, l_n, M) \end{array}$$

where $GS(P_0, \rho_0, v_0, P_1, \rho_1, v_1, l_1 \ldots, P_n, \rho_n, v_n, l_n, M)$ (or $GS(M)$ for short) is a formula, which characterizes that that someone other than ρ_i $(i = 1, \ldots, n)$ can possess $\overline{M}^{\rho_0, v_0}$.

The precise definition of $GS(M)$ and the proof of the above proposition are given by induction on M in the next subsection.

Lemma 5.5 *Let $M \in Recvs(P_0, l_0)$. Then, for $\rho_1, \cdots, \rho_n \in Ag$, we have that*

$$\models reach(\rho_0, v_0, P_0, l_0) \Rightarrow \neg secr(\rho_1, \cdots, \rho_n, \overline{M}^{\rho_0, v_0}).$$

Proof: If $\Omega \models reach(\rho_0, v_0, P_0, l_0)$ and $M \in Recvs(P_0, l_0)$ then ρ_0 must receive $\overline{M}^{\rho_0, v_0}$ and and enthronement e possesses $\overline{M}^{\rho_0, v_0}$. Thus $\Omega \models secr(\rho_1, \cdots, \rho_n, \overline{M}^{\rho_0, v_0}$. ∎

Proposition 5.6 ($(n,1)$-Secrecy Properties) *Given P_i, ρ_i, and l_i $(i = 0, \ldots, n)$. And $M \in Recvs(P_0, l_{P_0})$ and l_0 be the least l such that $M \in Recvs(P_0, l)$. We have the following:*

$$\models \; reach(\rho_0, v_0, P_0, l_0) \wedge \bigwedge_{0<i\leq n} rch(\rho_i, v_i, P_i, l_i) \Rightarrow GS(P_0, \rho_0, v_0, P_1, \rho_1, v_1, l_1 \ldots, P_n, \rho_n, v_n, l_n, M)$$

Proof: By Proposition 5.4 and Lemma 5.5.

When $\rho_0 = \rho_1$, $v_0 = v_1$ and $P_0 = P_1$, the resulting properties are called $(n, 0)$-*Secrecy Properties*.

The properties above are very significant because they describe exactly how a principal's knowledge is gained when the principal receives or sees a message which is originated by a group of n-principals.

Remark 5.7 A very special case of the Secrecy Properties is $(1, 0)$-*Secrecy Properties*, which corresponds to the *challenge-response axiom schema* in Cervesato *et al* [14] and the predicate `Source` in compositional logic [20]. However, our property is well defined by the structural complexity of messages and can be easily translated into concrete algorithms to compute the exact evolution of a principal's knowledge when the principal receives a message. In contrast, the *challenge-response axiom schema* [14] does not present us a concrete algorithm to yield axioms. The predicate `Source` [20] requires that the challenge has not been sent twice before the principal receives it. $(1, 0)$-Secrecy Properties presented here have no such requirements. Consider the case that principal ρ created a nonce c, encrypted it with principal τ_1 and τ_2's public key k_1 and k_2, respectively, and sent out the resulting two messages $\{c\}_{k_1}$ and $\{c\}_{k_2}$ one by one. By the presented properties, when seeing nonce c, principal ρ can conclude that $said(\tau_1, c) \vee said(\tau_2, c)$ holds.

Remark 5.8 $(1, 1)$-*Secrecy Properties* are beyond the *challenge-response axiom schema* in Cervesato *et al* [14] in that the condition of *challenge-response axiom schema* is that a principal said a challenge and herself sees the challenge, while $(1, 1)$-Secrecy Properties only require that a principal said a challenge and someone (possibly other than the principal) sees the challenge. The general $(n, 1)$-Secrecy Properties are even more general than the *send-receive axiom schema* [14] because the latter concerns only two principals.

Appendix A lists a set $\Theta_{\mathcal{P}}$ of formulas in LLS, which are proved to be valid and we take as axioms for our logic. For convenience, let $\Theta_{\mathcal{P}}(v_0, \cdots, v_k)$ denote the conjunction of all those axioms. We divide those valid properties in set $\Theta_{\mathcal{P}}(v_0, \cdots, v_k)$ into three parts: routine ones , kernel ones, and equivalence and comparison ones. The first part contains some routine properties on system variables; their justifications and validity proofs are straightforward. The second part of properties are very essential and useful; however, their justifications need to be demonstrated, and we provide the soundness proof in Section 5.3. Most properties in the last part hold trivially, but SPV should deal with them very subtly, because the number of such properties is surprisingly large.

5.3 Soundness Proof of $(n, 1)$-Secrecy Properties

In this section, we provide details of function GS in Proposition 5.4 and $(n, 1)$-Secrecy Properties and show their validity. We fix an instantiation space $\Omega = (\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S})$. We first define formula $R^i(M'', X)$, by induction on the complexity of M''. Intuitively, $R^i(M, X)$ means that principal ρ_i reveals X to someone other than $\rho_1, \cdots, \rho_n$ via saying $\overline{M}^{\rho_i, c_i}$.

Definition 5.9 Given ρ_i, P_i, c_i and $l_i \leq l_{P_i}$ for $i = 1, \ldots, n$. Let $R^i(M'', X) = (\overline{M''}^{\rho_i, c_i} = X) \vee \psi$, where ψ is defined by induction on M'' as follows.

1. $M'' = [M_1, \ldots, M_n]$. We have that $\psi = \bigvee_{1 \leq j \leq n} R^i(M_j, X)$.
2. $M'' = \{M_1\}_k$ and $M'' \notin Recg(P_i, l_i)$.

$$\psi = \bigvee_{\substack{i' \neq i \\ 0 < i' \leq n}} \left(\bigvee_{M_1'' \in Said(P_{i'}, l_{i'}) \cap Recg(P_{i'}, l_{i'})} \varphi_{i', M_1''} \right)$$

where

$$\varphi_{i', M_1''} = (\overline{M''}^{\rho_i, c_i} = \overline{M_1''}^{\rho_{i'}, c_{i'}}) \wedge R^{i'}(M_1'', X)$$

3. $M'' = \{M_1\}_K$ and $M'' \in Recg(P_i, l_{P_i})$. There are four subcases:
   (a) i. If K is the public key of $P_{i'}$, or the shared key of $P_{i'}$ with P_i, $(i' = 1, \ldots, n)$, $\psi = \neg(\overline{P_{i'}}^{\rho_i, c_i} = \rho_{i'})$.
   ii. If K is the hash key, then $\psi = false$.
   (b) If K is a private key, then $\psi = R^i(M_1, X)$.
   (c) If K is the public key of Q or the shared key of Q and P_i, where Q differs from $P_{i'}$ $(i' = 1, \ldots, n)$, then ψ equals

$$\left(\bigwedge_{i'=1,\ldots,n} (\overline{Q}^{\rho_i, c_i} \neq \rho_{i'}) \right) \wedge R^i(M_1, X) \wedge \bigwedge_{\rho' \in \mathcal{A}} \left((\overline{Q}^{\rho_i, c_i} = \rho') \Rightarrow sees(\rho', \overline{M''}^{\rho_i, c_i}) \right).$$

   (d) If K is a session key, then ψ equals

$$R^i(M_1, X) \wedge \bigwedge_{\rho_1', \rho_2' \in \mathcal{A}} \left(secr(\rho_1', \rho_2', \rho_i, \overline{K}^{\rho_i, c_i}) \Rightarrow \left(\begin{array}{l} \bigwedge_{i'=1,\ldots,n} (\rho_1' \neq \rho_{i'}) \wedge sees(\rho_1', \overline{M''}^{\rho_i, c_i}) \vee \\ \bigwedge_{i'=1,\ldots,n} (\rho_2' \neq \rho_{i'}) \wedge sees(\rho_2', \overline{M''}^{\rho_i, c_i}) \end{array} \right) \right).$$

4. Otherwise, $R^i(M'', X) = false$.

Given P_i, ρ_i, c_i and l_i $(i = 1, \ldots, n)$, let $TR(P_1, \rho_1, c_1, P_2, \cdots, P_n, \rho_n, c_n, X)$ (or $TR(X)$ for short) be the following formula:

$$\bigwedge_{i=1,\cdots,n} fresh(\rho_i, c_i, X) \Rightarrow \left(\bigvee_{i=1,\ldots,n, M'' \in \Delta_i} R^i(M'', X) \right)$$

where $\Delta_i = Sents(P_i, l_i)$.

The following definition give formulas $GS(P_0, \rho_0, c_0, P_1, \rho_1, c_1, l_1, \ldots, P_n, \rho_n, c_n, l_n, M)$, which means that someone other that $\rho_1, \cdots, \rho_n$ can possess $\overline{M}^{\rho_0, c_0}$.

Definition 5.10 Given P_i, ρ_i, c_i and $l_i (i = 0, \ldots, n)$. Suppose $M \in Poss(P_0, l_{P_0})$. Let $GS(P_0, \rho_0, c_0, P_1, \rho_1, c_1, l_1, \ldots, P_n, \rho_n, c_n, l_n, M)$ (or $GS(M)$ for short) be a formula function. We give the definition of $GS(M)$ by induction on M as follows:

1. $M = [M_1, \ldots, M_h]$: $GS(M)$ equals $\bigwedge_{1 \le i \le h} GS(M_i)$.
2. $M = \{M'\}_K$: There are several cases.
   (a) If K is a public key or hash key and $M \in Recg(P_0, l_{P_0})$, then $GS(M) = TR(\overline{M}^{\rho_0, c_0}) \vee GS(M')$.
   (b) If K is the private key of R, then there are two subcases:
      i. $GS(M) = GS(M') \wedge \bigwedge_{\tau \in \mathcal{A}}((\overline{R}^{\rho_0, c_0} = \tau) \Rightarrow said(\tau, \overline{M}^{\rho_0, c_0}))$, if R differs from P_i $(i = 1, \ldots, n)$.
      ii. $GS(M) = (\overline{P_i}^{\rho_0, c_0} = \rho_i) \Rightarrow TR(\overline{M}^{\rho_0, c_0}) \wedge said(\rho_i, \overline{M}^{\rho_0, c_0})$, if K is one of private keys of P_i's.
   (c) If K is a shared key between P_0 and R, there are four subcases (depend on whether P_0 or R is one of P_i's):
      i. $GS(M) = TR(\overline{M}^{\rho_0, c_0}) \wedge (said(\rho_0, \overline{M}^{\rho_0, c_0}) \vee ((\overline{P_i}^{\rho_0, c_0} = \rho_i) \Rightarrow said(\rho_i, \overline{M}^{\rho_0, c_0})))$, if ρ_0 is one of ρ_i's $(0 < i \le n)$ and K is one of the shared keys between P_0 and P_i $(0 < i \le n)$.
      ii. $GS(M) = (TR(\overline{M}^{\rho_0, c_0}) \wedge said(\rho_0, \overline{M}^{\rho_0, c_0})) \vee (GS(M') \wedge \bigwedge_{\tau \in \mathcal{A}}((\overline{R}^{\rho_0, c_0} = \tau) \Rightarrow said(\tau, \overline{M}^{\rho_0, c_0})))$, if K is one of the shared keys between P_0 and R, where P_0 is one of P_i's $(0 < i \le n)$ and R differ from P_i $(0 < i \le n)$.
      iii. $GS(M) = (GS(M') \wedge said(\rho_0, \overline{M}^{\rho_0, c_0})) \vee ((\overline{P_i}^{\rho_0, c_0} = \rho_i) \Rightarrow TR(\overline{M}^{\rho_0, c_0}) \wedge said(\rho_i, \overline{M}^{\rho_0, c_0})))$, if ρ_0 differs from ρ_i's $(0 < i \le n)$ and K is one of the shared keys between P_0 and P_i $(0 < i \le n)$.
      iv. $GS(M) = GS(M') \wedge ((said(\rho_0, \overline{M}^{\rho_0, c_0}) \vee \bigwedge_{\tau \in \mathcal{A}}((\overline{R}^{\rho_0, c_0} = \tau) \Rightarrow said(\tau, \overline{M}^{\rho_0, c_0})))$, if K is one of the shared keys between P_0 and R, where P_0 and R differ from P_i $(0 < i \le n)$.
   (d) If K is a session key and $M \in Recg(P_0, l_0)$, there are two cases:
      i. $GS(M) = TR(\overline{M}^{\rho_0, c_0}) \vee (((\overline{K}^{\rho_0, c_0} = \overline{K}^{\rho_i, c_i}) \Rightarrow TR(\overline{K}^{\rho_0, c_0})) \wedge GS(M'))$, if K is a key created by P_i for some $i = 1, \ldots, n$;
      ii. otherwise, $GS(M) = TR(\overline{M}^{\rho_0, c_0}) \vee GS(M')$
3. $M = N$: Here N is a nonce or other message (say a key) created by P_i for some i $(i = 1, \ldots, n)$. In this case, let $GS(M) = (\overline{N}^{\rho_0, c_0} = \overline{N}^{\rho_i, c_i}) \Rightarrow TR(\overline{N}^{\rho_0, c_0})$.
4. M is a shared secret of P_i and $P_{i'}$, where $(0 < i, i' \le n)$. In this case, let $GS(M) = (\overline{M}^{\rho_0, c_0} = \overline{M}^{\rho_i, c_i}) \Rightarrow TR(\overline{M}^{\rho_0, c_0})$.
5. Otherwise, $GS(M) = true$.

We now give a series of lemmas. The informal idea of following lemmas is that, if X can not be constructed by principals other than $\rho_1, \cdots, \rho_n$ and it is first via $\overline{M''}^{\rho_i, c_i}$ that principals other than $\rho_1, \cdots, \rho_n$ can see X, then $R^i(M, X)$ holds.

Lemma 5.11 *Given $\mathcal{A} = \{\rho_1, \cdots, \rho_n\} \subseteq Ag$, $\mathcal{A}' = Ag \cup \{e\} - \mathcal{A}$. Suppose for $i = 1, \cdots, n$, $\Omega \models fresh(\rho_i, c_i, X)$ and $\Omega \models reach(\rho_i, c_i, P_i, l_i) \wedge \neg reach(\rho_i, c_i, P_i, l_i + 1)$. Let $M'' \in Said(P_i, l_i)$ and X be a message in $\mathcal{M}$ such that*

1. *$X \in \mathbf{Nonce}(\mathcal{A})$; or*

2. *X is a shared secret of two principals in $\mathcal{A}$; or*

3. *X is of the form $\{m'\}_K$, where m' or K is not in $\mathbf{Poss}(\mathcal{A}', t')$ (i.e., $X \notin \mathbf{Encr}(\mathcal{A}', t')$).*

Then $\Omega \models R^i(M'', X)$ holds under the following conditions:

1. *$\mathcal{A}'$ sees X from $\overline{M''}^{\rho_i, c_i}$.*

2. *There is no m' such that $\mathcal{A}'$ sees X from m' with $t^{\mathbf{Sees}}(\mathcal{A}', m') < t^{\mathbf{Sees}}(\mathcal{A}', \overline{M''}^{\rho_i, c_i})$.*

Proof: We prove the lemma by induction on pair (M'', i) with the order $\prec$ such that $(M_1'', i_1) \prec (M_2'', i_2)$ iff

1. $\overline{M_1''}^{\rho_{i_1}, c_1} = \overline{M_2''}^{\rho_{i_2}, c_2}$, and $M_1'' \in Recg(P_{i_1}, l_{i_1})$ but $M_2'' \notin Recg(P_{i_2}, l_{i_2})$, or

2. $i_1 = i_2$, $M_1'' \in Recg(P_{i_1}, l_{i_1})$ and $M_2'' \in Recg(P_{i_2}, l_{i_2})$, but M_1'' is a submessage of M_2''.

Suppose that the conclusion hold for all $(M', i') \prec (M'', i)$. We consider the case of (M'', i) and assume the two conditions in this lemma holds. We want to show $R^i(M'', X)$ holds. There are four cases:

1. M'' is not a concatenation or encryption term. In this case, since the first condition implies $\overline{M''}^{\rho_i, c_i} = X$. Thus, $\Omega \models R^i(M'', X)$, which is equivalent to $\Omega \models \overline{M''}^{\rho_i, c_i} = X$, must be true.

2. $M'' = [M_1, \cdots, M_h]$. Since M'' satisfies the second condition of this lemma, we have that all M_j's $(0 < j \leq h)$ must do due to $t^{\mathbf{Said}}(\mathcal{A}', \overline{M_j}^{\rho_i, c_i}) \leq t^{\mathbf{Said}}(\mathcal{A}', \overline{M''}^{\rho_i, c_i})$ for every $1 \leq j \leq m$. On the other hand, since M'' satisfies the first condition of this lemma, at least one of them satisfies the first one if $(\overline{M''}^{\rho_i, c_i} \neq X)$. So, at least one of M_j's $(0 < j \leq m)$ satisfies the two conditions if $(\overline{M''}^{\rho_i, c_i} \neq X)$. We thus have by induction assumption, at least for one j, $\Omega \models R^i(M_j, X)$ holds if $(\overline{M''}^{\rho_i, c_i} \neq X)$. Consequently, $R^i(M'', X) = (\overline{M''}^{\rho_i, c_i} = X) \vee \bigvee_{1 \leq j \leq m} R^i(M_j, X)$ is satisfied by Ω.

3. $M'' = \{M_1\}_K$ and $M'' \notin Recg(P_i, l_i)$. By the condition $\Omega \models reach(\rho_i, P_i, c_i, l_i) \wedge \neg reach(\rho_i, P_i, c_i, l_i + 1)$, we conclude that f^{ρ_i, c_i} is a (ρ_i, c_i, P_i, l_i)-function. By $M'' \notin Recg(P_i, l_i)$, and $M'' \in Said(P_i, l_i)$, we have that $M'' \in Poss(P_i, l_i) - Recg(P_i, l_i)$. Thus, by the definition of message exchange model with a protocol, there is some t, $\overline{M''}^{\rho_i, c_i} \in \mathbf{Poss}(\rho_i, t+1) - \mathbf{Poss}(\rho_i, t)$ and $\overline{M''}^{\rho_i, c_i}$ is of the form $\{m_0\}_{k_0}$ with $k_0 \notin \mathbf{Poss}(\rho_i, t+1)$ or $m_0 \notin \mathbf{Poss}(\rho_i, t+1)$. By Lemma 3.2, $\overline{M''}^{\rho_i, c_i} \in \mathbf{Sees}(\rho_i, t+1)$.

   We first note that there are some $a \in Ag \cup \{e\}$ and $t' \leq t$, $\overline{M''}^{\rho_i, c_i} \in \mathbf{Said}(a, t') \cap \mathbf{Recg}(a, t')$ by Lemma 3.4. We further have that $a \in \mathcal{A}$; otherwise, $\mathcal{A}'$ sees X from $\overline{M''}^{\rho_i, c_i}$ at time t', which contradicts the second condition of this lemma. Thus, $\overline{M''}^{\rho_i, c_i} \in \mathbf{Said}(\rho_{i'}, t') \cap \mathbf{Recg}(\rho_{i'}, t')$.

   Because X is fresh in local session $(\rho_{i'}, c_{i'})$, so is $\overline{M''}^{\rho_i, c_i}$. Since $\rho_{i'}$ plays role $P_{i'}$ in the local session $(\rho_{i'}, c_{i'})$, all fresh messages that $\rho_{i'}$ said are those $\overline{M_1''}^{\rho_{i'}, c_{i'}}$ where $M_1'' \in Said(P_{i'}, l_{P_{i'}})$. Thus, we have that $\overline{M''}^{\rho_i, c_i} = \overline{M_1''}^{\rho_{i'}, c_{i'}}$ for some $M_1'' \in Said(P_{i'}, l_{P_{i'}})$. Moreover, because $\overline{M_1''}^{\rho_{i'}, c_{i'}} \in \mathbf{Recg}(\rho_{i'}, t')$, we conclude that $M_1'' \in Recg(P_{i'}, l_{P_{i'}})$.

   The two conditions hold for M_1'' and i' because $\overline{M''}^{\rho_i, c_i} = \overline{M_1''}^{\rho_{i'}, c_{i'}}$. Moreover, we have that $(M_1'', i') \prec (M'', i)$. Thus, by inductive assumption, we have $R^{i'}(M_1'', X)$. By the discussion above, we have that ψ holds.

4. $M'' = \{M_1\}_K$ and $M'' \in Recg(P_i, l_{P_i})$. We suppose that $\overline{M''}^{\rho_i, c_i} \neq X$; otherwise, $R^{i'}(M'', X)$ holds trivially.

   There are three subcases:

(a) If K is the public key of $P_{i'}$, or the shared key of $P_{i'}$ with P_i, $(i' = 1, \ldots, n)$, then the first condition of the lemma implies that $\neg(\overline{P_{i'}}^{\rho_i, c_i} = \rho_{i'})$. So, $R^i(M'', X)$ holds in this subcase.

(b) If K is the private key of Q, then the two conditions of the lemma hold for M_1. So, by inductive assumption again, we have $R^i(M_1, X)$.

(c) If K is the public key of Q or the shared key of Q and P_i, where Q differs from $P_{i'}$ $(i' = 1, \ldots, n)$. First, the first condition of the lemma implies that $(\overline{Q}^{\rho_i, c_i} \neq \rho_{i'})$ holds for all $i' = 1, \cdots, n$. Secondly, we have $R^i(M_1, X)$ because the two conditions hold for M_1. Finally, by the condition that $\mathcal{A}'$ sees X from $\overline{M''}^{\rho_i, c_i}$, we have that ρ' sees message $\overline{M''}^{\rho_i, c_i}$ if $\overline{Q}^{\rho_i, c_i} = \rho'$. Thus, we have that $R^i(M'', X)$ holds by the definition of $R^i(M'', X)$. ∎

By the above lemma, we obtain another lemma, which gives the meaning of TR formulas.

Lemma 5.12 *Given $\mathcal{A} = \{\rho_1, \cdots, \rho_n\} \subseteq Ag$, $\mathcal{A}' = Ag \cup \{e\} - \mathcal{A}$. Suppose $X \in \mathbf{Poss}(\mathcal{A}', t')$ and for $i = 1, \cdots, n$, $\Omega \models reach(\rho_i, c_i, P_i, l_i) \wedge \neg reach(\rho_i, c_i, P_i, l_i + 1)$. Then $\Omega \models TR(X)$ if*

1. *$X \in \mathbf{Nonce}(\mathcal{A})$; or*
2. *X is a shared secret of two principals in $\mathcal{A}$; or*
3. *X is of the form $\{m'\}_K$, where m' or K is not in $\mathbf{Poss}(\mathcal{A}', t')$ (i.e., $X \notin \mathbf{Encr}(\mathcal{A}', t')$)*

Proof: Suppose $X \in \mathbf{Poss}(\mathcal{A}', t')$ and $X \in \mathbf{Nonce}(\mathcal{A})$ or X is of the form $\{m'\}_k$ where m' or k is not in $\mathbf{Poss}(\mathcal{A}', t')$. By Lemmas 3.8 and 3.9, we conclude that there is an $m' \in \mathbf{Sub}(\mathbf{Sents}(\mathcal{A}, t'))$ such that $\mathcal{A}'$ sees X from m' and there is no m'' such that $\mathcal{A}'$ sees X from m'' and $t^{\mathbf{Sees}}(\mathcal{A}', m'') < t^{\mathbf{Sees}}(\mathcal{A}', m')$. Thus, we get $\Omega \models TR(X)$ immediately by Lemma 5.11. ∎

The next lemma is the main one for the Proof of Proposition 5.4.

Lemma 5.13 *Given $\rho_0 \in Ag$ and $\mathcal{A} = \{\rho_1, \cdots, \rho_n\} \subseteq Ag$, $\mathcal{A}' = Ag \cup \{e\} - \mathcal{A}$. Suppose for $i = 0, \cdots, n$, $\Omega \models reach(\rho_i, c_i, P_i, l_i) \wedge \neg reach(\rho_i, c_i, P_i, l_i + 1)$. Let $M_0 \in Poss(P_0, l_0)$, M be a submessage of M_0, and $t' = t^{\mathbf{Recvs}}(\rho_0, \overline{M_0}^{\rho_0, c_0})$. Then $\Omega \models secr(\rho_1, \cdots, \rho_n, \overline{M}^{\rho_0, c_0}) \vee GS(M)$ if $\overline{M}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$.*

Proof: Suppose $\Omega \not\models secr(\rho_1, \cdots, \rho_n, \overline{M}^{\rho_0, c_0})$, then someone other than $\rho_1, \cdots, \rho_n$ possesses $\overline{M}^{\rho_0, c_0}$.

We will show, by induction on the structure of M, $GS(M)$ holds if someone other than $\rho_1, \cdots, \rho_n$ possesses $\overline{M}^{\rho_0, c_0}$. Suppose that the conclusion of the claim holds for all submessages of M, and assume that $\overline{M}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. We want to show that $GS(M)$ holds. According the definition of GS, there are four cases:

1. $M = [M_1, \ldots, M_m]$: In this case, $\overline{M_j}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$ for all $(j = 1, \cdots, m)$. By inductive assumption, we have $GS(M_j)$. So, $GS(M) = \bigwedge_{1 \leq j \leq m} GS(M_j)$ holds.
2. $M = \{M'\}_K$: We consider the following four cases depending on the type of key term K.
   (a) K is a public key or hash key and $M \in Recg(P_0, l_{P_0})$. In this case, if $\overline{M'}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$, than $\Omega \models GS(M')$. Otherwise, by Lemma 5.12, we have that $\Omega \models TR(\overline{M}^{\rho_0, c_0})$ holds. Therefore, we have that $GS(M) = TR(\overline{M}^{\rho_0, c_0}) \vee GS(M')$ holds in Ω.
   (b) K is the private key of R. We consider two subcases:
      i. R differs from P_i $(i = 1, \ldots, n)$: In this case, principal $\overline{R}^{\rho_0, c_0}$ said $\overline{M}^{\rho_0, c_0}$. Moreover, we have $\overline{M'}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$ by $\overline{M}^{\rho_0, c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. Thus $\Omega \models GS(M')$ holds by inductive assumption. Therefore, we have that $GS(M) = \bigwedge_{\tau \in \mathcal{A}}((\overline{R}^{\rho_0, c_0} = \tau) \Rightarrow said(\tau, \overline{M}^{\rho_0, c_0})) \wedge GS(M')$ holds in Ω.

ii. K is a private keys of P_i: In this subcase, $GS(M) = (\overline{P_i}^{\rho_0,c_0} = \rho_i) \Rightarrow (TR(\overline{M}^{\rho_0,c_0}) \wedge said(\rho_i, \overline{M}^{\rho_0,c_0}))$. In this case, if $\Omega \models (\overline{P_i}^{\rho_0,c_0} = \rho_i)$, then we have that $\Omega \models TR(\overline{M}^{\rho_0,c_0})$ by Lemma 5.12. We can get also $\Omega \models said(\rho_i, \overline{M}^{\rho_0,c_0})$ by $\overline{M}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. So, we have $\Omega \models GS(M)$.

(c) K is a shared key term between P_0 and R: There are four subcases:

i. Both P_0 and R are among of P_i's: This subcase is the same as subcase 2(b)ii.

ii. P_0 is one of P_i's and R differs from P_i's ($0 < i \leq n$): Then, either ρ_0 or principal $\overline{P_i}^{\rho_0,c_0}$ said $\overline{M}^{\rho_0,c_0}$. In the former case, we have that $\Omega \models TR(\overline{M}^{\rho_0,c_0})$ by Lemma 5.12. In the latter case, we have $\overline{M'}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$ by $\overline{M}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. Thus $\Omega \models GS(M')$ holds by inductive assumption. Therefore, we have that $\Omega \models GS(M)$ holds in Ω.

iii. P_0 differs from P_i's ($0 < i \leq n$), but R is one of them: Then, either ρ_0 or principal $\overline{P_i}^{\rho_0,c_0}$ said $\overline{M}^{\rho_0,c_0}$. In the former case, we have $\overline{M'}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$ by $\overline{M}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. Thus $\Omega \models GS(M')$ holds by inductive assumption. In the latter case, we have that if $\overline{P_i}^{\rho_0,c_0} = \rho_i$, then $\Omega \models TR(\overline{M}^{\rho_0,c_0})$ by Lemma 5.12. Therefore, we have that $\Omega \models GS(M)$ holds in Ω.

iv. Both P_0 and R differ from P_i's ($0 < i \leq n$): Then, either ρ_0 or principal $\overline{P_i}^{\rho_0,c_0}$ said $\overline{M}^{\rho_0,c_0}$. Moreover, we have $\overline{M'}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$ by $\overline{M}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$. Thus $\Omega \models GS(M')$ holds by inductive assumption. Therefore, we have that $\Omega \models GS(M)$ holds in Ω.

(d) K is a session key. We consider two subcases.

i. $K \in NONCE(P_i)$ for some $i = 1, \ldots, n$. In this subcase, if $\overline{M'}^{\rho_0,c_0} \notin \mathbf{Poss}(\mathcal{A}', t')$ or $\overline{K}^{\rho_0,c_0} \notin \mathbf{Poss}(\mathcal{A}', t')$, then we have that $\Omega \models TR(\overline{M}^{\rho_0,c_0})$ by Lemma 5.12. If $\overline{M'}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$, then $\Omega \models GS(M')$ by the inductive assumption. If $\overline{K}^{\rho_0,c_0} \in \mathbf{Poss}(\mathcal{A}', t')$, then $\Omega \models (\overline{K}^{\rho_0,c_0} = \overline{K}^{\rho_i,c_i}) \Rightarrow TR(\overline{K}^{\rho_0,c_0})$ by Lemma 5.12. So, we have that $\Omega \models GS(M)$ holds.

ii. $K \in NONCE(R)$, where R differs from P_i's. In this subcase, we get that $\Omega \models GS(M)$ holds in the same way as in case 2a.

3. $M = N$: Here N is a nonce or other message (say a key) created by P_i for some i ($i = 1, \ldots, n$). In this case, if $(\overline{N}^{\rho_0,c_0} = \overline{N}^{\rho_i,c_i})$, then $\overline{N}^{\rho_0,c_0} \in \mathbf{Nonce}(\rho_i)$ and by Lemma 5.12, we get that $\Omega \models TR(\overline{N}^{\rho_0,c_0})$. Thus, $GS(M) = (\overline{N}^{\rho_0,c_0} = \overline{N}^{\rho_i,c_i}) \Rightarrow TR(\overline{N}^{\rho_0,c_0})$ holds in Ω.

4. M is a shared secret of two P_i and $P_{i'}$, where $0 < i, i' \leq n$. In this case, if $(\overline{M}^{\rho_0,c_0} = \overline{M}^{\rho_i,c_i})$, then $\overline{M}^{\rho_0,c_0}$ is a share secret of ρ_i and $\rho_{i'}$ and by Lemma 5.12, we get that $\Omega \models TR(\overline{M}^{\rho_0,c_0})$.

5. In other cases, $GS(M)$ is "true" and holds trivially. ∎

The Proof of Proposition 5.4: Straightforward By Lemma 5.13. ∎

6 Verifying Security Protocols with LLS

In this section, we show how to verify security protocols with LLS. We concentrate on protocol goals like authentication and Secrecy as well as nested epistemic ones. In the case one or more of these protocol goals can not be proved, there may be a flaw for the protocol considered.

6.1 Automatic Verification of Authentication and Secrecy

There are two kinds of goals in CAPSL [30]. One is of the form " PRECEDES P: Q | N, M ", where N is a nonce term. This security property means that in Q's last state, if a "Q"-agent for Q holds P, N, and M, then there is or was a "P"-agent for P holding the same values of Q, N, and M. We can formalize this property as

$$\left(\begin{array}{l} reach(\tau, v, Q, l_Q) \wedge \\ (\overline{P}^{\tau,v} = \rho) \wedge (\overline{N}^{\tau,v} = v) \end{array} \right) \Rightarrow \left(\begin{array}{l} role(\rho, v, P) \wedge (\overline{Q}^{\rho,v} = \tau) \wedge \\ (\overline{N}^{\rho,v} = v) \wedge (\overline{M}^{\rho,v} = \overline{M}^{\tau,v}) \end{array} \right).$$

The other kind of goals is of the form "SECRET N_0", where N_0 is a nonce term generated by a role, say P_0. We formalize this property in LLS as

$$\left(reach(\rho_0, v, P_0, l_{P_0}) \wedge \left(\bigwedge_{0<i\leq n} \overline{P_i}^{\rho_0,v} = \rho_i \right) \wedge (\overline{N_0}^{\rho_0,v} = v) \right) \Rightarrow secr(\rho_0, \rho_1, \cdots, \rho_n, \overline{N_0}^{\rho_0,v}).$$

Let $\phi(v)$ be one of the LLS formulas given above. The verification task is simply to prove that $(\forall v_0)\cdots(\forall v_n)\Theta_{\mathcal{P}}(v_0, \cdots, v_k) \Rightarrow \forall v\phi(\cdots, v)$ is valid, which is equivalent to showing the validity of $(\forall v_0)\cdots(\forall v_k)\Theta_{\mathcal{P}}(v_0, \cdots, v_k) \Rightarrow \phi(\cdots, c)$, where c is a new constant. Choosing a finite subset U of the Herbrand universe containing a new constant c, let $\Theta_{\mathcal{P}}^U = \bigwedge_{0\leq i\leq n,\ \pi_i\in U} \Theta_{\mathcal{P}}(\pi_0, \cdots, \pi_n)$. So, it suffices to show the validity of $\Theta_{\mathcal{P}}^U \Rightarrow \phi(\cdots, c)$. This can be done in propositional reasoning (as there will be no variables in the resulting formula), which allows us to automate our verification process with a SAT solver.

Note that there are possibly some valid properties that can not be proved under $\Theta_{\mathcal{P}}^U$, even under $\Theta_{\mathcal{P}}(v_0, \cdots, v_k)$. Our methodology is therefore incomplete (but sound), though we have not experienced such a case yet.

6.2 Constructing an Observational Theory for Verifying Epistemic Goals

We now demonstrate how to construct a concrete observation theory for the purpose of verification of epistemic goals. To verify security properties like $\alpha(v_1, \cdots, v_h) \Rightarrow K_\tau K_\rho \phi(v_1, \cdots, v_h)$, where α is an observable formula to τ, we introduce new constants $c_1, \cdots, c_h$ and prove $\alpha(c_1, \cdots, c_h) \Rightarrow K_\tau K_\rho \phi(c_1, \cdots, c_h)$. For this purpose, we construct a concrete observation theory $\Gamma_{\mathcal{P}}$. We first choose a finite subset U of the Herbrand universe containing $c_1, \cdots, c_h$. However, to remove term functions, for each $\pi \in U$, we introduce a new constant c^π (if π is c_i $(i \leq h)$ then let c^π be c_i). The system variable set $V_{\mathcal{P}}^U$ of $\Gamma_{\mathcal{P}}$ consists of those atomic formulas like $p(c^\pi)$ and $p(c^\pi, c^{\pi'})$, where p is an unary or binary predicate in LLS and $\pi, \pi' \in U$.

Secondly, instead of taking $\Theta_{\mathcal{P}}^U(\pi_1, \pi_2, \cdots)$ (which is defined in Section 6.1) as the common knowledge base, we use the following formula $\overline{\Theta_{\mathcal{P}}^U}$:

$$\left(\bigwedge_{\pi = g_\rho^N(\pi') \in U} c^\pi = \overline{N}^{\rho, c^{\pi'}} \right) \Rightarrow \Theta_{\mathcal{P}}^U(c^{\pi_1}, c^{\pi_2}, \cdots).$$

Finally, we identify, for each principal ρ_i $(1 \leq i \leq n)$, the set $O_{\rho_i}^U$ of observable variables to principal ρ_i, which includes the following system variables:

- $reach(\rho_i, c^\pi, P, l)$, where $\pi \in U$, $l \leq l_P$;
- $\overline{N}^{\rho_i, c^\pi} = c^{\pi'}$, where $P \in AG$, N is a label message term, and $\pi, \pi' \in U$;

- $norm(\rho_j, c^\pi, P)$, where $P \in AG$, $\pi \in U$, $1 \leq j \leq n$ and $Supp_3^{\rho_j,P} \in \mathcal{S}$; and
- $\overline{P}^{\rho_i,c^\pi} = \rho_j$, where $P \in AG$, $\pi \in U$ and $1 \leq j \leq n$.

Other variables may be observable to ρ_i, but the observability of those variables can be inferred implicitly from the observability of the variables in O_{ρ_i} under the common knowledge base $\Theta_{\mathcal{P}}^U$. We also note that c^π is regarded as concrete value of term π, and the observability of $\overline{N}^{\rho_1,c_1} = c^\pi$ to ρ_1 does not imply that of $\overline{N}^{\rho,c_1} = c^\pi$ to ρ_i (for $i \neq 1$).

Remark 6.1 If $i \neq j$, then $\overline{P_j}^{\rho_i,c} = \rho_j$ is different from $\overline{P_j}^{\rho_j,c} = \rho_j$. The former means that ρ_j plays role P_j in local run (ρ_i, c), which is observable to ρ_i, but the latter indicates that ρ_j plays role P_j in local run (ρ_j, c). In many cases, what we want to verify is that if ρ_j plays role P_j in local run (ρ_i, c), then it does so in local run (ρ_j, c). Similarly, the observation to $\overline{N_j}^{\rho_i,c}$ does not mean that ρ_i can observe $\overline{N_j}^{\rho_j,c}$, the nonce created by ρ_j.

Remark 6.2 In the most cases, we take $U = \{c_1, \cdots, c_h\}$. This means that we the Skolem functions g_ρ^N introduced by the label message existence axioms are not considered. The observation theory we consider is simply

$$\Gamma_{\mathcal{P}} = (V_{\mathcal{P}}^U, \Theta_{\mathcal{P}}^U, O_{\rho_1}^U, \cdots, O_{\rho_n}^U)$$

where

1. $V_{\mathcal{P}}^U$ consists of those atomic formulas like $p(\pi)$ and $p(\pi, \pi')$ for unary and binary predicates P in LLS and $\pi, \pi' \in U$.
2. $O_{\rho_i}^U$ includes the following:
   - $reach(\rho_i, \pi, P, l)$, where $\pi \in U$, $l \leq l_P$;
   - $\overline{N}^{\rho_i,\pi} = \pi'$, where $P \in AG$, N is a label message term, and $\pi, \pi' \in U$;
   - $norm(\rho_j, \pi, P)$, where $P \in AG$, $\pi \in U$, $1 \leq j \leq n$ and $Supp_3^{\rho_j,P} \in \mathcal{S}$; and
   - $\overline{P}^{\rho_i,\pi} = \rho_j$, where $P \in AG$, $\pi \in U$ and $1 \leq j \leq n$.

6.3 Case Study

In this subsection, we analyze by using our methodology the Revised Needham-Schroeder [28] protocol and the original one [34] which are typical authentication protocols from the literature on security. We also verify the Kerberos protocol [36], a more complex example of a security protocol.

6.3.1 The Revised Needham-Schroeder Protocol

For simplicity, we turn our attention to the concise fragment of the Needham-Schroeder protocol, in which the key distribution phase (i.e., the messages exchanged with the trusted server S) is omitted. Lowe [28] points out that the original Needham-Schroeder protocol [34] has the problem of lacking the identity of the responder and can be fixed by a small modification as follows.

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{B, N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

Now let us see how the deduction proceeds based on the framework we defined. We denote the Revised Needham-Schroeder protocol by RNS. To start the analysis of the protocol RNS, we consider those instantiation spaces $\Omega_{RNS} = (\Sigma, , \mathcal{L}, \mathcal{F}, \mathcal{S})$ with

1. $Ag = \{a, b\}$
2. $\mathcal{S} = \{Supp_2^{a,A}, Supp_2^{b,B}, Supp_3^{a,A}, Supp_3^{b,B}\}$

.

The specifications expected for this protocol are:

1. $O_a(v_a, v_b) \Rightarrow K_a K_b(said(a, v_a))$
2. $O_b(v_a, v_b) \Rightarrow K_b K_a(said(b, v_b))$,

where $O_a(v_a, v_b)$ and $O_b(v_a, v_b)$ are formulas that represent a and b's observation, respectively. Let n_a and n_b be two new constants, and consider $U = \{n_a, n_b\}$. To verify the above specifications, we consider the observation theory $\Gamma_{RNS} = (V_{RNS}^U, \Theta_{RNS}^U, O_a^U, O_b^U)$ for protocol RNS as indicated in Subsection 6.2. We shall show that in the resulting knowledge structure $\Gamma_{\mathcal{RNS}}$, both specifications hold.

Let

$$\begin{array}{l} Nrm = \{norm(a, n_a, A), norm(a, n_b, A), norm(b, n_a, B), norm(b, n_b, B)\} \\ O_a^{n_a} = \{\overline{N_a}^{a,n_a} = n_a, \overline{N_b}^{a,n_a} = n_b, role(a, n_a, A), \overline{B}^{a,n_a} = b\} \\ O_b^{n_b} = \{\overline{N_b}^{b,n_b} = n_b, \overline{N_a}^{b,n_b} = n_a, role(b, n_b, B), \overline{A}^{b,n_b} = a\} \end{array} .$$

By the definition of O_a^U, O_b^U, we get that $O_a^{n_a} \subseteq O_a^U$ and $O_b^{n_b} \subseteq O_b^U$. Because $Supp_3^{a,A}, Supp_3^{b,B} \in \mathcal{S}$, we also have that $Nrm \subseteq O_a^U$ and $Nrm \subseteq O_b^U$.

Now, we list some interesting formula in Θ_{RNS}^U. Role Assumptions w.r.t $Supp_2^{\tau,Q} \in \mathcal{S}$ are:

R1 $\Theta_{RNS}^U \models (said(a, n_b) \vee sees(a, n_b)) \Rightarrow role(a, n_b, A)$;

R2 $\Theta_{RNS}^U \models (said(b, n_a) \vee sees(b, n_a)) \Rightarrow role(b, n_a, B)$.

By Normality Constraints:

N1 $\Theta_{RNS}^U \models norm(a, n_a, A) \Leftrightarrow \big(\ role(a, n_a, A) \Rightarrow reach(a, n_a, A, 3)\ \big)$;

N2 $\Theta_{RNS}^U \models \Theta_{RNS}^U \models norm(b, n_b, B) \Leftrightarrow \big(\ role(b, n_b, B) \Rightarrow reach(b, n_b, B, 3)\ \big)$.

By By $(1, 0)$-secrecy property, we have that:

$$\Theta_{RNS}^U \models reach(a, n_a, A, 3) \Rightarrow GS(a, n_a, A, a, n_a, A, 3, \{B, N_a, N_b\}_{K_a})$$

By the definitions, we have that

$$\begin{array}{ll} & GS(A, a, n_a, A, a, n_a, 3, \{B, N_a, N_b\}_{K_a}) \\ = & TR(A, a, n_a, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a} \vee GS(A, a, n_a, A, a, n_a, 3, \{B, N_a, N_b\}) \quad \text{(By item 2(a)of Definition 5.10)} \end{array}$$

$$\begin{array}{ll} & TR(A, a, n_a, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}) \\ = & fresh(a, n_a, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}) \Rightarrow R^a(\{A, N_a\}_{K_b}, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}) \vee R^a(\{N_b\}_{K_b}, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}) \\ & \text{(By the Definition TR)} \end{array}$$

By Definition 5.9 and some Comparison Properties, we have that

$$\Theta_{RNS}^U \models \neg(R^a(\{A, N_a\}_{K_b}, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}) \wedge \neg R^a(\{N_b\}_{K_b}, \overline{\{B, N_a, N_b\}_{K_a}}^{a,n_a}))$$

Moreover, it is easy to check, by the definition of GS, that

$$\begin{array}{ll} & GS(A,a,n_a,A,a,n_a,3,\{B,N_a,N_b\}) \\ = & GS(A,a,n_a,A,a,n_a,3,B) \wedge GS(A,a,n_a,A,a,n_a,3,N_a) \wedge GS(A,a,n_a,A,a,n_a,3,N_b) \\ = & true \wedge GS(A,a,n_a,A,a,n_a,3,N_a) \wedge true \\ = & true \wedge ((\overline{N}^{a,n_a} = \overline{N}^{a,n_a}) \Rightarrow TR(A,a,n_a,\overline{N}^{a,n_a})) \wedge true \end{array}$$

The last is logically equivalent to $TR(A,a,n_a,\overline{N}^{a,n_a}))$, which is

$$fresh(a,n_a,\overline{N}^{a,n_a}) \Rightarrow (R^a(\{A,N_a\}_{K_b},\overline{N}^{a,n_a}) \vee R^a(\{N_b\}_{K_b},\overline{N_a}^{a,n_a}))$$

By Definition 5.9 and some Comparison Properties, we have that

$$\Theta^U_{RNS} \models \neg R^a(\{N_b\}_{K_b},\overline{N_a}^{a,n_a})),$$

which means that a can not reveal $\overline{N_a}^{a,n_a}$ from $\{N_b\}_{K_b}$.

By Definition 5.9 and some Comparison Properties again, we have that

$$\begin{array}{ll} & R^a(\{A,N_a\}_{K_b},\overline{N}^{a,n_a}) \\ = & \neg(\overline{B}^{a,n_a} = a) \wedge R^a(\{A,N_a\},\overline{N}^{a,n_a}) \wedge \bigwedge_{\rho' \in \{a,b\}}((\overline{B}^{a,n_a} = \rho') \Rightarrow sees(\rho',\overline{\{A,N_a\}_{K_b}}^{a,n_a})) \end{array}$$

which logically implies that

$$fresh(a,n_a,\overline{N}^{a,n_a}) \wedge (\overline{B}^{a,n_a} = b) \Rightarrow sees(b,\overline{\{A,N_a\}_{K_b}}^{a,n_a})$$

As a result, we have that

$$\mathbf{K1} \quad \Theta^U_{RNS} \models \left(\begin{array}{l} reach(a,n_a,A,3) \wedge \\ fresh(a,n_a,\overline{\{B,N_a,N_b\}_{K_a}}^{a,n_a}) \wedge \\ fresh(a,n_a,\overline{N_a}^{a,n_a}) \wedge \\ (\overline{B}^{a,n_a} = b) \end{array} \right) \Rightarrow sees(b,\overline{\{A,N_a\}_{K_b}}^{a,n_a})$$

Similarly, we have that

$$\mathbf{K2} \quad \Theta^U_{RNS} \models \Theta^U_{RNS} \models \left(\begin{array}{l} reach(b,n_b,B,3) \wedge \\ fresh(b,n_b,\overline{\{N_b\}_{K_b}}^{b,n_b}) \wedge \\ fresh(b,n_b,\overline{N_b}^{b,n_b}) \wedge \\ (\overline{A}^{b,n_b} = a) \end{array} \right) \Rightarrow sees(a,\overline{\{B,N_a,N_b\}_{K_a}}^{b,n_b})$$

Then by Sees-role Constraints:

$$\mathbf{S1} \quad \Theta^U_{RNS} \models \left(\begin{array}{l} role(b,n_a,B) \wedge sees(b,\overline{\{A,N_a\}_{K_b}}^{a,n_a}) \\ \wedge fresh(b,n_a,\overline{\{A,N_a\}_{K_b}}^{a,n_a}) \end{array} \right) \Rightarrow \bigvee_{M \in Sees^B_{\mathcal{P}}} \overline{\{A,N_a\}_{K_b}}^{a,n_a} = \overline{M}^{b,n_a};$$

$$\mathbf{S2} \quad \Theta^U_{RNS} \models \left(\begin{array}{l} role(a,n_b,A) \wedge sees(a,\overline{\{B,N_a,N_b\}_{K_a}}^{b,n_b}) \\ \wedge fresh(a,n_b,\overline{\{B,N_a,N_b\}_{K_a}}^{b,n_b}) \end{array} \right) \Rightarrow \bigvee_{M \in Sees^A_{\mathcal{P}}} \overline{\{B,N_a,N_b\}_{K_a}}^{b,n_b} = \overline{M}^{a,n_b}.$$

Since $Sees^A_{\mathcal{P}} = \{N_a,N_b,B,\{B,N_a,N_b\}_{K_a}\}$ and $Sees^B_{\mathcal{P}} = \{A,[A,N_a],N_a,N_b,\{N_b\}_{K_b},\{A,N_a\}_{K_b}\}$. By Comparison Constraints, we thus have [4]

[4] If $\overline{N_b}^{b,n_a}$ (i.e. n_b) could equal $\overline{\{A,N_a\}}^{a,n_a}$ (i.e. $\{a,n_a\}$), constraint **C1** would not hold. As a result, the specification could not be verified and an attack can be found even for this protocol. Dealing with more *type violation* is our future work.

C1 $\Theta^U_{RNS} \models \ (\bigvee_{M\in Sees^B_{\mathcal{P}}} \overline{\{A,N_a\}_{K_b}}^{a,n_a} = \overline{M}^{b,n_a}) \Rightarrow \left(\begin{array}{l} (\overline{A}^{b,n_a} = \overline{A}^{a,n_a}) \wedge (\overline{N_a}^{b,n_a} = \overline{N_a}^{a,n_a}) \\ \wedge(\overline{K_b}^{a,n_a} = \overline{K_b}^{b,n_a}) \end{array} \right)$;

C2 $\Theta^U_{RNS} \models \ (\bigvee_{M\in Sees^A_{\mathcal{P}}} \overline{\{B,N_a,N_b\}_{K_a}}^{b,n_b} = \overline{M}^{a,n_b}) \Rightarrow \left(\begin{array}{l} (\overline{N_a}^{b,n_b} = \overline{N_a}^{a,n_b}) \wedge (\overline{N_b}^{a,n_b} = \overline{N_b}^{b,n_b}) \wedge \\ (\overline{K_a}^{b,n_b} = \overline{K_a}^{a,n_b}) \wedge (\overline{B}^{a,n_b} = \overline{B}^{b,n_b}) \end{array} \right)$.

By Constraints **R2, N1, N2, K1, S1, C1** and other trivial constraints, we can easily have that $\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow O^{n_b}_b$. On the other hand, by **R1, N1, N2, K2, S2, C2** and other trivial constraints, we get that $\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow O^{n_a}_a$.

By Proposition 2.3, from $\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow (Nrm \wedge O^{n_b}_b)$ we obtain that $\Theta^U_{RNS} \models K_a(Nrm \wedge O^{n_a}_a) \Rightarrow K_a(Nrm \wedge O^{n_b}_b)$. Because $Nrm \wedge O^{n_a}_a$ is a-observational formula, we have that $\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow K_a(Nrm \wedge O^{n_b}_b)$. Moreover, from $\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow (Nrm \wedge O^{n_a}_a)$, we have $\Theta^U_{RNS} \models K_b(Nrm \wedge O^{n_b}_b) \Rightarrow K_b(Nrm \wedge O^{n_a}_a)$ and $\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow K_b(Nrm \wedge O^{n_a}_a)$. Thus, we have that $\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow K_aK_b(Nrm \wedge O^{n_a}_a)$. Similarly, we can get $\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow K_bK_a(Nrm \wedge O^{n_b}_b)$. As $\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow said(a,n_a)$ and $\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow said(b,n_b)$, we obtain that

$$\Theta^U_{RNS} \models Nrm \wedge O^{n_a}_a \Rightarrow K_aK_bsaid(a,n_a)$$

and

$$\Theta^U_{RNS} \models Nrm \wedge O^{n_b}_b \Rightarrow K_bK_asaid(b,n_b)$$

In conclusion, under some conditions such as Bob's nonce must not be equal to the concatenation of Alice's ID and Alice's nonce, the nested epistemic goals of Protocol NS holds. In particular, Alice can know Bob knows that Alice said her nonce when Alice finished a local session of protocol NS, and Bob can know Alice knows that Bob said his nonce when Bob finished a local session of protocol NS.

6.3.2 The Original Needham-Schroeder Protocol

Here we show that why and how the original NS protocol can not satisfy our specifications. our analysis exactly coincides with Lowe's [27] from another angle. The original Needham-Schroeder protocol is:

1. $A \longrightarrow B : \{A, N_a\}_{K_b}$
2. $B \longrightarrow A : \{N_a, N_b\}_{K_a}$
3. $A \longrightarrow B : \{N_b\}_{K_b}$

We can see the main difference in this case from revised version's is in Sees-role Constraints:

C2 $\left(\bigvee_{M\in Sees^A_{\mathcal{P}}} \overline{\{N_a,N_b\}_{K_a}}^{b,n_b} = \overline{M}^{a,n_a}\right) \Rightarrow \left(\begin{array}{l} (\overline{N_a}^{b,n_b} = \overline{N_a}^{a,n_a}) \wedge (\overline{N_b}^{a,n_a} = \overline{N_b}^{b,n_b}) \wedge \\ (\overline{K_a}^{b,n_b} = \overline{K_a}^{a,n_a}) \end{array} \right)$

So, this time, b can never get $\overline{B}^{a,n_a} = \overline{B}^{b,n_b}$, that is, the responder b can never know if the initiator a knows he or she (a) is talking to the right responder (b), which is the cause of impersonating attacks. In our specification, that is, $(M(\Gamma_{NS}), W_{O_b}) \not\models K_bK_asaid(b,n_b)$. And this is the very point Lowe gives out in [27].

Also, we can successfully verify the "SECRET" goal for nonce N_a, but we fail to show that for N_b.

We notice that when we fail to prove a goal of a protocol, there are two cases: one is the protocol has indeed a flaw and the goal can not be achieved; the other is that the goal can still be achieved but our axiom system is not complete enough to prove it. The latter case is interesting, which forces us to extend our axiom system.

6.3.3 The Kerberos Protocol

In this subsection, we verify the Kerberos protocol [36] with our methodology. Kerberos aims to distribute a fresh symmetric shared key by a trusted server and provides the authentication between parties A and B. The Kerberos protocol is defined as follows:

$A \longrightarrow S : A, B$
$S \longrightarrow A : \{T_s, K_{ab}, B, \{T_s, K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$
$A \longrightarrow B : \{T_s, K_{ab}, A\}_{K_{bs}}, \{A, T_a\}_{K_{ab}}$
$B \longrightarrow A : \{T_a + 1\}_{K_{ab}}$

We denote the Kerberos protocol by Ke. To start the analysis of the protocol Ke, we consider those instantiation spaces $\Omega = (\Sigma, \mathcal{L}, \mathcal{F}, \mathcal{S})$ with $Ag = \{a, b, s\}$ and $\mathcal{S} = \left\{ \begin{array}{l} Supp_2^{a,A}, Supp_2^{b,B}, Supp_2^{s,S}, \\ Supp_3^{a,A}, Supp_3^{b,B}, Supp_3^{s,S} \end{array} \right\}$.

The specifications expected for this protocol are: $O_a(v_a, v_s) \Rightarrow K_a K_b(said(a, v_a) \wedge said(s, \overline{K_{ab}}^{s,v_s}))$ and $O_b(v_a, v_s) \Rightarrow K_b K_a(said(b, v_a) \wedge said(s, \overline{K_{ab}}^{s,v_s}))$, where $O_a(v_a, v_s)$ and $O_b(v_a, v_s)$ are formulas that represent a and b's observations, respectively. Let t_a and t_s be two new constants, and consider $U = \{t_a, t_s\}$. In this case, $\overline{\Theta_{Ke}^U}$ is the same as Θ_{Ke}^U. To verify the two security properties, we construct the observation theory $(V_{Ke}^U, \Theta_{Ke}^U, O_a^U, O_b^U, O_s^U)$ for the protocol Ke as defined above, and show that $\Theta_{Ke}^U \vdash O_a \Rightarrow K_a K_b(said(a, t_a) \wedge said(s, \overline{K_{ab}}^{s,t_s}))$ and $\Theta_{Ke}^U \vdash O_b \Rightarrow K_b K_a(said(b, t_a) \wedge said(s, \overline{K_{ab}}^{s,t_s}))$.

Let $Nrm = \left\{ norm(\rho, c, P) \,\middle|\, \rho = a, b, s;\ c \in \mathcal{C}(\rho); P = A, B, S \right\}$. Variables in Nrm are observable to all principals due to suppositions $Supp_3^{a,A}, Supp_3^{b,B}$ and $Supp_3^{s,S}$ for the instantiation spaces.

$O_a^{t_s} = \left\{ role(a, t_s, A), \overline{B}^{a,t_s} = b, \overline{S}^{a,t_s} = s, \overline{T_a}^{a,t_s} = t_a, \overline{T_s}^{a,t_s} = t_s \right\}$.
$O_b^{t_s} = \left\{ role(b, t_s, B), \overline{A}^{b,t_s} = a, \overline{S}^{b,t_s} = s, \overline{T_a}^{b,t_s} = t_a, \overline{T_s}^{b,t_s} = t_s \right\}$.
$O_s^{t_s} = \left\{ role(s, t_s, S), \overline{A}^{s,t_s} = a, \overline{B}^{s,t_s} = b, \overline{T_a}^{s,t_s} = t_a, \overline{T_s}^{s,t_s} = t_s \right\}$.

Clearly $O_a^{t_s} \subseteq O_a^U$, $O_b^{t_s} \subseteq O_b^U$ and $O_s^{t_s} \subseteq O_s^U$; in other words, variables in $O_a^{t_s}$, $O_b^{t_s}$ and $O_s^{t_s}$ are observable to a, b and s, respectively.

We now show that $\Theta_{Ke}^U \vdash Nrm \wedge O_a^{t_s} \Rightarrow O_s^{t_s} \wedge O_b^{t_s}$ and $\Theta_{Ke}^U \vdash Nrm \wedge O_b^{t_s} \Rightarrow O_s^{t_s} \wedge O_a^{t_s}$

We first apply $(0, 1)$-Secrecy Properties for the second message
$M_2 = \{T_s, K_{ab}, B, \{T_s, K_{ab}, A\}_{K_{bs}}\}_{K_{as}} \in Recvs(A, 2)$.
We thus have $\Theta_{Ke}^U \vdash reach(a, t_s, A, 2) \Rightarrow GS(A, a, t_s, M_2)$.
After simplification, it turns out that $\Theta_{Ke}^U \vdash reach(a, t_s, A, 2) \Rightarrow ((\overline{S}^{a,t_s} = s) \wedge said(s, \overline{M_2}^{a,t_s}))$.
Thus, $\Theta_{Ke}^U \vdash Nrm \wedge O_a^{t_s} \Rightarrow said(s, \overline{M_2}^{a,t_s})$.
However, $\Theta_{Ke}^U \vdash said(s, \overline{M_2}^{a,t_s}) \Rightarrow said(s, t_s)$.
By Role Assumption, we have also that $\Theta_{Ke}^U \vdash said(s, t_s) \Rightarrow role(s, t_s, S)$.
By Said-role Properties, $\Theta_{Ke}^U \vdash\ said(s, \overline{M_2}^{a,t_s}) \wedge fresh(s, t_s, \overline{M_2}^{a,t_s}) \Rightarrow (\overline{M_2}^{a,t_s} = \overline{M_2}^{s,t_s})$.
By Freshness Properties, $\Theta_{Ke}^U \vdash Nrm \wedge O_a^{t_s} \Rightarrow fresh(s, t_s, \overline{M_2}^{a,t_s})$.
Finally, by Comparison Properties,
$\Theta_{Ke}^U \vdash Nrm \wedge O_a^{t_s} \Rightarrow\ (\overline{A}^{a,t_s} = \overline{A}^{s,t_s}) \wedge (\overline{B}^{a,t_s} = \overline{B}^{s,t_s}) \wedge (\overline{K_{ab}}^{a,t_s} = \overline{K_{ab}}^{s,t_s}) \wedge (\overline{S}^{a,t_s} = \overline{S}^{s,t_s}) \wedge (\overline{M_{31}}^{a,t_s} = \overline{M_{31}}^{s,t_s})$
where $M_{31} = \{T_s, K_{ab}, A\}_{K_{bs}}$.
By the discussion above, we have that $\Theta_{Ke}^U \vdash Nrm \wedge O_a^{t_s} \Rightarrow O_s^{t_s}$.

To get knowledge about b from a's observation, we consider message $M_4 = \{T_a+1\}_{K_{ab}} \in Recvs(A,4)$. First, we use $(2,0)$-Secrecy Properties and get that

$\Theta^U_{Ke} \vdash reach(a,t_s,A,4) \wedge rch(a,t_s,A,4) \wedge rch(s,t_s,S,2) \Rightarrow GS(A,a,t_s,A,a,t_s,4,S,s,t_s,2,M_4)$

where $GS(M_4)$ equals $TR(\overline{M_4}^{a,t_s}) \vee \Big((\overline{K_{ab}}^{s,t_s} = \overline{K_{ab}}^{a,t_s}) \Rightarrow\ TR(\overline{K_{ab}}^{a,t_s}) \wedge GS(\{T_a+1\})\Big)$.

It is not difficult to see that $TR(\overline{M_4}^{a,t_s})$ logically equivalent to $false$. And $TR(\overline{K_{ab}}^{a,t_s})$ logically implies $fresh(a,t_s,\overline{K_{ab}}^{a,t_s}) \wedge fresh(s,t_s,\overline{K_{ab}}^{a,t_s}) \Rightarrow R^a(M_{31},\overline{K_{ab}}^{a,t_s})$.

Thus, we will eventually have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow sees(b,\overline{M_2}^{a,t_s})$.

And hence $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow sees(b,t_s)$.

Similarly, by Role Assumption, $\Theta^U_{Ke} \vdash sees(b,t_s) \Rightarrow role(b,t_s,B)$.

By See-role and Comparison Properties,

$\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow \Big(\ (\overline{A}^{b,t_s} = \overline{A}^{a,t_s}) \wedge (\overline{K_{ab}}^{b,t_s} = \overline{K_{ab}}^{a,t_s}) \wedge (\overline{T_s}^{b,t_s} = \overline{T_s}^{a,t_s}) \wedge (\overline{S}^{b,t_s} = \overline{S}^{a,t_s})\ \Big)$.

To get $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow (\overline{T_a}^{b,t_s} = \overline{T_a}^{a,t_s})$, we consider again $M_4 = \{T_a+1\}_{K_{ab}} \in Recvs(A,4)$.

However, we use $(3,0)$-Secrecy Properties and get that

$\Theta^U_{Ke} \vdash rch(a,t_s,A,4) \wedge rch(b,t_s,B,2) \wedge rch(s,t_s,S,2) \Rightarrow GS(A,a,t_s,A,a,t_s,4,B,b,t_s,2,S,s,t_s,2,M_4)$.

We can get $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow (\overline{M_4}^{b,t_s} = \overline{M_4}^{a,t_s})$ by the definition of $GS(A,a,t_s,A,a,t_s,4,B,b,t_s,2,S,s,t_s,2,M_4)$, which leads to $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow (\overline{T_a}^{b,t_s} = \overline{T_a}^{a,t_s})$.

Therefore, $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow O^{t_s}_b$.

By similar method above, we can obtain the knowledge that principal b gains when b receives message $M_3 = \{M_{31}, M_{32}\}$. We then use this message term M_3 to apply three different Secrecy Properties.

Firstly, by $(0,1)$-Secrecy Properties we get

$\Theta^U_{Ke} \vdash\ \ reach(b,t_s,B,1) \Rightarrow GS(B,b,t_s,M_{31}) \wedge GS(B,b,t_s,M_4)$.

Considering $GS(B,b,t_s,M_{31})$, with other routine constraints we get $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow O^{t_s}_s$.

Secondly, by $(2,0)$-Secrecy Properties and other routine constraints, we can obtain that

$\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow sees(a,\overline{M_2}^{s,t_s})$.

Then, we have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow sees(a,t_s)$.

By Role Assumption again, we have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow role(a,t_s,A)$.

By Sees-role Properties and Comparison Properties, we have

$\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow \Big(\ (\overline{B}^{a,t_s} = \overline{B}^{s,t_s}) \wedge (\overline{S}^{a,t_s} = \overline{S}^{s,t_s}) \wedge (\overline{T_s}^{a,t_s} = \overline{T_s}^{s,t_s}) \wedge (\overline{K_{ab}}^{a,t_s} = \overline{K_{ab}}^{s,t_s})\ \Big)$.

Finally, by $(3,0)$-Secrecy Properties and other routine constraints, we have

$\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow (\overline{\{A,T_a\}}^{a,t_s} = \overline{\{A,T_a\}}^{s,t_s})$.

To sum up, from b's observation $Nrm \wedge O^{t_s}_b$, we can obtain s's observation $O^{t_s}_s$ and a's observation $O^{t_s}_a$.

Now we verify the two specifications of this protocol. Let $\varphi = said(a,t_a) \wedge said(s,\overline{K_{ab}}^{s,t_s})$.

First, by the above discussion, we have that $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow \varphi$, because $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow O^{t_s}_s \wedge O^{t_s}_a$, $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow said(a,t_a)$ and $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_s \Rightarrow said(s,\overline{K_{ab}}^{s,t_s})$.

Thus, $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow (Nrm \wedge O^{t_s}_b \wedge (Nrm \wedge O^{t_s}_b \Rightarrow \varphi))$.

So, by the definition of modality K_b, we have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_b \Rightarrow K_b\varphi$.

However, we have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow Nrm \wedge O^{t_s}_b$, and hence $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow K_b\varphi$.

Thus, $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow (Nrm \wedge O^{t_s}_a \wedge (Nrm \wedge O^{t_s}_a \Rightarrow K_b\varphi))$.

By the definition of modality K_a, we have $\Theta^U_{Ke} \vdash Nrm \wedge O^{t_s}_a \Rightarrow K_aK_b\varphi$.

We now achieve the conclusion $\Theta^U_{Ke} \vdash O^U_a \Rightarrow K_aK_b(said(a,t_a) \wedge said(s,\overline{K_{ab}}^{s,t_s}))$.

By the same argument, we can obtain $\Theta^U_{Ke} \vdash O^U_b \Rightarrow K_b K_a(said(b, t_a) \wedge said(s, \overline{K_{ab}}^{s,t_s}))$.

Therefore, we have verified the nested epistemic goals for the Kerberos Protocol. We can also verify some routine goals such as SECRET and PRECEDES. Nevertheless, we do not intend to provide more detailed analysis here and we refer the readers to the analysis done by Cervesato et al [12].

7 Experimental Results

We have implemented and further optimized our security protocol verifier tool SPV, with a SAT solver SBSAT [22]. We have verified a lot of protocols from the Security Protocols Open Repository (http://www.lsv.ens-cachan.fr/spore/index.html). We present only the partial data here; for more details on the tool, please refer to Appendix B. The experiments were conducted on a PC with AMD Opteron242, 2GB DDR memory, running under Redhat Linux 7.0 with gcc 3.2.2.

7.1 Efficiency and Scaling

The following table summarizes the results of our preliminary experiments in terms of the number of variables, properties generated and time required for some well-known protocols.

Protocol	Variables	Properties	Time (s)
Needham-Schroeder	3548	6296	53.485
Rev Needham-Schroeder	3548	6332	49.609
Den-Sacco shared key	2178	2905	25.045
Woo and Lam Pi	1984	3248	26.214
Kao Chow Authen v1	8197	14316	330.086
Kerberos V5	8834	14946	327.585
The Simplified SET purchase	61065	83747	31364
The SET purchase	147839	222563	224630

We can see that with higher complexity of the protocols, which is decided by both message structures, principals involved and protocol steps, SPV needs more time to deal with them by generating more variables and properties. After all the variables and properties have been generated, the tasks are passed to the SAT solver. Most of the time are spent on the generations of variables and properties, and the time for verification is just within seconds. So there will be a lot of scope for optimizations and scaling to utilize the power of the underlying SAT solver SBSAT [22].

7.2 Verified Specifications

SPV can verify a lot of interesting security properties including SECRET and PRECEDES in CAPSL [30], as well as two-level epistemic specifications like "Alice observes (knows) Bob observes (knows) Alice said something". Some experimental results are shown below (see Appendix B for more details.)

We use the notation $\mathbf{P}^A_B(N, M)$ to refer to "PRECEDES A:B| N,M". As shown in the flowing table, the confirmed specifications are signed by "+", and the rest by "−".

Protocol	SECRET	PRECEDES
Needham-Schroeder	$+N_a - N_b$	$-\mathbf{P}_B^A(N_a, N_b) + \mathbf{P}_A^B(N_a, N_b)$
Rev Needham-Schroeder	$+N_a + N_b$	$+\mathbf{P}_B^A(N_a, N_b) + \mathbf{P}_A^B(N_a, N_b)$
Den-Sacco shared key	$+K_{ab}$	$-\mathbf{P}_B^A(T_s, K_{ab}) - \mathbf{P}_A^B(T_s, K_{ab})$
Woo and Lam Pi	$-N_b$	$+\mathbf{P}_B^A(N_b) - \mathbf{P}_A^B(N_b)$
Kao Chow Authen v1	$-N_a - N_b + K_{ab}$	$+\mathbf{P}_B^A(N_a, N_b) + \mathbf{P}_A^B(N_a, N_b)$
Kerberos V5	$+K_{ab}$	$+\mathbf{P}_B^A(T_s, K_{ab}) + \mathbf{P}_A^B(T_s, K_{ab})$
The SET purchase	+PurchAmt	$+\mathbf{P}_P^C$ (XID,CardSecret)

Also, let $\mathbf{K}_{AB}^1 N$ stand for some one-level epistemic specification $O_a \Rightarrow K_a(said(b, X) \vee sees(b, X))$, where a is for A and runs a local session with b for B and X for N, and O_a express a's observations when reaching the final step of the protocol. Similarly, $\mathbf{K}_{AB}^2 N$ stands for two level ones $O_a \Rightarrow K_a K_b(said(a, Y) \vee sees(a, Y))$, where a and b are the same as above and Y is for N in a corresponding local session of b. The partial epistemic goals we have verified are shown in the table below.

Protocol	EPISTEMIC GOALS
Needham-Schroeder	$+\mathbf{K}_{AB}^1 N_a + \mathbf{K}_{AB}^2 N_b - \mathbf{K}_{BA}^2 N_a$
Rev Needham-Schroeder	$+\mathbf{K}_{AB}^2 N_b + \mathbf{K}_{BA}^2 N_a$
Den-Sacco shared key	$-\mathbf{K}_{AB}^1 T_s - \mathbf{K}_{BA}^1 T_s$
Woo and Lam Pi	$-\mathbf{K}_{AB}^1 N_b + \mathbf{K}_{BA}^1 N_b - \mathbf{K}_{AB}^2 N_b$
Kao Chow Authen v1	$+\mathbf{K}_{AB}^2 N_b + \mathbf{K}_{BA}^2 N_a$
Kerberos V5	$+\mathbf{K}_{AB}^2 T_a + \mathbf{K}_{BA}^2 T_a$
The SET purchase	$+\mathbf{K}_{CM}^1$ Chall_C$-\mathbf{K}_{CM}^2$ Chall_M + $\mathbf{K}_{MC}^2$ Chall_C

For more detailed information for the experimental results carried out with SPV, we refer the readers to Appendix B or the following link:

`http://www.cs.sysu.edu.cn/~skl/spv.htm`

As for the SET purchase protocol, we notice that we do not intend to offer more detailed analysis as done in [8] with Isabelle; what we want know from these experiments is whether an automatic justification-oriented approach can be scaled to the size of the SET purchase phase protocol.

8 Discussion

In this section, we review several approaches of formal analysis of security protocols.

8.1 Epistemic Logic Approaches

The epistemic logic approaches to verification of security protocols was originated by BAN logic [11] and its various alternatives [23, 2, 13, 40, 37], etc. GNY [23], Mao [13], and AT [2] add to and reformulate rules of BAN so that they can have a more powerful reasoning ability. AT goes a further step to give a clear model-theoretic semantics which guarantees that the reasoning in this logic is sound. VO [40] adds rules to BAN to reason about the key-agreement protocols. SVO [37] is a unified authentication protocol logic that encompasses three of these expansions (GNY, AT, VO) and BAN. However, BAN logic and its alternatives operate on some difficult-to-establish idealizations of protocols instead of original protocols, and thus are hard to be transformed into algorithms directly from protocols.

The presented LLS logic can be regarded as an epistemic logic approach. However, this logic is easier to be understood for general users who are not familiar with modal logics. In fact, the epistemic modality 'observation' in our approach can be eliminated because it is defined directly by the set of constraints (axioms) and, for every principal, the set of observable variables to this principal.

8.2 Justification-Oriented Approaches

Cervesato, Meadows and Pavlovic [14] also give an interesting encapsulated authentication logic about partial orders of protocol actions that encapsulates secrecy requirements and honesty rules as assumptions. Both the encapsulated authentication logic and the presented work are based on the reasonings about the *observation* of the principals. However, they differ in that the encapsulated authentication logic is not provided with an algorithm to obtain all the axioms for an arbitrary given protocol, while our LLS presents a concrete algorithm for building a complete knowledge structure (see also Remark 5.8). Moreover, our approach works on Clark-Jacob like definitions of protocols, and the notion of instantiation function characterizes precisely the meaning of a protocol. Although the Clark-Jacob like definition is somewhat vague, our model captures exactly the underlying vagueness because, given a CME model, there may be different instantiation spaces.

In [20], the authors introduce a compositional logic, which can be applied to analyze the derivation of a family of protocols [17]. The presented work is similar to theirs in that both of them reason about all possible runs of a protocol, without explicitly reasoning about steps that might be carried out by the attacker, and thus provide the basis for a protocol correctness proof. However, they differ in the following four aspects:

- The $(n, 1)$-secrecy properties identified in the presented approach are beyond those axioms in compositional logic (See Remark 5.7).
- The axioms in LLS are mainly in propositional logic, and an implemented verifying system has been carried out using SAT solvers, while there are no automation results for the compositional logic yet.
- The compositional logic currently deals only with public key protocols, while the presented logic is much more flexible and allow public keys, shared keys, private keys, hash keys, as well as freshly generated key to be used in the concerned protocol.
- Finally, compositional logic forbids "crosstalk", while ours forbids only those cross talks with "fresh messages" (See Remark 4.4.)

8.3 Discussion about Semantics

There are basically two kinds of semantics models for protocol logics: event trace model and Kripke structure. Event trace models are used to identify protocol participants' behaviors; while Kripke structures are employed to model protocol participants' belief or knowledge after or during running a protocol. In [37], Syversion and Oorschot interpret the BAN-like logics via the protocol trace model, while using the Kripke structure to model principals' belief. Halpern and Pucella [25] compare message passing systems with Strand Space models.

In the presented approach, we essentially have two semantics models: instantiation spaces and observation theories. The instantiation space model is a concrete and natural one, which is employed to demonstrate the soundness of the presented approach. Nevertheless, the instantiation space model is not appropriate for model checking security properties because of its infinity and concreteness nature.

Fortunately, an observation theory can be regarded as a symbolic Kripke structure, which presents a succinct and algorithmic manageable way to interpret the observation (knowledge) modalities.

8.4 Automatic Verification Tools

Automated approaches can be divided into two classes: model checking within bounded sessions and theorem proving for unbounded sessions.

Security protocols can be verified or attacks can be found by general-purpose model checkers like $Mur\phi$ [32], FDR [28], or by specific model checkers like Brutus [21], SATMC [4], Constraint Solver [31] and OFMC [6]. But they must assume bounded sessions, falsification-oriented and cannot prove the the concerned properties for the actual infinite runs.

The theorem proving approach is inference-based, aiming at the proofs of the concerned properties under unbounded sessions. Examples of protocol verification by generic theorem provers are Isabelle [7] by inductive proofs. The specific tools of protocol verification are Athena [35], NPA [15], and ProVerif [9], etc. Those automated approaches which rely on the generic theorem prover such as Isabelle have to be semi-automatic and user-guided, which require expertise with theorem provers.

Athena [35] is rather efficient to prove the properties of a protocol with arbitrary runs or generate a counterexample, due to the compact representation of state by the Strand Space Model, but there is a risk because the validation procedure is not guaranteed to terminate for general cases when allowing arbitrary configurations of the protocol execution. Although termination can be forced by bounding the number of concurrent protocol runs and the length of messages, there is no general and precise principles to tell how to do it and at this point Athena makes no difference to model checkers and fails to prove some properties with infinite runs.

SATMC [4] is the first tool to utilize the power of modern SAT solvers for the verification of security protocols, as part of the AVISPA project [1]. The SATMC makes use of the strength of SAT solvers as the *searching engine* in their protocol falsification procedure, while we use SAT solvers as the *proof engine* for the properties under unbounded sessions. Furthermore, SATMC builds a propositional formula encoding all the possible attacks of a bounded length, and thus fails to find any other attacks beyond this length. OFMC [6] employs several symbolic techniques to explore the state space in a demand-driven way and is quite successful in finding attacks in bounded sessions but it can not prove properties in unbounded sessions. Generally speaking, AVISPA project is falsification-oriented in that the user needs to define each role by writing out the state transition explicitly in their front end language HLPSL, while the input language of SPV is just the actual definitions of the protocols and is more user-friendly since it has more intimacy to its original forms.

HERMES [10] is a tool for the verification of the secrecy of bounded and unbounded protocols by their abstraction steps. Although it is rather efficient in finding flaws, it may suffer from the problem of giving out a false attack, which is not valid on the concrete level while valid at the abstraction level. Furthermore, HERMES only deals with secrecy and does not consider other properties such as authentication yet.

ProVerif [9] is also a very efficient tool for secrecy properties. Nevertheless, it also suffers from the problem of nontermination for some simple protocols with unbounded sessions. Moreover, we do not think that the underlying logic programming languages are convenient to formalize $(n, 1)$-Secrecy Properties presented in LLS. Also, its input language is based on process calculus and thus it may be hard for non-expert users to write out complicated protocols.

9 Concluding Remarks

We have presented a new protocol logic based on a quite natural semantic model called Instantiation Space. Our logic is very flexible, and it can deal with complex message formats with various arbitrarily nested encryptions. Most importantly, this logic is implemented and resulted in a robust Security Protocol Verifier (SPV), with which we are able to automatically verify some interesting complex properties for many security protocols including industrial-strength ones such as the SET purchase phase protocol. Although the SET purchase phase protocol can be analysed by semi-automatic verifiers like theorem prover Isabelle [8], its security properties, to the best of our knowledge, have not been proved by the fully automatic verifiers other than SPV.

The main contributions of this paper are as follow:

1. We present a quite natural semantic model called Instantiation Space, which is directly based the log files recoding principals' message data flows. Various Properties of principals' behavior and the underlying communication environment can be conveniently characterized by using this model.

2. The $(n, 1)$-secrecy properties identified as kernel axioms in our protocol logic LLS are quite general, which beyond those related axioms in most other protocol logics such as compositional logic.

3. By using the notion of an observation theory, we provide a natural and the theoretically solid way to calculate a principal's knowledge and a principal's knowledge about other principal's knowledge. As a result, We are able to verify nested epistemic goals for security protocols.

Our future work will focus on the following extension and enhancement of our theory: firstly, we could add more cryptographic primitives such as the Diffie-Hellman algorithm [18] to our logic just in the similar way Paul van Oorschot does in [40], and more message constructor like "exclusive or". Secondly, we can incorporate complexity-theoretic model such as [16] in which correctness is defined in terms of probability and complexity theory from lower level, cryptographic considerations, into our theory to give it more fidelity for real world applications. Finally, we can consider new kinds of complex security protocols in our logic such as those with alternative execution paths [8], cryptographic voting protocols [26], and contract signing protocols [5], along with many new properties such as non-repudiation and anonymity [29], etc. Also, we will explore optimizations and heuristics necessary to make SPV more efficient.

References

[1] Automated Validation of Internet Security Protocols and Applications (AVISPA), http://www.avispa-project.org/.

[2] M. Abadi and M.R. Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 201–216, 1991.

[3] P.Lincoln and N.Durgin, J.Mitchell, and A.Scedrov. Undecidability of bounded security protocols. In *Proceedings of Workshop on Formal Methods and Security Protocols (FMSP'99), Trento, Italy,*, 1999.

[4] Alessandro Armando and Luca Compagna. SATMC: a SAT-based Model Checker for Security Protocols. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, LNAI 3229, pages 730–733, Lisbon, Portugal, September 2004. Springer-Verlag.

[5] Michael Backes, Anupam Datta, Ante Derek, John C. Mitchell, and Mathieu Turuani. Compositional analysis of contract signing protocols. In *18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005), 20-22 June 2005, Aix-en-Provence, France*, pages 94–110, 2005.

[6] David A. Basin, Sebastian Mödersheim, and Luca Viganò. An on-the-fly model-checker for security protocol analysis. In *Computer Security - ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjøvik, Norway, October 13-15, 2003*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.

[7] Giampaolo Bella and L. C. Paulson. Using Isabelle to prove properties of the Kerberos authentication system. In *DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.

[8] Giampaolo Bella, Lawrence C. Paulson, and Fabio Massacci. The verification of an industrial payment protocol: the SET purchase phase. In *ACM Conference on Computer and Communications Security*, pages 12–20, 2002.

[9] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings of 14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*, pages 82–96. IEEE Computer Society, 2001.

[10] Liana Bozga, Yassine Lakhnech, and Michaël Périn. Hermes: An automatic tool for verification of secrecy in security protocols. In *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003*, pages 219–222, 2003.

[11] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), 1990.

[12] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, and Andre Scedrov. A formal analysis of some properties of kerberos 5 using msr. In *CSFW*, pages 175–, 2002.

[13] C.Boyd and W.Mao. On a limitation of BAN logic. In *Proceedings of EUROCRYPT'93*, Springer-Verlag LNCS No. 765, pages 240–247, 1993.

[14] Iliano Cervesato, Catherine Meadows, and Dusko Pavlovic. An Encapsulated Authentication Logic for Reasoning About Key Distribution Protocol. In *Eighteenth Computer Security Foundations Workshop — CSFW-18*, Aix-en-Provence, France, 20–22 June 2005. IEEE Computer Society Press.

[15] C.Meadows. The NRL Protocol Analyzer:an overview. In *Proc. 2nd Conf. of the Second International Conference on the Practical Applications of Prolog*, 1994.

[16] A. Datta, A. Derek, and and M. Turuani J. C. Mitchell, and V. Shmatikov. Probabilistic polynomial-time semantics for a protocol security logic. In *Proceedings of 32nd International Colloquium on Automata, Languages and Programming (ICALP-05)*, pages 16–29, 2005.

[17] Anupam Datta, Ante Derek, John C. Mitchell, and Dusko Pavlovic. A derivation system for security protocols and its logical formalization. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003), 30 June - 2 July 2003, Pacific Grove, CA, USA*, pages 109–125. IEEE Computer Society, 2003.

[18] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[19] D. Dolev and A.C. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29(8):198–208, August 1983.

[20] Nancy A. Durgin, John C. Mitchell, and Dusko Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4):677–722, 2003.

[21] E.M.Clarke, S. Jha, and W.Marrero. Verifying Security protocols with Brutus. *ACM Transactions on Software Engineering and Methodology*, 9(4):443–487, October 2000.

[22] John V. Franco, Michal Kouril, John S. Schlipf, Jeffrey Ward, Sean Weaver, Michael Dransfield, and W. Mark Vanfleet. Sbsat: a state-based, bdd-based satisfiability solver. In *Proceedings of the Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003.*, volume 2919 of *Lecture Notes in Computer Science*, pages 398–410. Springer, 2003.

[23] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about beliefs in cryptographic protocols. In *Proc. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 234–248, Los Alamitos, California, 1990. IEEE Computer Society Press.

[24] Li Gong and Paul Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 44–55, 1995.

[25] Joseph Y. Halpern and Riccardo Pucella. On the relationship between strand spaces and multi-agent systems. *ACM Trans. Inf. Syst. Secur.*, 6(1):43–70, 2003.

[26] Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: A systems perspective. In *Fourteenth USENIX Security Symposium (USENIX Security 2005)*, August 2005.

[27] G. Lowe. A heirarchy of authentication specifications. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, 1997.

[28] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Margaria and Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, Vol 1055 of Lecture Notes in Computer Science, pages 147–166. Springer Verlag, 1996.

[29] Catherine Meadows. Ordering from satan's menu: a survey of requirements specification for formal analysis of cryptographic protocols. *Sci. Comput. Program.*, 50(1-3):3–22, 2004.

[30] Jonathan K. Millen. Common Authentication Protocol Specification Language (CAPSL), http://www.csl.sri.com/users/millen/capsl/.

[31] Jonathan K. Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of 8th ACM Conference on Computer and Communications Security*, pages 166–175, 2001.

[32] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murphi. In *Proc. of IEEE Symposium on Security and Privacy*, 1997.

[33] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[34] R.M.Needham and M.D.Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, 1978.

[35] D. Song, S. Berezin, and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1,2):47–74, 2001.

[36] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, pages 191–202, 1988.

[37] P. F. Syversion and P.C. van Oorschot. A unified cryptographic protocol logic. Technical Report NRL Publication 5540-227, Naval Research Lab, 1996.

[38] Paul F. Syverson. Adding time to a logic of authentication. In *ACM Conference on Computer and Communications Security*, pages 97–101, 1993.

[39] F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces. Technical report, The MITRE Corporation, November,1997.

[40] P. van Oorschot. Extending cryptographic logics of belief to key agreement. In *Proc First ACM Conf on Computer and Communications Security*, pages 232–243, November 1993.

[41] Thomas Y. C. Woo and Simon S. Lam. A semantic model for authentication protocols. In *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, May 1993.

Appendix

A A List of Valid Properties

We divide valid properties in into three parts: routine ones , kernel ones, and equivalence and comparison ones.

A.1 Routine Properties

Normality Properties: $\models norm(\rho, c, P) \Leftrightarrow (role(\rho, c, P) \Rightarrow reach(\rho, c, P, l_P))$.

Nonce Properties:

$$\models \ (\overline{N}^{\rho,c_1} = c_2) \Rightarrow \left(\begin{array}{l} (\overline{M}^{\rho,c_1} = \overline{M}^{\rho,c_2}) \wedge (fresh(\rho, c_1, X) \Leftrightarrow fresh(\rho, c_2, X)) \wedge \\ (reach(\rho, c_1, P, l) \Leftrightarrow reach(\rho, c_2, P, l)) \end{array} \right)$$

where N be a nonce or stamp or a temporally created key in the protocol.

Reach Properties:

1. $\models \left(\begin{array}{l} (reach(\rho, c, P, 0) \Leftrightarrow role(\rho, c, P)) \wedge \neg reach(\rho, c, P, l_P + 1) \wedge \\ (reach(\rho, c, P, l + 1) \Rightarrow reach(\rho, c, P, l)) \end{array} \right)$

2. $\models reach(\rho, c, P, l) \Rightarrow \left(\ poss(\rho, \overline{M_1}^{\rho,c}) \wedge sees(\rho, \overline{M_2}^{\rho,c}) \wedge said(\rho, \overline{M_3}^{\rho,c}) \ \right)$,
   where $M_1 \in Poss(P, l)$, $M_2 \in Sees(P, l)$ and $M_3 \in Said(P, l)$.

Concatenation Properties: For $[X_1, \ldots, X_m] \in \overline{MSG_{\mathcal{P}}}^{\nu}$ and $i = 1, \ldots, m$,
$\models poss(\rho, [X_1, \ldots, X_m]) \Rightarrow poss(\rho, X_i)$
$\models sees(\rho, [X_1, \ldots, X_m]) \Rightarrow sees(\rho, X_i)$
$\models said(\rho, [X_1, \ldots, X_m]) \Rightarrow said(\rho, X_i)$.

Possession Properties:

1. $\models \overline{P}^{\tau,c} = \rho \Rightarrow poss(\rho, \overline{M}^{\tau,c})$, where $\overline{M}^{\tau,c} \in INIT(P)$.

2. $\models role(\tau, c, Q) \wedge (\overline{P}^{\tau,c} = \rho) \Rightarrow poss(\rho, \overline{K_{PQ}}^{\tau,c})$, where K_{PQ} is a shared secret between P and Q.

3. $\models (\overline{P}^{\tau,c} = \rho) \wedge \neg(\overline{R}^{\tau,c} = \rho) \wedge \neg(\overline{Q}^{\tau,c} = \rho) \Rightarrow \neg poss(\rho, \overline{K_{RQ}}^{\tau,c})$, where K_{RQ} is a shared secret between R and Q.

4. $\models (\overline{P}^{\tau,c} = \rho) \Rightarrow \neg poss(\rho, \overline{K_Q^{-1}}^{\tau,c})$, where Q differs from Q and K_Q is the public key of Q.

5. $\models reach(\tau, c, Q, l) \wedge poss(\rho, \overline{M}^{\tau,c}) \wedge poss(\rho, \overline{K}^{\tau,c}) \Rightarrow poss(\rho, \overline{\{M\}_K}^{\tau,c})$,
   where $\{M\}_K \in Recg(Q, l)$.

6. $\models reach(\tau, c, Q, l) \wedge poss(\rho, \overline{\{M\}_K}^{\tau,c}) \wedge poss(\rho, \overline{K^{-1}}^{\tau,c}) \Rightarrow poss(\rho, \overline{M}^{\tau,c})$,
   where $\{M\}_K \in Recg(Q, l)$.

7. $\models (sees(\rho, X) \Rightarrow poss(\rho, X)) \wedge (said(\rho, X) \Rightarrow poss(\rho, X))$.

Seeing Properties: For $\{M\}_K \in Recg(Q, l)$,
$\models reach(\tau, c, Q, l) \wedge sees(\rho, \overline{\{M\}_K}^{\tau,c}) \wedge poss(\rho, \overline{K^{-1}}^{\tau,c}) \Rightarrow sees(\rho, \overline{M}^{\tau,c})$.

Saying Properties: For $\{M\}_K \in Recg(Q, l)$,
$\models reach(\tau, c, Q, l) \wedge said(\rho, \overline{\{M\}_K}^{\tau,c}) \wedge poss(\rho, \overline{K}^{\tau,c}) \wedge poss(\rho, \overline{M}^{\tau,c}) \Rightarrow said(\rho, \overline{M}^{\tau,c})$.

Said-role Properties:

$\models role(\rho, c, P) \wedge said(\rho, X) \wedge fresh(\rho, c, X) \wedge \bigwedge_{M \in Said(P,\ l)} \neg(X = \overline{M}^{\rho,c}) \Rightarrow reach(\rho, c, P, l+1)$

Sees-role Properties:

$\models role(\rho, c, P) \wedge sees(\rho, X) \wedge fresh(\rho, c, X) \wedge \bigwedge_{M \in Sees(P,\ l)} \neg(X = \overline{M}^{\rho,c}) \Rightarrow reach(\rho, c, P, l+1)$.

Freshness Properties: For $\mu = (\tau, c')$, $\{M\}_K \in Recg(Q, l)$

1. $\models fresh(\rho, c, c)$
2. $\models fresh(\rho, c, X_i) \Rightarrow fresh(\rho, c, [X_1, \ldots, X_m])$.
3. $\models \left(\begin{array}{l} reach(\tau, c', Q, l) \wedge fresh(\rho, c, \overline{M}^{\mu}) \wedge \\ poss(\rho, c, \overline{M}^{\mu}) \wedge (poss(\rho, c, \overline{K}^{\mu}) \vee poss(\rho, c, \overline{K^{-1}}^{\mu})) \end{array} \right) \Rightarrow fresh(\rho, c, \overline{\{M\}_K}^{\mu})$.
4. $\models \left(\begin{array}{l} reach(\tau, c', Q, l) \wedge fresh(\rho, c, \overline{K}^{\mu}) \wedge \\ poss(\rho, c, \overline{M}^{\mu}) \wedge (poss(\rho, c, \overline{K}^{\mu}) \vee poss(\rho, c, \overline{K^{-1}}^{\mu})) \end{array} \right) \Rightarrow fresh(\rho, c, \overline{\{M\}_K}^{\mu})$.
5. $\models fresh(\rho, c, \overline{T}^{\rho,c})$, where T is a label message term.

Monotonicity of Secrecy : If $\{\rho_1, \cdots, \rho_n\} \subseteq \{\tau_1, \cdots, \tau_{n'}\}$, then

$$\models secr(\rho_1, \cdots, \rho_n, X) \Rightarrow secr(\tau_1, \cdots, \tau_{n'}, X).$$

A.2 Kernel Properties

Proposition A.1 (Label Message Existence)

$$\models \forall v \exists v' \left(\bigvee_{P \in AG,\ N \in Poss(P, l_P)} role(\rho, v, P) \Rightarrow (\overline{N}^{\rho,v} = v') \right)$$

where N is a label message term.

To remove the quantifiers in the above formulas, we introduce Skolem functions g_ρ^N in LLS, and have that

$$\models \bigvee_{P \in AG,\ N \in Poss(P, l_P)} role(\rho, v, P) \Rightarrow (\overline{N}^{\rho,v} = g_\rho^N(v))$$

Role Assumption:

1. If $Supp_1^\tau \in \mathcal{S}$ then
$$\models_{\mathcal{S}} (said(\tau, c) \vee sees(\tau, c)) \Rightarrow \bigvee_{Q \in AG} role(\tau, c, Q)$$
2. If $Supp_2^{\tau,Q} \in \mathcal{S}$ then
$$\models_{\mathcal{S}} (said(\tau, c) \vee sees(\tau, c)) \Rightarrow role(\tau, c, Q).$$

Signature Properties: Given P, ρ, Q, τ, c, l. Suppose K is a public key or hash term, $M_i = \{M'\}_K \in Recg(Q, l)$ and $M = \{[M_1, \cdots, M_i, \cdots, M_h]\}_{K_P^{-1}} \in Poss(Q, l)$. Then, if $supp_4^\rho \in \mathcal{S}$, we have that

$$\models reach(\tau, c, l) \wedge \overline{P}^{\tau,c} = \rho \wedge said(\rho, \overline{M}^{\tau,c}) \Rightarrow poss(\rho, \overline{M'}^{\tau,c})$$

$(n,1)$**-Secrecy Properties:** Given P_i, ρ_i, c_i, and l_i $(i=0,\ldots,n)$. And $M \in Recvs(P_0, l_{P_0})$ and l_0 be the least l such that $M \in Recvs(P_0,l)$. We have the following:

$$\models \begin{array}{l} reach(\rho_0,c_0,P_0,l_0) \wedge \bigwedge_{0<i\leq n} rch(\rho_i,c_i,P_i,l_i) \Rightarrow \\ GS(P_0,\rho_0,c_0,P_1,\rho_1,c_1,l_1\ldots,P_n,\rho_n,c_n,l_n,M) \end{array}$$

where $GS(P_0,\rho_0,c_0,P_1,\rho_1,c_1,l_1\ldots,P_n,\rho_n,c_n,l_n,M)$ (or $GS(M)$ for short) is a formula, which intuitively indicates that someone other than ρ_i $(i=1,\ldots,n)$ can possess $\overline{M}^{\rho_0,c_0}$.

Secrecy-Derivation Given P_i, ρ_i, c_i, and l_i $(i=0,\ldots,n)$. Suppose $M \in Poss(P_0,l_0)$. Then, we have that

$$\models \begin{array}{l} reach(\rho_0,c_0,P_0,l_0) \wedge \bigwedge_{0<i\leq n} rch(\rho_i,c_i,P_i,l_i) \Rightarrow \\ secr(\rho_1,\cdots,\rho_n,\overline{M}^{\rho_0,c_0}) \vee GS(P_0,\rho_0,c_0,P_1,\rho_1,c_1,l_1\ldots,P_n,\rho_n,c_n,l_n,M) \end{array}$$

A.3 Equivalence and Comparison Properties

Equivalence Properties:

1. $\models X_1 = X_2 \Rightarrow (\Delta(\rho,X_1) \Leftrightarrow \Delta(\rho,X_2))$,
   where Δ stands for *poss*, *sees*, and *said*.
2. $\models X_1 = X_2 \Rightarrow (fresh(\rho,c,X_1) \Leftrightarrow fresh(\rho,c,X_2))$
3. $\models (X_1 = X_2) \wedge (X_2 = X_3) \Rightarrow (X_1 = X_3)$

Comparison Properties:

1. $\models (\overline{K}^{\tau,c_2} = \overline{K}^{\rho,c_1}) \Leftrightarrow (\overline{K^{-1}}^{\tau,c_2} = \overline{K^{-1}}^{\rho,c_1})$
   where K is the public key of role P.
2. $\models (\overline{P}^{\tau,c_2} = \overline{P}^{\rho,c_1}) \Leftrightarrow (\overline{K}^{\tau,c_2} = \overline{K}^{\rho,c_1})$,
   where K is the public key of role P.
3. $\models \ (role(\rho,c_1,P) \wedge role(\tau,c_2,Q) \wedge (\overline{P}^{\tau,c_2} = \rho)) \Rightarrow ((\overline{Q}^{\rho,c_1} = \tau) \Leftrightarrow \overline{K}^{\rho,c_1} = \overline{K}^{\tau,c_2})$
   where K is the shared secret of P and Q.
4. $\models \ (\overline{K_{PQ}}^{\tau,c_2} = \overline{K_{PQ}}^{\rho,c_1}) \Leftrightarrow \left(\begin{array}{l} ((\overline{P}^{\tau,c_2} = \overline{P}^{\rho,c_1}) \wedge (\overline{Q}^{\tau,c_2} = \overline{Q}^{\rho,c_1})) \vee \\ ((\overline{P}^{\tau,c_2} = \overline{Q}^{\rho,c_1}) \wedge (\overline{Q}^{\tau,c_2} = \overline{P}^{\rho,c_1})) \end{array} \right)$
   where K_{PQ} is the shared secret of P and Q.
5. $\models role(\rho,c,P) \Rightarrow (\overline{K_{PQ}^{-1}}^{\rho,c} = \overline{K_{PQ}}^{\rho,c})$
   where K is the shared key between P and Q.
6. $\models poss(\rho,\overline{K}^{\rho,c}) \Rightarrow (\overline{K^{-1}}^{\rho,c} = \overline{K}^{\rho,c})$ where K is a temporally created share key.
7. $\models [X_1,\ldots,X_m] = [Y_1,\ldots,Y_{m'}] \Leftrightarrow \bigvee_{[Y_1,\ldots,Y_{m'}]\approx[M_1,\ldots,M_m]} \left(\bigwedge_{1\leq i\leq m} X_i = M_i \right)$.
8. $\models reach(\rho,c_1,P,l_1) \wedge reach(\tau,c_2,Q,l_2) \Rightarrow \left(\begin{array}{l} (\overline{\{M\}_K}^{\rho,c_1} = \overline{\{M'\}_{K'}}^{\tau,c_2}) \Leftrightarrow \\ (\overline{M}^{\rho,c_1} = \overline{M'}^{\tau,c_2} \wedge \overline{K}^{\rho,c_1} = \overline{K'}^{\tau,c_2}) \end{array} \right)$,
   where $\{M\}_K \in Recg(P,l_1)$ and $\{M'\}_{K'} \in Recg(Q,l_2)$.

9. $\models$ $(poss(\tau, \overline{M}^{\rho,c_1}) \wedge poss(\tau, \overline{K}^{\rho,c_1})) \vee poss(\tau, \overline{K^{-1}}^{\rho,c_1}) \Rightarrow$
$(\overline{\{M\}_K}^{\rho,c_1} \neq \overline{\{M'\}_{K'}}^{\tau,c_2}) \vee reach(\tau, c_2, Q, l_2 + 1)$,
where $\{M'\}_{K'} \notin Recg(Q, l_2)$.

10. $\models \neg \left(\overline{\{M\}_K}^{\rho,c_1} = \overline{M'}^{\tau,c_2} \right)$,
where M' is either a plain text or concatenation of more than two messages.

11. $\models reach(\tau, c_2, Q, l) \Rightarrow \neg \left(\overline{M}^{\tau,c_2} = \overline{M'}^{\tau,c_2} \right)$,
where $M, M' \in Poss(Q, l)$ and it is assumed in the protocol that M differs from M'.

12. $\models \neg(\rho_1 = \rho_2) \wedge \neg(c_1 = c_2) \wedge \neg(\rho = c)$,
where ρ_1 differs from ρ_2 and c_1 differs from c_2.

13. $\models \neg(\rho = \overline{M_1}^{\tau,c}) \wedge \neg(c_1 = \overline{M_2}^{\tau,c_2})$,
where M_1 is a message other than roles and M_2 is a message other than nonces.

B More on SPV

B.1 Characteristics of SPV

The newest version of SPV is 3.8.2. It's now orders of magnitude faster than the first version. It has been designed and constructed robustly. Its main components are:

1. Analyze the input file, convert it into various related data-structures.
2. Generate all the properties that are needed.
3. Utilize the powerful SAT solver SBSAT [22] to compute the specification.

In all of the components, many techniques have been used for speedup, such as on-the-fly techniques to generate the variables, lookup-table to take out the constraints. A good result has been gained. For example, preliminary results for the Kerberos V5 protocol [36] required nearly 1 hour, but now after optimisations it only takes 6 minutes.

The accepted input file is defined very clearly and intuitively. It provides many powerful features:

1. Macros are supported. Users can define macros by themselves, and this simplifies the input process for big protocols. (Macro definitions begin with `#macro`).
2. Define all the variables needed in the protocol, such as agent, nonce and key etc. (Variable definitions begin with `#variable`).
3. Initialize all relations among the variables defined above. (Initialization section begins with `#initialize`).
4. The protocol body and the format are also kept simple, an example is given below. (The protocol body begins with `#protocol`).
5. The definition of role assumptions: some details are provided in the paper. (The role definition begins with `#role_assumption`).

6. The definition part: here the users can define their assumptions a little different from that in the paper. (It begins with `#definition`).

7. The specification part: here users can describe all the specifications they want to verify. (It begins with `#specification`).

B.2 Verifying the SET Purchase Phase Protocol with SPV

As an example, we give the input file of the SET purchase phase protocol. The definition is based on that in [8]. We do not intend to offer more detailed analysis as done in [8] with Isabelle; what we want know from these experiments is whether the automatic justification-oriented approach can be scaled to the size of the SET purchase phase protocol.

```
#macro
HOD = cryp(hash,(OrderDesc,PurchAmt));

PIHead = LID_M,XID,HOD,PurchAmt,M,cryp(hash,(XID,CardSecret));
OIData = XID,Chall_C,HOD,Chall_M;
PANData = PAN,PANSecret;
PIData = (PIHead,PANData);

PID_SIGN =cryp(_kc,(cryp(hash,PIData),cryp(hash,(OIData)))),
          cryp(kp,(PIHead,cryp(hash,(OIData)),PANData));

OID_SIGN = OIData,cryp(hash,PIData);

#variable agent : cust,mer,paygate;
agent_term : C,M,P;
nonce :chall_cust,chall_mer;
nonce_term : Chall_C,Chall_M;
gussable_lable_message : xid,lid_m; g
ussable_lable_message_term :XID,LID_M;
publicKey_term : km,kc,kp; privateKey_term:_km,_kc,_kp;
generalMessage_term : OrderDesc,PurchAmt,AuthCode,PAN;

shareSecrecy_term : CardSecret,PANSecret;

#initialize
name(cust,C);
name(mer,M);
name(paygate,P);
name(chall_cust,Chall_C);
name(chall_mer,Chall_M);
name(xid,XID);
name(lid_m,LID_M);
```

```
nonce(C,Chall_C);
nonce(M,Chall_M);

gussable_lable_message(M,XID);
gussable_lable_message(M,LID_M);

inverse_key(C,kc,_kc);
inverse_key(M,km,_km);
inverse_key(P,kp,_kp);

initial_general_message(OrderDesc,C);
initial_general_message(OrderDesc,M);
initial_general_message(PurchAmt,C);
initial_general_message(PurchAmt,M);
initial_general_message(PAN,C);
initial_general_message(PAN,P);
initial_general_message(AuthCode,P);


share_secrecy(C,P,CardSecret);
share_secrecy(C,P,PANSecret);

#protocol
C,M : M,Chall_C;
M,C : cryp(_km,(M,XID,Chall_C,Chall_M));
C,M : PID_SIGN,OID_SIGN;
M,P : cryp(kp,cryp(_km,(LID_M, XID,cryp(hash,(OIData)),HOD,PID_SIGN)));
P,M : cryp(km,cryp(_kp,(LID_M,XID,PurchAmt,AuthCode)));
M,C : cryp(_km,(LID_M, XID,Chall_C,cryp(hash,(PurchAmt))));

#role_assumption

sees(cust,chall_cust)+said(cust,chall_cust)
>role(cust,chall_cust,C);
sees(cust,chall_mer)+said(cust,chall_mer)
>role(cust,chall_mer,C);
sees(mer,chall_cust)+said(mer,chall_cust)
>role(mer,chall_cust,M);
sees(mer,chall_mer)+said(mer,chall_mer)
>role(mer,chall_mer,M);
sees(cust,xid)+said(cust,xid)
>role(cust,xid,C);
sees(mer,xid)+said(mer,xid)
>role(mer,xid,M);
sees(paygate,xid)+said(paygate,xid)
>role(paygate,xid,P);
```

```
#define normal_state = 1;


#specification
know(cust,(said(mer,chall_cust)));
know(mer,(said(cust,chall_mer)));
know(cust,know(mer,(said(cust,chall_mer))));
know(mer,know(cust,(said(mer,chall_cust))));

know(cust,(role(mer,xid,M)));
know(cust,(role(paygate,xid,P)));
know(mer,(role(cust,xid,C)));
know(mer,(role(paygate,xid,P)));
know(paygate,(role(cust,xid,C)));
know(paygate,(role(mer,xid,M)));

know(mer,(equal(local(cust,xid,OrderDesc),local(mer,xid,OrderDesc))));
know(mer,(equal(local(cust,xid,PurchAmt),local(mer,xid,PurchAmt))));
know(paygate,(equal(local(cust,xid,OrderDesc),local(paygate,xid,OrderDesc))));
know(paygate,(equal(local(cust,xid,PurchAmt),local(paygate,xid,PurchAmt))));


#goal
secret OrderDesc;
secret PurchAmt;

precedes C : P | XID,CardSecret;
precedes P : C | XID,CardSecret;
precedes C : M | XID,OrderDesc;
precedes M : C | XID,OrderDesc;
precedes C : M | XID,PurchAmt;
precedes M : C | XID,PurchAmt;
precedes M : P | XID,OrderDesc;
precedes P : M | XID,OrderDesc;
precedes M : P | XID,PurchAmt;
precedes P : M | XID,PurchAmt;
```

The experiments have been conducted on a PC with AMD Opteron242, 2GB DDR memory, running under Redhat Linux 7.0 with gcc 3.2.2. The following gives the results during dealing with the input and the generating all the properties before verifications of the specifications. (The unit for time is seconds)

Variables	Properties Generated	Total Time
147839	222563	224630

We remark that if we slightly simplify the SET purchase protocol proposed by [8], the running time can be significantly reduced. We omit "LID_M" in the SET purchase protocol (which is optional according to Formal Protocol Definition of SET (Version 1.0)) and unify PAN, CardSecret and PanSecret,

and then obtain a simplified version of this protocol. For the simplified version, the running is 10 times reduced as shown in the following table:

Variables	Properties Generated	Total Time
61065	83747	31364

For both version of SET purchase protocol, we have verified the properties as shown in the following table:

Specification	Yes/No
$K_{cust}said(mer, chall_cust)$	Yes
$K_{mer}said(cust, chall\_mer)$	Yes
$K_{cust}role(mer, xid, M)$	Yes
$K_{cust}role(paygate, xid, P)$	No
$K_{mer}role(cust, xid, C)$	Yes
$K_{mer}role(paygate, xid, P)$	Yes
$K_{paygate}role(cust, xid, C)$	Yes
$K_{paygate}role(mer, xid, M)$	Yes
$K_{mer}(local(cust, xid, OrderDesc)$ $= local(mer, xid, OrderDesc))$	Yes
$K_{mer}(local(cust, xid, PurchAmt)$ $= local(mer, xid, PurchAmt))$	Yes
$K_{paygate}(local(cust, xid, OrderDesc)$ $= local(paygate, xid, OrderDesc))$	No
$K_{paygate}(local(cust, xid, PurchAmt)$ $= local(paygate, xid, PurchAmt))$	Yes
$K_{cust}K_{mer}said(cust, chall\_mer)$	No
$K_{mer}K_{cust}said(mer, chall\_cust)$	Yes
$secrecy(cust, mer, local(cust, chall\_cust, OrderDesc))$	Yes
$secrecy(cust, mer, paygate, local(cust, chall\_cust, PurchAmt))$	Yes
$\mathbf{P}^{cust}_{paygate}(xid, CardSecret)$	Yes
$\mathbf{P}^{paygate}_{cust}(xid, CardSecret)$	No
$\mathbf{P}^{cust}_{mer}(xid, OrderDesc)$	Yes
$\mathbf{P}^{mer}_{cust}(xid, OrderDesc)$	No
$\mathbf{P}^{cust}_{mer}(xid, PurchAmt)$	Yes
$\mathbf{P}^{mer}_{cust}(xid, PurchAmt)$	Yes
$\mathbf{P}^{mer}_{paygate}(xid, OrderDesc)$	No
$\mathbf{P}^{paygate}_{mer}(xid, OrderDesc)$	No
$\mathbf{P}^{mer}_{paygate}(xid, PurchAmt)$	Yes
$\mathbf{P}^{paygate}_{mer}(xid, PurchAmt)$	No

Here the property such as

$$K_{mer}role(cust, xid, C)$$

means that the $mer(merchant)$ observes that it is $cust(customer)$ who plays the role of C that the protocol defines, with the challenge number xid.

The property such as

$$K_{mer}(local(cust, xid, OrderDesc) = local(mer, xid, OrderDesc))$$

means that the $mer(merchant)$ observes that the $OrderDesc(OrderDescription)$ in the local run of $cust$ with the challenge number xid is exactly the same as in the local run of mer with challenge number xid. This is a more intuitive way to depict the *matching* in [20] and *correspondence* in [41], which is very important in authentication.

The property such as

$$K_{cust}K_{mer}said(cust, chall\_mer)$$

means that $cust(customer)$ observes that $mer(merchant)$ have observed that it is $cust(customer)$ who said the $chall\_mer$. This kind of nested specifications is a rather strong assertion since it concerns about other agents' observations from the reasoning of one agent's observations.

Once the common knowledge base of all the properties have been generated, the verifications of the properties can be reduced to SAT solving. By the power of the SAT solver SBSAT [22], we gain substantial efficiency from the experiments. We can see that most of the time are spent on the generations of variables and properties, so the time for verifications is just within seconds. This means that there will be lots of scope for further optimizations and scaling to utilize the power of the underlying SBSAT [22].

Furthermore, we have given more features and flexibilities in the user-interface. We permit the users to decide in the following way if they want to accept the *norm* to be in the common knowledge as defined in $Supp_3$.

```
normal_state = x;  //here  x=0 means the users do not accept; x=1
//means they do; others are illegal input. The default value is 1;
```

We also permit the users to define each agent's observations by the following means:

```
observer_variable(agent)={ a; b; c; ... };
// If the users do not define them,the default observations for the agent
//are given by the definition in the paper.
```

And SPV can let the users design their own specific definitions for each specification to be verified.

```
#define  //1 ... #specification //2 ... #define  //3 ...
#specification //4
 ...
//The verification of specification 2 will be on the basis of definition 1 and 4 of the 3.
```

SPV is powerful and friendly to users. It is open, flexible and scalable. We believe that this tool will be an interesting contribution to the security verification and formal methods communities.

B.3 Some other examples

We have tested almost all protocols in Clark-Jacob Libary. Here we give inputs and outputs for some of them with SPV.

B.3.1 Needham-Schroeder

```
// Needham-Schroeder
#variable
   agent_name : alice,bob;
   agent: A,B;
```

```
   nounce_name : na,nb;
   nounce: Na,Nb;
   publicKey: ka,kb;
   privateKey: _ka,_kb;
#initialize
   name(alice,A);
   name(bob,B);
   name(na,Na);
   name(nb,Nb);
   nounce(A,Na);
   nounce(B,Nb);
   inverse_key(A,ka,_ka);
   inverse_key(B,kb,_kb);

#protocol
A,B : cryp(kb,(A,Na));
B,A : cryp(ka,(Na,Nb));
A,B : cryp(kb,Nb);

#role_assumption
sees(alice,nb)+said(alice,nb) > role(alice,nb,A);
sees(bob,na)+said(bob,na) > role(bob,na,B);

#specification
know(alice,(said(bob,na)));
know(bob,(said(alice,nb)));
know(alice,know(bob,(said(alice,nb))));
know(bob,know(alice,(said(bob,na))));

#goal
secret Na;
secret Nb;
precedes   A : B | Na, Nb;
precedes B : A | Na, Nb;


real    0m51.120s user    0m48.120s sys     0m2.700s

We spend 42.1 seconds to generate all the constrains! the number
of variables is 3548 and the number of constrains is 6296


==========**********==========
know(alice,said(bob,na)) OK, 0.04 seconds

know(bob,said(alice,nb)) OK, 0.04 seconds
```

```
know(alice,know(bob,said(alice,nb))) OK, 0.38 seconds

know(bob,know(alice,said(bob,na))) NO, 0.42 seconds


==================================secrecy=================================
--------------secrecy(alice,bob,local(alice,na,Na))-----------------
OK, 0.34 seconds

==================================secrecy=================================
--------------secrecy(bob,alice,local(bob,nb,Nb))-----------------
FAIL, 0.36 seconds

=================================precedes=================================
--reach(bob,nb,B,3)*equal(alice,local(bob,nb,A))*equal(na,local(bob,nb,Na)))>
(role(alice,na,A)*equal(bob,local(alice,na,B))*equal(nb,local(alice,na,Nb))*
equal(local(bob,nb,Nb),local(alice,na,Nb)))-----

------------
FAIL, 0.05 seconds

=================================precedes=================================
---reach(alice,na,A,3)*equal(bob,local(alice,na,B))*equal(na,local(alice,na,Na)))>
(role(bob,na,B)*equal(alice,local(bob,na,A))*equal(na,local(bob,na,Na))*
equal(local(bob,na,Nb),local(alice,na,Nb)))---------

--------
OK, 0.05 seconds
```

B.3.2 Rev Needham-Schroeder

```
// Rev Needham-Schroeder
#variable
agent_name : alice,bob;
agent : A,B;
nounce_name : na,nb;
nounce : Na,Nb;
publicKey : ka,kb;
privateKey : _ka,_kb;

#initialize
name(alice,A); name(bob,B); name(na,Na); name(nb,Nb);
nounce(A,Na); nounce(B,Nb);
inverse_key(A,ka,_ka);
inverse_key(B,kb,_kb);

#protocol
```

```
A,B : cryp(kb,(A,Na));
B,A : cryp(ka,(B,Na,Nb));
A,B : cryp(kb,Nb);

#role_assumption
sees(alice,nb)+said(alice,nb) > role(alice,nb,A);
sees(bob,na)+said(bob,na) > role(bob,na,B);

#define normal_state = 1;

#specification
know(alice,(said(bob,na)));
know(bob,(said(alice,nb)));
know(alice,know(bob,(said(alice,nb))));
know(bob,know(alice,(said(bob,na))));

#goal
secret Na; secret Nb;
precedes   A : B | Na, Nb;
precedes B : A | Na, Nb;

real    0m54.238s user    0m46.530s sys     0m2.990s


We spend 43.38 seconds to generate all the constrains! the number
of variables is 3548 and the number of constrains is 6332
============================================================
know(alice,said(bob,na)) OK, 0.06 seconds

know(bob,said(alice,nb)) OK, 0.05 seconds

know(alice,know(bob,said(alice,nb))) OK, 0.42 seconds

know(bob,know(alice,said(bob,na))) OK, 0.35 seconds

================================********secrecy*********================================
--------------secrecy(alice,bob,local(alice,na,Na))----------------
OK, 0.37 seconds

================================********secrecy*********================================
--------------secrecy(bob,alice,local(bob,nb,Nb))-----------------
OK, 0.36 seconds

================================********precedes*********================================
---reach(bob,nb,B,3)*equal(alice,local(bob,nb,A))*equal(na,local(bob,nb,Na)))>
(role(alice,na,A)*equal(bob,local(alice,na,B))*equal(nb,local(alice,na,Nb))*
equal(local(bob,nb,Nb),local(alice,na,Nb)))----
```

```
-------------
OK,  0.04 seconds

============================********precedes**********================================
--reach(alice,na,A,3)*equal(bob,local(alice,na,B))*equal(na,local(alice,na,Na)))>
(role(bob,na,B)*equal(alice,local(bob,na,A))*equal(na,local(bob,na,Na))*
equal(local(bob,na,Nb),local(alice,na,Nb)))---------

--------
OK, 0.05 seconds
```

B.3.3 Denning-Sacco shared key

```
//Denning-Sacco shared key
//1.   A -> S :   A, B
//2.   S -> A :   {B, Kab, T, {Kab, A, T}Kbs}Kas
//3.   A -> B :   {Kab,A, T}Kbs

#variable
agent_name : alice,bob,server;
agent : A,B,S;
stamp_name: ts;
timeStamp : Ts;
shareKey : Kas,Kbs;
dynamicKey :Kab;

#initialize
name(alice,A); name(bob,B); name(server,S);name(ts,Ts);
time_stamp(S,Ts);
dynamic_key(S,Kab);
share_key(A,S,Kas);
share_key(B,S,Kbs);

#protocol
A,S : A,B;
S,A :cryp(Kas,(B,Kab,Ts,cryp(Kbs,(Kab,A,Ts))));
A,B :cryp(Kbs,(Kab,A,Ts));

#role_assumption
said(alice,ts)+ sees(alice,ts) >role(alice,ts,A);

sees(bob,ts) > role(bob,ts,B);

said(server,ts) > role(server,ts,S);
```

```
#specification
know(alice,(sees(bob,ts))); //
know(bob,(said(alice,ts))); //
know(alice,know(bob,(said(alice,ts)))); //
know(bob,know(alice,(sees(bob,ts)))); //

#goal
secret Kab;

precedes A : B | Ts,Kab;

precedes B : A | Ts,Kab;


real    0m25.881s user    0m22.280s sys     0m0.860s

We spend 16.08 seconds to generate all the constrains! the number
of variables is 2178 and the number of constrains is 2905

============================================================
 know(alice,sees(bob,ts))
NO, 0.02 seconds

know(bob,said(alice,ts)) NO, 0.02 seconds

know(alice,know(bob,said(alice,ts))) NO, 0.1 seconds

know(bob,know(alice,sees(bob,ts))) NO,  0.11 seconds

==============================********secrecy**********==============================
--------------secrecy(server,alice,bob,local(server,ts,Kab))----------------
OK, 0.21 seconds

==============================********precedes**********==============================
---reach(bob,ts,B,1)*equal(alice,local(bob,ts,A))*equal(ts,local(bob,ts,Ts)))>
(role(alice,ts,A)*equal(bob,local(alice,ts,B))*equal(ts,local(alice,ts,Ts))*
equal(local(bob,ts,Kab),local(alice,ts,Kab)))---

--------------

FAIL, 0.01 seconds
==============================********precedes**********==============================
--reach(alice,ts,A,3)*equal(bob,local(alice,ts,B))*equal(ts,loca(alice,ts,Ts)))>
(role(bob,ts,B)*equal(alice,local(bob,ts,A))*equal(ts,local(bob,ts,Ts))*
equal(local(bob,ts,Kab),local(alice,ts,Kab)))-------

----------
```

```
FAIL, 0.01 seconds
```

B.3.4 Woo and Lam Pi

```
//Woo and Lam Pi
//1..   A -> B :   A
//2..   B -> A :   Nb
//3..   A -> B :   {A,B,Nb}shared(A, S)
//4..   B -> S :   {A, B, Nb, {A, B, Nb}shared(A, S)}shared(B, S)
//5..   S -> B :   {A, B, Nb}shared(B, S)
#variable
agent_name : alice,bob,server;
agent : A,B,S;
nounce_name: nb;
nounce : Nb;
shareKey : Kas,Kbs;

#initialize

name(alice,A); name(bob,B); name(server,S); name(nb,Nb);

nounce(B,Nb); share_key(A,S,Kas); share_key(B,S,Kbs);

#protocol

A,B : A;

B,A : Nb;

A,B : cryp(Kas,(A,B,Nb));

B,S : cryp(Kbs, (A,B,Nb,cryp(Kas,(A,B,Nb))));

S,B : cryp(Kbs,(A,B,Nb));

#role_assumption

said(alice,nb)+ sees(alice,nb) > role(alice,nb,A);
sees(bob,nb)+said(bob,nb) > role(bob,nb,B);
said(server,nb)+sees(server,nb) > role(server,nb,S);
```

```
#specification know(alice,(said(bob,nb)));
know(bob,(said(alice,nb)));
know(alice,know(bob,(said(alice,nb))));
know(bob,know(alice,(said(bob,nb))));

#goal
secret Nb;
precedes A : B | Nb,Nb;
precedes B : A | Nb,Nb;


real    0m27.782s user    0m24.770s sys     0m1.000s

We spend 18.61 seconds to generate all the constrains! the number
of variables is 1984 and the number of constrains is 3248

============================================================================
know(alice,said(bob,nb)) NO, 0.02 seconds

know(bob,said(alice,nb)) OK, 0.02 seconds

know(alice,know(bob,said(alice,nb))) NO, 0.09 seconds

know(bob,know(alice,said(bob,nb))) NO, 0.1 seconds


================================********secrecy**********=================================
--------------secrecy(bob,alice,server,local(bob,nb,Nb))-----------------
FAIL, 0.21 seconds

================================********precedes**********=================================
----reach(bob,nb,B,5)*equal(alice,local(bob,nb,A))*equal(nb,local(bob,nb,Nb)))>
(role(alice,nb,A)*equal(bob,local(alice,nb,B))*equal(nb,local(alice,nb,Nb))*
equal(local(bob,nb,Nb),local(alice,nb,Nb)))-----

------------
OK, 0.02 seconds

================================********precedes**********=================================
--reach(alice,nb,A,3)*equal(bob,local(alice,nb,B))*equal(nb,local(alice,nb,Nb)))>
(role(bob,nb,B)*equal(alice,local(bob,nb,A))*equal(nb,local(bob,nb,Nb))*
equal(local(bob,nb,Nb),local(alice,nb,Nb)))---------

--------
FAIL, 0.02 seconds
```

B.3.5 Kao Chow Authentication v.1

```
//Kao Chow Authentication v.1
//1.   A -> S :   A, B, Na
//2.   S -> B :   {A, B, Na, Kab}Kas, {A, B, Na, Kab}Kbs
//3.   B -> A :   {A, B, Na, Kab}Kas, {Na}Kab, Nb
//4.   A -> B :   {Nb}Kab

#variable
agent_name : alice,bob,server;
agent : A,B,S;
nounce_name: na,nb;
nounce : Na,Nb;
shareKey : Kas,Kbs;
dynamicKey : Kab;

#initialize
name(alice,A); name(bob,B); name(server,S); name(na,Na); name(nb,Nb); nounce(A,Na);
nounce(B,Nb);
dynamic_key(S,Kab);
share_key(A,S,Kas);
share_key(B,S,Kbs);

#protocol
A,S : A,B,Na;
S,B :cryp(Kas,(A,B,Na,Kab)),cryp(Kbs,(A,B,Na,Kab));
B,A : cryp(Kas,(A,B,Na,Kab)),cryp(Kab,Na),Nb;
A,B : cryp(Kab,Nb);

#role_assumption

said(alice,nb)+ sees(alice,nb) > role(alice,nb,A);
said(alice,na)+sees(alice,na) > role(alice,na,A);
sees(bob,na)+ said(bob,na) > role(bob,na,B);
said(bob,nb)+sees(bob,nb) > role(bob,nb,B);
sees(server,na)+said(server,na) > role(server,na,S);

#specification
know(alice,(said(bob,na)));
know(bob,(said(alice,nb)));
know(alice,know(bob,(said(alice,nb))));
know(bob,know(alice,(said(bob,na))));

#goal
secret Na; secret Nb; secret Kab;
precedes   A : B | Na, Nb;
```

```
precedes   B : A | Na, Nb;

real    5m16.218s user    5m4.690s sys     0m4.000s

We spend 275.66 seconds to generate all the constrains! the number
of variables is 8179 and the number of constrains is 14316


============================================================

know(alice,said(bob,na)) OK, 0.1 seconds

know(bob,said(alice,nb)) OK, 0.1 seconds

know(alice,know(bob,said(alice,nb))) OK, 0.99 seconds

know(bob,know(alice,said(bob,na))) OK, 0.84 seconds

==========================*********secrecy**********=====================
--------------secrecy(alice,bob,server,local(alice,na,Na))----------------

FAIL, 1.5 seconds

==========================*********secrecy**********=====================
--------------secrecy(bob,alice,server,local(bob,nb,Nb))-----------------
FAIL, 2.12 seconds


==========================********precedes**********=====================
--------------secrecy(server,alice,bob,local(server,na,Kab))-----------------
OK, 0.98 seconds


==========================********precedes**********=====================

---reach(bob,nb,B,3)*equal(alice,local(bob,nb,A))*equal(na,local(bob,nb,Na)))>
(role(alice,na,A)*equal(bob,local(alice,na,B))*equal(nb,local(alice,na,Nb))*
equal(local(bob,nb,Nb),local(alice,na,Nb)))-----

------------
OK, 0.1 seconds

==========================********precedes**********=====================

---reach(alice,na,A,3)*equal(bob,local(alice,na,B))*equal(na,local(alice,na,Na)))>
(role(bob,na,B)*equal(alice,local(bob,na,A))*equal(na,local(bob,na,Na))*
equal(local(bob,na,Nb),local(alice,na,Nb)))---------
```

```
--------

OK, 0.09 seconds
```

B.3.6 Kerberos V

```
// Kerberos V

#variable
agent_name : alice,bob,server;
agent : A,B,S;
stamp_name : ts,ta;
timeStamp : Ts,Ta;
shareKey : Kas,Kbs;
dynamicKey: Kab;

#initialize name(alice,A); name(bob,B); name(server,S);
name(ta,Ta); name(ts,Ts);
time_stamp(S,Ts);
time_stamp(A,Ta);
share_key(A,S,Kas);
share_key(B,S,Kbs);
 dynamic_key(S,Kab);

#protocol
A,S : A,B;
S,A : cryp(Kas,(Ts,Kab,B,cryp(Kbs,(Ts,Kab,A))));
A,B : cryp(Kbs,(Ts,Kab,A)),cryp(Kab,(A,Ta));
B,A : cryp(Kab,Ta);

#role_assumption
said(alice,ta)+sees(alice,ta) > role(alice,ta,A);
said(alice,ts)+sees(alice,ts) > role(alice,ts,A);
said(bob,ta)+sees(bob,ta) > role(bob,ta,B);
said(bob,ts)+sees(bob,ts) > role(bob,ts,B);
said(server,ts) > role(server,ts,S);

#specification
know(alice,(said(bob,ta))); //ok
know(bob,(said(alice,ta))); //ok
know(alice,know(bob,(said(alice,ta)))); //ok
```

```
know(bob,know(alice,(said(bob,ta)))); //ok

#goal
secret     Kab;
precedes A : B | Ts,Kab;
precedes B : A | Ts,Kab;

real    5m27.585s user    5m16.800s sys     0m3.300s

We spend 305.53 seconds to generate all the constrains! the number
of variables is 8834 and the number of constrains is 14946
==========************************************************
 know(alice,said(bob,ta)) OK, 0.08 seconds

know(bob,said(alice,ta)) OK, 0.07 seconds

know(bob,said(alice,ta)) OK, 0.88 seconds

know(bob,know(alice,said(bob,ta))) OK, 0.86 seconds


==================================secrecy=================================

--------------secrecy(server,alice,bob,local(server,ts,Kab))-----------------

OK, 1.45 seconds

==================================precedes================================

----reach(bob,ts,B,2)*equal(alice,local(bob,ts,A))*equal(ts,local(bob,ts,Ts)))>
(role(alice,ts,A)*equal(bob,local(alice,ts,B))*equal(ts,local(alice,ts,Ts))*
equal(local(bob,ts,Kab),local(alice,ts,Kab)))---

OK, 0.08 seconds


==================================precedes================================
---reach(alice,ta,A,4)*equal(bob,local(alice,ta,B))*equal(ts,local(alice,ta,Ts)))>
(role(bob,ts,B)*equal(alice,local(bob,ts,A))*equal(ta,local(bob,ts,Ta))*
equal(local(bob,ts,Kab),local(alice,ta,Kab)))----------------

OK, 0.08 seconds
```