***Journal of Information Security*, 2010, 1, 45-55**
doi:10.4236/jis.2010.12006 Published Online October 2010 (http://www.SciRP.org/journal/jis)

Extending the Strand Space Method with Timestamps: Part I the Theory*

Yongjian Li[1,2], Jun Pang[3]
[1]*Chinese Academy of Sciences, Institute of Software, Laboratory of Computer Science, Beijing, China*
[2]*The State Key Laboratory of Information Security, Beijing, China*
[3]*University of Oldenburg, Department of Computer Science, Safety-critical Embedded Systems, Oldenburg, Germany*
E-mail: *lyj238@ios.ac.cn, jun.pang@informatik.uni-oldenburg.de*
Received June 23, 2010; *revised September* 14, 2010; *accepted July* 12, 2010

Abstract

In this paper, we present two extensions of the strand space method to model Kerberos V. First, we include time and timestamps to model security protocols with timestamps: we relate a key to a crack time and combine it with timestamps in order to define a notion of recency. Therefore, we can check replay attacks in this new framework. Second, we extend the classic strand space theory to model protocol mixture. The main idea is to introduce a new relation $\mapsto$ to model the causal relation between one primary protocol session and one of its following secondary protocol session. Accordingly, we also extend the definition of unsolicited authentication test.

Keywords: Strand Space, Kerberos V, Theorem Proving, Verification, Isabelle/HOL

1. Introduction

The strand space model [1] is a formal approach to reasoning about security protocols. For a legitimate regular participant, a strand s represents a sequence of messages that the participant would receive or send as part of a run as his/her role of the protocol. A typical message has the form of $\{|h|\}_K$ denoting the encryption of h using key K. An element of the set of messages is called a *term*. A term t' is a subterm of t is written as $t' \sqsubset t$. Usually, we call a strand element *node*. Nodes can be either positive, representing the transmission of a term, or negative, representing the reception of a term. For the penetrator, the strand represents atomic deductions. More complex deductions can be formed by connecting several penetrator strands. Hence, a strand space is simply a set of strands with a trace mapping. Two kinds of causal relation (arrow), $\rightarrow$ and $\Rightarrow$, are introduced to impose a graphic structure on the nodes of the space. The relation $\preceq$ is defined to be the reflexive and transitive closure of these two arrows, modelling the causal order of the events in the protocol execution. The formal analysis based on strand spaces can be carried on the notion of bundles. A bundle is a causally well-founded set of nodes and the two arrows, which sufficiently formalizes a session of a protocol. In a bundle, it must be ensured that a node is included only if all nodes that proceed it are already included. For the strand corresponding to a principal in a given protocol run, we construct all possible bundles containing nodes of the strand. In fact, this set of bundles encodes all possible interactions of the environment with that principal in the run. Normally, reasoning about the protocol takes place on this set of bundles.

However, the original strand space model has its semantical limitations to analyze the real-world protocols such as Kerbeoros protocols. First, it does not include timestamps as formalized message components, and therefore can not model security protocols with timestamps. In fact, the strand space model [1] as given by Thayer Fábrega, Herzog, and Guttman is only benchmarked on nonce-based protocols such as the Needham-Schroeder protocol and the Otway-Rees protocol. But many modern protocols use timestamps to prevent replay attacks, so this deficiency of the strand space theory makes it difficult to analyze these protocols. Second, it

*This is a revised and extended version of the homonymous paper appearing in the Proceedings the Eighth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007, IEEE Computer Society). The main modifications have been made on the presentation of the technical material, with the purpose of having full details. The first author is supported by grants (No.60496321, 60421001) from National Natural Science Foundation of China.

does not address issues of the protocol dependency when several protocols are mixed together. Many real-world protocols are divided into causally related multiple phases (or subprotocols), such as the Kerberos and Neuman-Stubblebine protocols. One phase may be used to retrieve a ticket from a key distribution center, while a second phase is used to present the ticket to a security-aware server. To make matters more complex, many protocols such as Kerbeors use timestamps to guarantee the recency of these tickets, that is, such tickets are only valid for an interval, and multiple sub-protocol sessions can start in parallel by the same agent using the same ticket if the ticket does not expire. Little work has been done to formalize the causal relation between protocols in a protocol mixture environment.

The aim of this paper is twofold. The first aim is to extend the strand space theory to cover the aforementioned two semantical features. Briefly, we include time and timestamps to model security protocols with timestamps: we relate a key to a crack time and combine it with timestamps in order to define a notion of recency. Therefore, we can check replay attacks in this new framework. We also extend the classic strand space theory to model protocol mixture: a new relation $\mapsto$ is introduced to model the causal relation between one primary protocol session and one of its following secondary protocol session. Despite the extensions, we hope that the extended theory still maintains the simple and powerful mechanism to reason about protocols. The second aim is practical. We hope to apply the extended theory to the analysis of some real-world protocols. Here we select Kerberos V as our case study. Kerberos V is appropriate because it covers both timestamps and protocol mixture semantical features.

2. Motivations

2.1. A Short Introduction to Kerberos V

The first version of Kerberos protocol was developed in the mid eighties as part of project Athena at MIT [2]. Over twenty years, different versions of Kerberos protocols have evolved. Kerberos V (**Figure 1** and **Figure 2**) is the latest version released by the Internet Engineering Task Force (IETF) [4]. It is a password-based system for authentication and authorization over local area networks. It is designed with the following aims: once a client authenticates himself to a network machine, the process of obtaining authorization to access another network service should be completely transparent to him. Namely, the client only needs enter his password once during the authentication phase. In order to access some network service, the client needs to communicate with two trusted

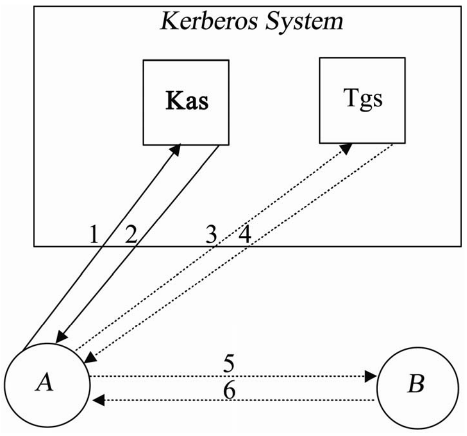

Figure 1. The layout of Kerberos V.

Authentication phase

$1. A \rightarrow Kas : \{\!|A, Tgs|\!\}$

$2. Kas \rightarrow A : \{\!| \underbrace{\{\!|A, Tgs, authK, T_a|\!\}_{K_{Tgs}}}_{authTicket}, \{\!|A, Tgs, authK, T_a|\!\}_{K_A} |\!\}$

Authorisation Phase

$3. A \rightarrow Tgs : \{\!|\{\!|A, Tgs, authK, T_a|\!\}_{K_{Tgs}}, \{\!|A, t_2|\!\}_{authK}, B|\!\}$

$4. Tgs \rightarrow A : \{\!| \underbrace{\{\!|A, B, servK, T_S|\!\}_{K_B}}_{servTicket}, \{\!|A, B, servK, T_S|\!\}_{authK} |\!\}$

Service Phase

$5. A \rightarrow B : \{\!|\{\!|A, B, servK, T_S|\!\}_{K_B}, \{\!|A, t_3|\!\}_{servK}|\!\}$

$6. B \rightarrow A : \{\!|t_3|\!\}_{servK}$

Figure 2. Kerberos V: message exchanging.

servers **Kas** and *Tgs*. **Kas** is an authentication server (or the key distribution center) and it provides keys for communication between clients and ticket granting servers. *Tgs* is a ticket granting server and it provides keys for communication between clients and application servers. The full protocol has three phases each consisting of two messages between the client and one of the servers in turn. Messages 2 and 4 are different from those in Kerberos IV [2,4] in that nested encryption has been cancelled. Later we will show that this change does not affect goals of the protocol.

2.2. Timestamps

Timestamps are heavily used in the Kerberos protocols to guarantee the recency of messages. The strand space model cannot express security protocols with timestamps, although Guttman [5] provided a notion of recency and he used it to analyze replay attacks of a variant of the Yahalom protocol, it is still impossible to analyze security protocols with timestamps. Timestamps are mainly used to avoid replay attacks in the literature of security protocols. Usually such attacks occur in protocols that involve a message encrypted by a session key, and the session key itself is sent as a part of a message which is encrypted by a long-term key. Although penetrators can never obtain a long-term key K if K is not sent as a part of a message, it is usually assumed that m will be obtained from $\{|m|\}_K$ via cryptanalysis by a penetrator after some time t, especially if a session key SK is a component of m, then it will be compromised after the time t. Here, we say that the time t is the crack time of K, and every key will be related to a crack time. Although the penetrator cannot obtain m from $\{|m|\}_K$ during a protocol session provided that $\{|m|\}_K$ did not occur in any old session and K's crack time is longer than the time of a session allowed, he still may replay stale messages and use the old compromised session keys to launch attacks if some message of the protocol does not contain necessary information to indicate its recency.

For example, in the Needham-Schroeder symmetric key protocols (see **Figure 3**), when B receives the third message $\{|A,K|\}_{K_B}$, although B can infer that it was generated by S, he is not certain of its recency because no such information is available. Perhaps $\{|A,K|\}_{K_B}$ has occurred in an old session, and a penetrator has cryptanalyzed the conversation to obtain the session K. In that case, the penetrator can start a session by resending $\{|A,K|\}_{K_B}$, and later return $\{|N_b+1|\}_K$. Denning and Sacco [6] pioneered the use of timestamps to fix the flaw of the protocol. A timestamp t, which is a number, is employed in the ticket $\{|A,K,t|\}_{K_B}$ by S to mark the time of issue, and will be compared with the current time by the receiver B to check whether the ticket is recent. In this paper, we will assume that all agents are synchronized via a global clock, so an agent knows the time when receiving or sending a message.

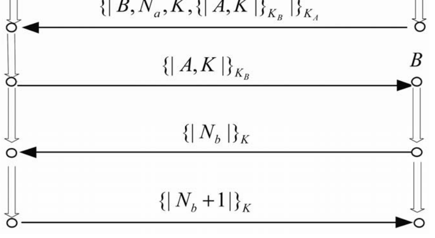
A
S
B
$\{|A,\{|B,N_a|\}_{K_A}|\}$
$\{|B,N_a,K,\{|A,K|\}_{K_B}|\}_{K_A}$
$\{|A,K|\}_{K_B}$
$\{|N_b|\}_K$
$\{|N_b+1|\}_K$

Figure 3. Needham-Schroeder symmetric key protocol.

[1]It is not the time to obtain K from $\{|m|\}_K$.

In this paper, we extend the strand space model with such features. A crack time is attached to every key. The crack-time of a key K is the time needed by a penetrator to break an encrypted message $\{|m|\}_K$.[1] We model a timestamp in the same way as atomic messages. A regular agent can attach a timestamp in a message to indicate when it sends the message, and check whether a received message encrypted by a key K is recent by comparing the timestamp in the message with the current time and the crack time of K. Once a message $\{|m|\}_K$ is no longer recent, a penetrator can break the message to obtain m.

2.3. Protocol Mixture

Another important feature of Kerberos, which is difficult to model in strand space, is protocol mixture. Kerberos protocol comprises three protocol phases: authentication, authorization, and service protocol phases. Once a client has passed an authentication phase and obtained an authentication ticket, then he can use the ticket to start multiple sessions of the authorization protocol phases in parallel to obtain different service tickets to access the services he needs provided that the authentication ticket does not expire. Similarly, once the client has gone through a session of the authorization phase, then he can use the service ticket obtained to access the service server for many times provided that the service ticket does not expire. Usually we refer to a protocol as one primary protocol, and the protocol following it as a secondary protocol. We note that other researchers have discussed the problem of protocols mixture [7,8], but they emphasized more on independency between two protocols. Namely, if they have disjoint encryption, then the first protocol is independent of the second. By this they mean that if the first protocol can achieve a security goal (either an authentication goal or a secrecy goal) when executed in isolation, then it still achieves the same security goal when executed in combination with the second protocol. In their theory, one primary and one secondary strands are rather independent of each other.

However, in Kerberos protocols, a secondary strand cannot be independent of its primary strand, and the events of a secondary strand has temporal relation with the events of the primary strand. For example, assuming that a client A runs a session s' of an authorization phase of Kerberos V, then he must have passed an authentication phase s. When A receives the second message in the session s', he must ensure that the current time should be before the ticket $\{|A,Tgs,authK,T_a|\}_{K_{Tgs}}$ expires, so A needs know the time T_a when the ticket is created, and checks how much time has elapsed until now. This side condition cannot be expressed without the semantical specification of s, because in the intended case the ticket is a term encrypted with Tgs's long-term key, which is unintelligible to A, A cannot know T_a from the ticket. Then A can only know the time T_a from the previous authentication phase s. Therefore, we need to formalize the facts that s' follows s, and A holds all the knowledge of s when he runs s', and there should be causal relation between events in s and those in s'. Such semantical features are not covered in [7,8].

In order to model the aforementioned causal relation between a primary strand and its following secondary strands, we introduce a new relation $\mapsto$ between strands. $s \mapsto s'$ holds if s is a primary protocol strand and s' is a subsequent secondary protocol strand. E.g., let s and s' be client strands in an authentication phase and authorization phase in Kerberos V respectively, $s \mapsto s'$ means that a client runs an authentication session s, and subsequently starts an authorization session s'. In practice, if $s \mapsto s'$, then s and s' may be two different processes started by the same client, and when the client starts s, he knows all the events which have occurred in s. This knowledge is useful for the client to perform actions in s'. E.g., when a client starts an authorization session, he uses an authentication ticket which is obtained in the preceding authentication session, and he knows the time when the ticket is created. So a causal relation should be imposed on two events which occur in a primary strand and its subsequent secondary strand.

Figure 4 illustrates a possible protocol execution of Kerberos V using the relation $\mapsto$. A client runs an instance in authentication phase, which is represented by the strand i_1. Following the primary protocol instance, the same client may run three authorisation subprotocol instances in parallel, which are showed in the strands i_{21}, i_{22}, and i_{23} respectively. Tr_{21} is a subtree which is a collection of client strands in the service phase. Tr_{22} and Tr_{23} are similar to Tr_{21}. Note that the semantics of the relation $\mapsto$ means that i_{21} and i_{22} and i_{23} inherits all the same knowledge from i_1, so they shares the same *authTicket*, $authK$, Tgs, T_a, *etc*. Therefore, if $term(i_1,1) = \{|authTicket, \{|A, Tgs, authK, T_a|\}_{K_A}|\}$ then then it must be the case that

$$term(i_{11},1) = \{|\{|authTicket, \{|A, t_1|\}_{authK}, B_1|\}$$

and

$$term(i_{13},1) = \{|\{|authTicket, \{|A, t_2|\}_{authK}, B_2|\}$$

for some t_1, t_2, B_1 and B_2. Here $t_1(B_1)$ can be different from $t_2(B_2)$. This means that the client use the same *authTicket* to obtain two different server tickets for accessing servers B_1 and B_2. Without the relation

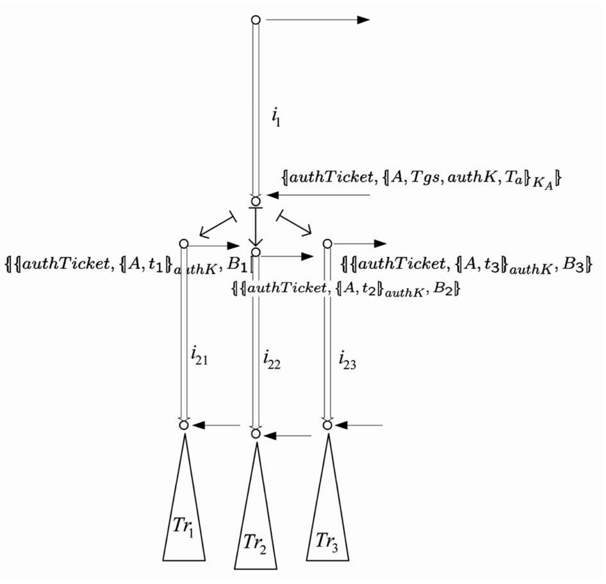
i_1
$\{|authTicket, \{|A, Tgs, authK, T_a|\}_{K_A}|\}$
$\{|\{|authTicket, \{|A, t_1|\}_{authK}, B_1|\}$
$\{|\{|authTicket, \{|A, t_3|\}_{authK}, B_3|\}$
$\{|\{|authTicket, \{|A, t_2|\}_{authK}, B_2|\}$
i_{21}
i_{22}
i_{23}
Tr_1
Tr_2
Tr_3

Figure 4. An illustration of protocol mixture.

$\mapsto$, i_{21} and i_1 are independent, therefore the knowledge inherence relation between them can not be imposed.

We extend the relation $\Rightarrow$ in the strand space model in the way that $n_1 \Rightarrow n_2$ holds if $n_1 = (s,i)$ and $n_2 = (s,i+1)$, or $n_1 = (s, length(tr(s)) - 1)$ and $n_2 = (s',0)$ and $s \mapsto s'$. Namely, the edge means either that n_1 is an immediate causal predecessor of n_2 on the same strand s or that n_1 is the last event in a primary strand s and n_2 is the first event in the subsequent secondary strand s'.

Structure of the Paper. In Section 3, we present the theory of the strand space method with our two extensions. We devote Section 5 to a new definition of unsolicited authentication test. We discuss related work and conclude the paper in Section 6.

3. Preliminaries

3.1. Messages and Actions

The set of messages is defined as the following BNF notation:

$$\begin{array}{rllll} h & ::= & \textbf{name}(A) & \mid & \textbf{nonce}(n) \\ & \mid & \textbf{key}(K) & \mid & \textbf{timestamp}(t) \\ & \mid & \{|h_1, h_2|\} & \mid & \textbf{enc}(h, K) \end{array}$$

where A is an element from a set of agents, n from a set of nonces, K from a set of keys, and t from a set of times. Here we assume that **Time** is the set of all natural numbers. $t_1 < t_2$ means that the time t_1 is earlier than t_2. We represent a timestamp by marking t as **timestamp**(t). Except this extension, the definitions of other kinds of messages are the same as those in the classic strand space theory. We call a key symmetric if $K^{-1} = K$. Otherwise, K is a public key and K^{-1} is private. For each K, we define $cracktime(K)$ as the crack time of K. $\{|h_1, h_2|\}$ is called a composed message. We will write $\{|\{|h_1, h_2|\}, h_3|\}$ as $\{|h_1, h_2, h_3|\}$. $\{|h_1, h_2|\} = \{|h_1', h_2'|\}$ if and only if $h_1 = h_1'$ and $h_2 = h_2'$. We abbreviate $\mathbf{enc}(h, K)$ as $\{|h_K|\}$, denoting the encryption of h using key K. In our formulation, we use K_A to define a long-term key shared between an agent (also called a client) A and a server, and clients have distinct keys. An element of the set of messages is also called a *term*. Terms of the form **name**(A), **nonce**(n), **timestamp**(t), or **key**(K) are said to be atomic.[2] The set of all messages is denoted by **Message**. A message h is a text message if $h \neq K$ for any K. The set of all atomic text messages is denoted by T. We frequently need the subterm relation on messages. A term g' is a subterm of g is written as $g' \sqsubset g$.

Definition 1 *The subterm relation* $\sqsubset$ *is defined inductively as the smallest relation such that* $g \sqsubset g$, $g \sqsubset \{|h|\}_K$ *if* $g \sqsubset h$, *and* $g \sqsubset \{|h_1, h_2|\}$ *if* $g \sqsubset h_1$ *or* $g \sqsubset h_2$.

In our extended strand space model, we need to revise the definition of actions. The main point is to record the time when an action takes place. The transmission of a term g at time t is denoted by $(t, +, g)$, and the reception of a term g at t is denoted by $(t, -, g)$. Both are the possible actions that participants and a penetrator can take. We represent the set of finite sequences of actions by (**Time**, ±, **Message**)*.

3.2. Strands and Strand Spaces

A strand space Σ is a set of strands with a trace mapping $tr : \Sigma \to (\mathbf{Time}, \pm, \mathbf{Message})^*$. A strand element is called a node. (s,i) is the i-th node on strand s ($0 \le i < length(s)$). We use $n \in s$ to denote that a node n belongs to the strand s. The set of all the nodes is denoted by $\mathcal{N}$. If $n = (s,i)$ and $tr(s)_i = (t, \sigma, g)$, then we define $time(n)$ and $term(n)$ and $sign(n)$ to be the occurring time, the term and the sign of the node n, respectively. Namely, $time(n) = t$, $term(n) = g$, and $sign(n) = \sigma$. We call a node positive if its term has sign $+$, and negative if its term has sign $-$. A strand is a protocol history from the point of view of a single participant in a protocol run, so we explicitly define an attribute function $attr : \Sigma \to A$ to indicate which agent's peer a strand is. Namely, $attr(s) = a$ means that a is the agent who performs actions of the strand s in the run.

[2]For convenience, we often write *A*, *n*, *K* and *t* instead of **name** (*A*), **nonce** (*n*), **key** (*K*), and **timestamp** (*t*).

As mentioned in Section 2, we introduce a relation $\mapsto$ between strands to model protocol mixture, and $s \mapsto s'$ holds if s is a primary protocol strand, and s' is a subsequent secondary protocol strand. To make our theory sound, we also restrict the relation $\mapsto$ to be a tree-like one with the following principles. First, $\mapsto$ is irreflexive, *i.e.* $s \not\mapsto s$. Second, every strand has at most one $\mapsto$ predecessor, meaning if $s \mapsto s''$ and $s' \mapsto s''$, then $s = s'$. The two restrictions are consistent with our intuition on protocol mixture. The first principle says that one protocol session can not follow itself, this simply means that the primary protocol session and any one of its following secondary protocol sessions are different. The second principle shows that one secondary protocol session follows a unique primary protocol session.

Two kinds of causal relation (arrow), $\to$ and $\Rightarrow$, are introduced to impose a graph structure on the nodes of Σ. To be more precise, the relation $n \Rightarrow n'$ holds between nodes n and n' if $n = (s,i)$ and $n' = (s, i+1)$ and $time(n) \le time(n')$, or $n = (s, length(tr(s)) - 1)$ and $n' = (s', 0)$ and $s \mapsto s'$ and $time(n) \le time(n')$. This relation means that the event n' immediately follows n. On the other hand, the relation $n \to n'$ holds for nodes n and n' if $term(n) = term(n') = g$ for some term g, $sign(n) = +$ and $sign(n') = -$, and $time(n) \le time(n')$. This represents that n sends a message g and n' receives the message at a later time. Obviously, here we require that the two relations must respect the order of time. The relation $\preceq$ is defined to be the reflexive and transitive closure of $\to$ and $\Rightarrow$, modelling the causal order of the events in the protocol execution. We say that a term g *originates* at a node n if and only if n is positive, $g \sqsubset term(n)$, and there is no node n' such that $n' \Rightarrow^+ n$ and $g \sqsubset term(n')$; We say that g uniquely originates if and only if there exists an unique node n such that g originates from node n. Nonces and other recently generated terms such as session keys are usually uniquely originated.

3.3. Penetrator Strands

The symbol **Bad** is defined to denote the set of all the penetrators, and if an agent is not in **Bad**, then it is regular. There is a set of keys that are known initially to all the penetrators, denoted as $\mathbf{K}_{\mathcal{P}}$. $\mathbf{K}_{\mathcal{P}}$ usually contains all the public keys, all the private keys of all the penetrators, and all the symmetric keys initially shared between all the penetrators and principals playing by the protocol rules. It can also contain some keys to model known-key attacks. In this paper, we only need the fact that if an agent is not a penetrator then his shared key cannot be penetrated, which is formalized as follows.

Axiom 1 *If* $A \notin \mathbf{Bad}$, *then* $K_A \notin \mathbf{K}_{\mathcal{P}}$.

In the classic strand space theory, a penetrator can intercept messages, generate messages that are computable from its initial knowledge and the messages it intercepts. These actions are modelled by a set of penetrator strands, and they represent atomic deductions. More complex deduction actions can be formed by connecting several penetrator strands. In our extension, we assume that penetrators share their initial knowledge and can cooperate each other by composing their strands. Besides the behaviors inherited from classic strand space theory, a penetrator has the ability to crack an encrypted message once the message is no longer recent (see $KC_{K,h}$ strand).

Definition 2 *A penetrator's trace relative to* $\mathbf{K}_{\mathcal{P}}$ *is one of the following, where* $t, t_1, t_2, t_3 \in \mathbf{Time}$ *and* $t_1 \le t_2 \le t_3$:

- M_g (text message): $[(t,+,g)]$, where $g \in T$.
- K_K (key): $[(t,+,K)]$, where $K \in \mathbf{K}_{\mathcal{P}}$.
- C_{gh} (concatenation): $[(t_1,-,g),(t_2,-,h),(t_3,+,\{|g,h|\})]$.
- $S_{g,h}$ (separation): $[(t_1,-,\{|g,h|\}),(t_2,+,g),(t_3,+,h)]$.
- $E_{h,K}$ (encryption): $[(t_1,-,K),(t_2,-,h),(t_3,+,\{|h|\}_K)]$.
- $D_{h,K}$ (decryption): $[(t_1,-,K^{-1}),(t_2,-,\{|h|\}_K),(t_3,+,h)]$.
- $KC_{K,h}$ (key-crack): $[(t_1,-,\{|h|\}_K),(t_2,+,h)]$, where $t_1 + cracktime(K) < t_2$.

In our theory, if a strand s belongs to a penetrator, namely, $attr(s) \in \mathbf{Bad}$, then s must be a penetrator strand. If a strand is not a penetrator strand, then it is regular. A node is called *regular* if it is not in the penetrator strands. Except the key crack strand ($KC_{K,h}$), our penetrator model is similar to the one in [1]. Here M_g (or K_K) does not imply that a penetrator can issue any unguessable terms which are not in his initial knowledge such as nonces and session keys. Because when we introduce secrecy or authentication properties about an unguessable term t for all penetrators, we usually assume that t uniquely originates from a regular strand, and this implicitly eliminates the possibility that any penetrator can originate t. Intuitively, we use $\mapsto$ to model regular agents to start a primary protocol session and then starts multiple parallel secondary protocol sessions, so a penetrator strand cannot be mixed with any other strand. To be more precise, for all penetrator strands s and all strands s', we have that $s \not\mapsto s'$ and $s' \not\mapsto s$. This implies that a penetrator strand can only be composed with other strands by the relation $\rightarrow$.

3.4. Bundles

The formal analysis based on strand spaces is carried on the notion of bundles, which represents the protocol execution under some configuration. A bundle is a causally well-founded graph, which sufficiently formalizes a session of a protocol.

Definition 3 *Suppose* $\mathcal{B}\ (N_{\mathcal{B}},(\rightarrow_{\mathcal{B}} \cup \Rightarrow_{\mathcal{B}}))$, $\rightarrow_{\mathcal{B}} \subseteq \rightarrow$, *and* $\Rightarrow_{\mathcal{B}} \subseteq \Rightarrow$. $\mathcal{B}$ *is a bundle if*

- $N_{\mathcal{B}}$ and $\rightarrow_{\mathcal{B}}$ and $\Rightarrow_{\mathcal{B}}$ are finite;
- If the sign of a node n is $-$, and $n \in N_{\mathcal{B}}$, then there is a unique positive node n' such that $n' \in N_{\mathcal{B}}$ and $n' \rightarrow_{\mathcal{B}} n$;
- If $n' \Rightarrow n$ and $n \in N_{\mathcal{B}}$, then $n' \in N_{\mathcal{B}}$ and $n' \Rightarrow_{\mathcal{B}} n$;
- $\mathcal{B}$ is acyclic.

Suppose $\mathcal{B}$ is a bundle, we say $n \in \mathcal{B}$ if n is a node in $N_{\mathcal{B}}$, and use $\preceq_{\mathcal{B}}$ to denote the reflexive and transitive closure of the relation $\rightarrow$ and $\Rightarrow$ in $\mathcal{B}$. In a bundle, it must be ensured that a node is included only if all nodes that proceed it are already included. So a bundle $\mathcal{B}$ has the following properties:

Lemma 1 (Bundle well foundedness) *Let* $\mathcal{B}$ *be a bundle. Then* $\preceq_{\mathcal{B}}$ *is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in* $\mathcal{B}$ *has* $\preceq_{\mathcal{B}}$ *minimal members.*

We have formalized the above extended strand space theory in the theorem prover Isabelle/HOL [9]. See [10] for details.

4. Penetrator's Knowledge Closure Property

In this section, we will describe a useful property on penetrator strands. This property specifies what knowledge can be obtained from some special message set. First we need to define a key is regular w.r.t. a node m in a bundle.

Definition 4 *A key* K *is regular w.r.t. a node* m *in a bundle* $\mathcal{B}$, *denoted by* $regular(k,m,\mathcal{B})$, *if and only if the following condition holds: for any node* n *in* $\mathcal{B}$, *if* $term(n) = K$ *and* $time(n) \le time(m)$, *then* n *must be regular.*

This definition is about K's secrecy w.r.t. a node m in a bundle $\mathcal{B}$, which means that K cannot be penetrated before m in the bundle. In most of the cases, we only consider security properties for a protocol in a given bundle, so it is natural for us to just consider whether a key can potentially be penetrated in this bundle. Besides, we also need consider temporal restriction $time(n) \le time(m)$ because we discuss K's secrecy a timed framework.

Definition 5 *Let* m *be a node in a bundle* $\mathcal{B}$. *A message* t, *is a component w.r.t.* m *in bundle* $\mathcal{B}$, *denoted by* $component\ (t,m,\mathcal{B})$, *if*

1) $(\forall g \quad h.t \neq \{|g,h|\})$;

2) $\left(\forall kh.t = \{|h|\}_k \rightarrow \left(regular\left(k^{-1},m,\mathcal{B}\right)\right)\right)$

Intuitively, $component(t,m,\mathcal{B})$ means that t basic unit that can not be analyzed in $\mathcal{B}$ by penetrators. Namely, t can not be detached because t is not a concatenated form; and if t is an encrypted form of $\{|h|\}_K$ t can not be decrypted before m in $\mathcal{B}$ because k^{-1} can not be penetrated before m.

Definition 6 *Let* m *be a node in a bundle* $\mathcal{B}$. a *is a message which uniquely originates at some node* n. *A message set* M *is a test suite for* a *w.r.t.* m *in* $\mathcal{B}$, *denoted by* $suite(M,a,m,n,\mathcal{B})$ *if*

1) $\forall t \in M.a\sqsubset t \rightarrow$ $component\,(t,m,\mathcal{B})$

2) $\forall t \in M.a\sqsubset t \rightarrow (\ \forall k\ h.t = \{|h|\}_k \ \rightarrow \ time(m) \ \leq \ time(n) + cracktime(k))$

3) $\forall t.a\not\sqsubset t \rightarrow t \in M$;

Intuitively, $suite(M,a,m,n,\mathcal{B})$ means that for any $t \in M$ such that $a\sqsubset t$, t can not be detached or decrypted before m because such t is a component w.r.t. m in bundle $\mathcal{B}$; furthermore, if t contains a and is of the form $\{|h|\}_K$ for some k and h, t can not be cracked before m because the duration between m and n is less than k's crack time, and this is guaranteed by (2). Recall that $time(n)$ is the first time when a occurs because a uniquely originates at n.

Now we need introduce a function $synth$ on a message set H, which captures the "building up" aspect of penetrator's ability [4,11]. $synth(H)$ is defined to be the least set that includes H, agents, timestamps and is closed under pairing, and encryption.

Definition 7 *Consider a message set* H, $synth(H)$ *is a message set which is defined inductively as follows*:

1) $A \in synth(H)$ *if* A *is an agent name*;

2) $t \in synth(H)$ *if* t *is a timestamp*;

3) $m \in synth(H)$ *if* $m \in H$;

4) $\{|h|\}_k \in synth(H)$, *if* $h \in synth(H)$ *and* $k \in H$;

5) $\{|g,h|\} \in synth(H)$, *if* $g \in synth(H)$ *and* $h \in synth(H)$.

In the context of this paper, we usually assume that a is an unguessable atomic message such as a nonce, which is uniquely originated from a regular strand and encrypted in a message. Let $M_0 = \{t \mid a\sqsubset t \wedge t \in M\}$, in later discussions we usually assume that M_0 is the set of messages which is emitted by some regular strands. f M is a test suite for a w.r.t. m in b, then the set $synth\,(M)$ is a knowledge closure which penetrators can synthesize in the bundle b from M. Namely, if the messages received in a penetror strand are in $synth\,(M)$, then the messages sent in the strand must still be in $synth\,(M)$.

Before we prove the closure property, we need two useful lemmas, as shown below:

Lemma 2 *If* M *is a test suite for* a *w.r.t.* m *in* $\mathcal{B}$, *and* $\{|g,h|\} \in synth\,(M)$, *then* $g \in synth\,(M)$ *and* $h \in synth\,(M)$.

Lemma 3 If $\{|h|\}_K \in synth(M)$, then $h \in synth(M)$ or $\{|h|\}_K \in M$.

Let a be an atomic message that uniquely originates at some node n, m be a positive penetrator node in a bundle $\mathcal{B}$ such that and $a\sqsubset term(m)$. Suppose M is a test suite for a w.r.t. m in the bundle $\mathcal{B}$, if any message that the penetrator can receive in the strand is in $synth(M)$, then the penetrator can only send a term which is still in $synth(M)$. **Figure 5** illustrates such behaviors of penetrators on knowledge, where (a) shows the cases for $C_{g,h}$, $E_{h,K}$, and $D_{h,K}$; (b) shows the case for $S_{g,h}$; and (c) shows the case for $KC_{K,h}$.

Lemma 4 *Let* m *be a positive penetrator node in a*

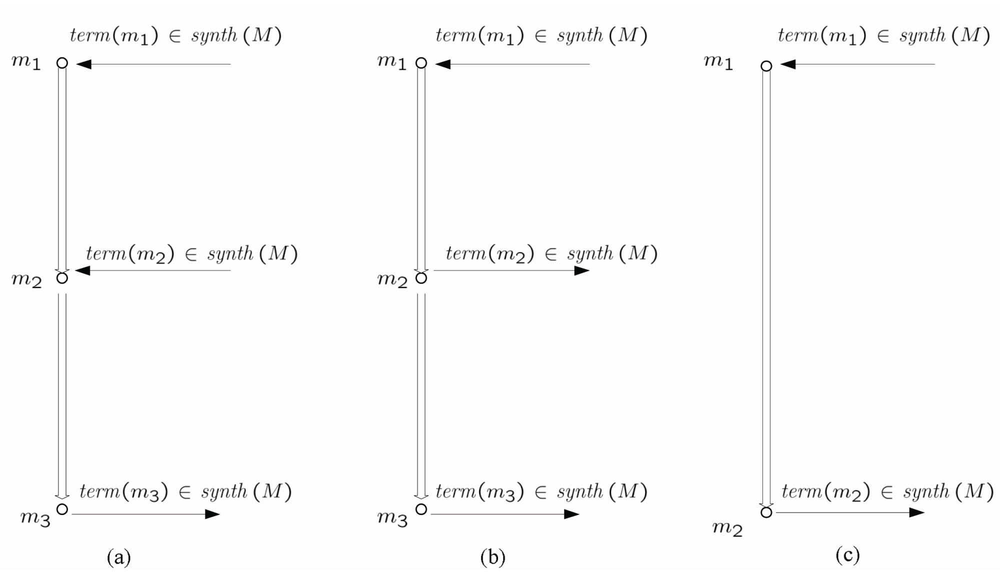
$term(m_1) \in synth\,(M)$
m_1
$term(m_2) \in synth\,(M)$
m_2
$term(m_3) \in synth\,(M)$
m_3
(a)
$term(m_1) \in synth\,(M)$
m_1
$term(m_2) \in synth\,(M)$
m_2
$term(m_3) \in synth\,(M)$
m_3
(b)
$term(m_1) \in synth\,(M)$
m_1
$term(m_2) \in synth\,(M)$
m_2
(c)

Figure 5. Penetrator's knowledge closure property.

bundle $\mathcal{B}$, a *be an atomic message that uniquely originates at a regular node* n, M *be a message set such that* $suite(M,a,m,n,\mathcal{B})$, *and* $term(m') \in synth(M)$ *for any node such that* $m' \Rightarrow^{+} m$, *then* $term(m) \in synth(M)$.

Proof. For convenience, the assumption that $term(m) \in synth(M)$ for any node such that $m \Rightarrow^{+} n$ is referred as (1) in the proof as follows.

By case analysis on the form of penetrator strand, we can easily exclude the cases when m is in a strand M_g, K_K. If thus, we can conclude that a originates at m. This contradicts with the fact that uniquely originates at a regular node n. Therefore, m is in a strand i such that i is $C_{g,h}$, $S_{g,h}$, $E_{h,K}$, $D_{h,K}$, or $KC_{K,h}$.

Case 1: i is in $C_{g,h}$, then $index(m) = 2$, $term(i,0) = g$, $term(i,1) = h$, and $term(m) = \{|g,h|\}$ for some g, h, and $sign(i,0) = -$, and $sign(i,1) = -$. From the assumption (1), we have $term(i,0) \in synth(M)$ and $term(i,1) \in synth(M)$, then $g \in synth(M)$ and $h \in synth(M)$; By the definition of $synth$ operator, $\{|g,h|\} \in synth(M)$, then $term(m) \in synth(M)$.

Case 2: i is in $S_{g,h}$, then $index(m) = 1$, or $index(m) = 2$, $term(i,0) = \{|g,h|\}$, $term(i,1) = g$, and $term(m) = h$ for some g, h. From the assumption (1), we have $term(i,0) \in synth(M)$, $\{|g,h|\} \in synth(M)$, by Lemma 4, we have $g \in synth(M)$ and $h \in synth(M)$. So $term(m) \in synth(M)$.

Case 3: i is in $E_{h,K}$, then $index(m) = 2$, $term(i,0) = K$, $term(i,1) = h$, and $term(m') = \{|h|\}_K$ for some K, h, and $sign(i,0) = -$, and $sign(i,1) = -$. From the assumption (1), $term(i,0) \in synth(M)$ and $term(i,1) \in synth(M)$, then $K \in synth(M)$ and $h \in synth(M)$; by the definition of $synth$, we have $\{|h|\}_K \in synth(M)$, then $term(m) \in synth(M)$.

Case 4: i is in $D_{h,K}$, then $index(m) = 2$, $term(i,0) = K^{-1}$, $term(i,1) = \{|h|\}_K$, and $term(m) = h$ for some K, h, and $sign(i,0) = -$, and $sign(i,1) = -$. From the assumption (1), we have $term(i,0) \in synth(M)$ and $term(i,1) \in synth(M)$, therefore $K^{-1} \in synth(M)$ and $\{|h|\}_K \in synth(M)$, by Lemma 4, we have either (4-1) $term(m) = h \in synth(M)$ or (4-2) $\{|h|\}_K \in M$. From (4-1), the lemma can be proved at once. For the case (4-2), there are also two subcases, either (4-2-1) $a \not\sqsubset \{|h|\}_K$ or (4-2-2) $a \sqsubset \{|h|\}_{KK}$. From (4-2-1), we have $a \not\sqsubset h$, by M is a test suite for a in b, so $h \in M$, then $h \in$ **synth** M, then **term** $m' \in$ **synth** M. From (4-2-2), then by M is a test suite for a in b, we have **component** $\{|h|\}_K$ b, then we have $regular(K^{-1}, m, \mathcal{B})$. From this and $(i,0) \in \mathcal{B}$ and $term(i,0) = K^{-1}$, then i is regular, but this contradicts with that m is in a penetrator strand.

Case 5: i is in $KC_{K,h}$, then $index(m) = 1$, $term(i,1) = h$, $term(i,0) = \{|h|\}_K$, (2) $term(i,0) + cracktime(K) < term(i,1)$. From the assumption (1), we have $\{|h|\}_K \in synth(M)$. From this, by Lemma 3, we have either (5-1) $h \in synth(M)$ or (5-2) $\{|h|\}_K \in M$. From (5-1), the lemma can be proved at once. For the case (5-2), there are also two subcases, either (5-2-1) $a \not\sqsubset \{|h|\}_K$ or (5-2-2) $a \sqsubset \{|h|\}_K$. From (5-2-1), we have $a \not\sqsubset h$, by the definition of $suite(M,a,m,n,\mathcal{B})$, so $h \in M$, then $h \in synth(M)$. From (5-2-2), then by the definition of $suite(M,a,m,n,\mathcal{B})$, we have (3) $time(m) \le time(n) + cracktime(k)$. From $a \sqsubset term(i,0)$, and a uniquely originates at n, we have $time(n) \le time(i,0)$. Then we have

$$time(n) + cracktime(k) \le time(i,0) + cracktime(k),$$

with (3), we have $time(m) \le time(i,0) + cracktime(k)$. But this contradicts with (2).

On the other side, a strand's receiving nodes get messages which are all in $synth(M)$, but a new message, which is not in $synth(M)$, is sent in the strand, then the strand must be regular because a penetrator strand can not create such a term. The result can be simply inferred from Lemma 4.

Lemma 5 *Let mbe a positive node in a bundle* $\mathcal{B}$, a *be an atomic message that uniquely originates at a regular node* n, M *be a message set such that* $suite(M,a,m,n,\mathcal{B})$, *and* $term(m') \in synth(M)$ *for any node such that* $m' \Rightarrow^{+} m$, *and* $term(m) \notin synth(M)$, *then* m *is regular*·

For Lemma 4 and 5, we have two comments:

1) Lemma 4 characterizes the knowledge closure properties of a penetrator's operations on messages. It says that if a penetrator only receives messages in $synth(M)$, where M is a test suite for some atomic message a, then the augmented knowledge of the penetrator is still in $synth(M)$ after the receiving actions.

2) Lemma 5 provides a key technique to prove the authentication guarantee that m is regular. Intuitively, condition (1) of $suite$ requires the secrecy of the inverse key k^{-1} for any key k which is used to encrypt any message in M containing a; condition (2) of operator $suite$ is a recency restriction that these encrypted messages containing a can not be cracked until m. Therefore this lemma provides a means of using secrecy and recency restriction to prove authentication guarantee. We will see this result is very useful for us to check whether a strand is regular in the next sections.

Note that the two lemmas relates the algebraic operator $synth$ in trace theory [4,11] with penetrator's strand ability to deduce knowledge, which is the most important one which differs our work from the classical strand space theory. Such closure properties are not available in the classical strand space theory because message algebra operators such as $synth$ are not formalized.

5. Unsolicited Tests

In [12] (Subsection 4.2.3), a negative node n is an unsolicited test for $\{|h|\}_K$, if $\{|h|\}_K$ is a *test component* for any atomic text a in n, and K cannot be penetrated in the strand space. Then an unsolicited test for $\{|h|\}_K$ in a bundle $\mathcal{B}$ can guarantee the existence of a positive regular node of which $\{|h|\}_K$ is a component. We simplify this definition of unsolicited tests by the following two aspects:

1) we consider a node n is an unsolicited test for $\{|h|\}_K$ in a bundle $\mathcal{B}$;

2) we only require that $\{|h|\}_K$ is a subterm of the term of n, and K is regular w.r.t. n in the bundle $\mathcal{B}$ instead of a strand space.

In our formulation, unsolicited authentication test is a kind of regularity about an encrypted term $\{|h|\}_K$, which is a subterm of a node n where K cannot be penetrated before n in a bundle $\mathcal{B}$. Then it can be ensured that there is a positive regular node m originating $\{|h|\}_K$ as a subterm, *i.e.*, m has $\{|h|\}_K$ as a subterm and it also holds that $\{|h|\}_K \not\sqsubset term(m')$ for any node $m' \preceq_{\mathcal{B}} m$. Intuitively, the reason why m must be regular lies in that K cannot be penetrated before m in $\mathcal{B}$. So the penetrator cannot create $\{|h|\}_K$ by encrypting h with K.

Definition 8 *Given a bundle* $\mathcal{B}$. *A node* n *in* $\mathcal{B}$ *is an unsolicited test for* $\{|h|\}_K$ *if* $\{|h|\}_K \sqsubset term(n)$, *and* K *is regular w.r.t.* n *in* $\mathcal{B}$.

Lemma 6 (Unsolicited authentication test) $\mathcal{B}$ *is a given bundle. Let* n *be an unsolicited test for* $\{|h|\}_K$. *Then there exists a positive regular node* m *in* $\mathcal{B}$ *such that* $m \preceq_{\mathcal{B}} n$ *and* $\{|h|\}_K \sqsubset term(m)$ *and* $\{|h|\}_K \not\sqsubset term(m')$ *for any node* m' *such that* $m' \preceq_{\mathcal{B}} m$.

Proof. Let $P =_{df} \{x \mid x \preceq_{\mathcal{B}} n \wedge \{|h|\}_K \sqsubset term(x)\}$. Obviously, $m \in P$. By Lemma 1, there exists a node m' such that m' is minimal in P, which means that $\{|h|\}_K \sqsubset term(m')$, $m' \preceq_{\mathcal{B}} n$, and for all y such that $y \preceq_{\mathcal{B}} m'$, $y \notin P$. Hence, $\{|h|\}_K \not\sqsubset term(y)$.

First, we prove that the sign of m' is positive by contradiction. If $sign(m') = -$, then by the upward-closed property of a bundle there must be another node m'' in $\mathcal{B}$ such that $sign(m'') = +$ and $m'' \to m'$. Then we have (a) $m'' \preceq_{\mathcal{B}} m'$ and (b) $term(m') = term(m'')$. By (a) and $m' \preceq_{\mathcal{B}} n$, we have $m'' \preceq_{\mathcal{B}} n$. By (b) and $\{|h|\}_K \sqsubset term(m')$, we have $\{|h|\}_K \sqsubset term(m'')$. Hence, $m'' \in P$ which contradicts with the minimality of m'.

Second, we prove that m' is regular. We show that a contradiction can be derived if m' is in a penetrator strand. Here, we only analyze cases when m' is in either $C_{g,g'}$ (concatenation strand), $E_{g,K'}$ (encryption strand), or $KC_{K',g}$ (key crack strand). Other cases are either straightforward or can be analyzed in a similar way.

- m' is in $i \in C_{g,g'}$.

By the form of the strand $C_{g,g'}$ and the fact that m' is a positive node, we have $m' = (i,2)$, $term(m') = \{|g, g'|\}$, $term(i,0) = g$, and $term(i,1) = g'$ for some g, g'. By the upwards-closed property of a bundle, we have that nodes $(i,0)$ and $(i,1)$ must be in $\mathcal{B}$. By $\{|h|\}_K \sqsubset \{|g, g'|\}$, we have either $\{|h|\}_K \sqsubset g$ or $\{|h|\}_K \sqsubset g'$, *i.e.* $\{|h|\}_K \sqsubset term(i,0)$ or $\{|h|\}_K \sqsubset term(i,1)$. So either node $(i,0) \in P$, or node $(i,1) \in P$. Both cases contradict with the minimality of m'.

- m' is in $i \in E_{g,K'}$.

By the form of the strand $E_{g,K'}$ and the fact that m' is a positive node, we have $m' = (i,2)$, $term(m') = \{|g|\}_{K'}$, $term(i,0) = K'$, and $term(i,1) = g$ for some g and K'. So $\{|h|\}_K \sqsubset \{|g|\}_{K'}$. Then it is straightforward that either (1) $\{|h|\}_K \sqsubset g$ or (2) $h = g$ and $K = K'$. For the first case, we have $\{|h|\}_K \sqsubset term(i,1)$. It is easy to derive a contradiction by the same argument as before. For the second case, by the definition of the relation $\Rightarrow$, we have (a) $time(i,0) \le time(i,2)$. And by definition of P, we also have (b) $time(m') \le time(n)$. Hence, $time(i,0) \le time(n)$. However, by the assumption that K must be regular w.r.t. n in $\mathcal{B}$, $term(i,0)$ must be regular, and this contradicts with the fact that i is a penetrator strand.

- m' is in $i \in KC_{K',g}$.

By the form of the strand $KC_{K',g}$, and the fact that m' is a positive node, we have $m' = (i,1)$, $term(m') = g$, $term(i,0) = \{|g|\}_{K'}$ for some g and K', and

$$time\,(i,0) + cracktime(K) < time(m').$$

By $\{|h|\}_K \sqsubset term(m') = g$, so $\{|h|\}_K \sqsubset term(i,0) = \{|g|\}_{K'}$. Obviously $(i,0) \preceq_{\mathcal{B}} m' \preceq_{\mathcal{B}} n$. So $(i,0) \in P$, which contradicts with the minimality of m'.

The proof totally depends on the well-founded induction principle on bundles, and we have formalized the proof of this lemma in Isabelle/HOL in our inductive strand space model, and the proof scripts are available at [10]. In fact, lemma 6 provides a useful proof method to reason about authentication properties basing on secrecy properties. Note that the premise that n is an unsolicited test for $\{|h|\}_K$ requires that K is regular w.r.t. n in $\mathcal{B}$, which is an assumption on the secrecy of K. And the conclusion is an authentication guarantee of the existence of a regular node m. Besides, compared with the original version of unsolicited test, our result also has two extensions that $m \preceq_{\mathcal{B}} n$ and m is minimal (*i.e.*, $\{|h|\}_K \not\sqsubset term(m')$ for any node m' such that $m' \preceq_{\mathcal{B}} m$). We find that the extended version of unsolicited authentication test is quite useful in many cases, especially in the verification of authentication properties of symmetric key based protocols. In [13], we have used a version of unsolicited authentication test in the classical strand space theory to give new proofs of authentication properties of the Otway-Rees protocol. In this work, we have successfully applied unsolicited authentication test to our study of the Kerberos V protocol in the next paper.

6. Conclusions and related Work

This work is an extension of [14]. We have added two new semantical features in our new framework: timestamp and protocol mixture. In essence, our treatment of timestamps is to add a global clock to the underlying execution model, and to extend every action by a temporal annotation. This allows us to align the timestamps sent in the protocol messages with the actual occurrence times of the corresponding actions. Although it is quite straightforward, it gives a powerful mechanism to reason about recency of a message. For protocol mixture, we admit a realistic assumption that a regular agent can start multiple parallel secondary sessions once he has finished a primary protocol session, and he holds all the information of the primary protocol session when he begins a secondary protocol session. So we introduce a causal relation $\mapsto$ between strands to model the protocol dependency. The above two semantical features are seldom discussed in previous works of strand space literature.

Despite the aforementioned extensions in semantics, the definition of a bundle, which is the cornerstone of the strand space theory, remains unchanged. So the induction principle on the well-foundedness of a bundle is still effective in our model. Based on this principle, we have proved an extended result of the unsolicited authentication test.

In the literature, most of the existing approaches for protocol analysis have not concentrated on timestamps and replay attacks. These include the CSP model-checking approach [15], the rank functions [16], and the Multi-Set Rewriting formalism (MSR) [17]. Paulson and Bella's inductive method [4,11] is one exception. They not only have extended their method to model replay attacks, but also have succeeded in applying their method to the Yahalom protocol and the Kerberos IV protocol. Recently, Bozga *et al.* [18] proposed an approach based on timed automata, symbolic verification techniques and temporal logic to analyze security protocols with timestamps. But they haven't applied their approach to any real-world security protocols.

For protocol mixture, there have been a few works to reason rigorously about protocol interactions. For instance, Meadows studied the Internet Key Exchange protocol, emphasizing the potential interactions among its specific sub-protocols [19]. The analysis work was conducted in the NRL protocol analyzer. Recently, Cremers discussed the feasibility of multi-protocol attacks, and his work is done in the operational semantical framework which considers a so-called type flaw attacks [20]. All these works, including [7], focus on protocol interactions by message exchanging. Instead, our work emphasizes on the dependency between a primary protocol session and a secondary protocol session. Here we assume that when a regular agent starts a secondary protocol session, he should be aware that he has finished a corresponding primary protocol session, and he maintains all the information obtained in the primary protocol session, such as tickets and the creation time of the tickets. These modelling assumptions fit well with the real-world environments where the Kerberos protocols run.

7. References

[1] F. Javier Thayer, J. C. Herzog and J. D. Guttman, "Strand Spaces: Proving Security Protocols Correct," *Journal of Computer Security*, Vol. 7, No. 1, 1999, pp. 191-230.

[2] S. P. Miller, J. I. Neuman, J. I. Schiller and J. H. Saltzer, "Kerberos Authentication and Authorisation System," Technical Report, Technical Plan Section E.2.1, MIT, Athena, 1989.

[3] K. R. C. Neuman and S. Hartman, "The Kerberos Network Authentication Service (v5)," Technical report, Internet RFC 4120, July 2005.

[4] G. Bella, "Inductive Verification of Cryptographic Protocols," PhD thesis, Cambridge University Computer Laboratory, 2000.

[5] J. D. Guttman, "Key Compromise, Strand Spaces, and the Authentication Tests," *Proceedings of* 7*th Conference on the Mathematical Foundations of Programming Semantics*, ENTCS 45, 2001, pp. 1-21.

[6] D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols," *Communications of the ACM*, Vol. 24, No. 8, 1981, pp. 533-536.

[7] F. Javier Thayer, J. C. Herzog and J. D. Guttman, "Mixed Strand Spaces," *Proceedings of* 12*th IEEE Computer Security Foundations Workshop*, 1999, pp. 72-82.

[8] J. D. Guttman and F. Javier Thayer, "Protocol Independence through Disjoint Encryption," *Proceedings of* 13*th IEEE Computer Security Foundations Workshop*, 2000, pp. 24-34.

[9] T. Nipkow, L. C. Paulson and M. Wenzel, "Isabelle/HOL—A Proof Assistant for Higher-Order Logic," LNCS 2283. Spinger, 2002.

[10] Y. Li, "Strand Space and Security Protocols". http://lcs.ios.ac.cn/˜lyj238/strand.html

[11] L. C. Paulson, "The Inductive Approach to Verifying Cryptographic Protocols," *Journal of Computer Security*, Vol. 6, No. 1-2, 1998, pp. 85-128.

[12] J. D. Guttman and F. Javier Thayer, "Authentication

Tests and the Structure of Bundles," *Theoretical Computer Science*, Vol. 283, No. 2, 2002, pp. 333-380.

[13] Y. Li and J. Pang, "Generalized Unsolicited Tests for Authentication Protocol Analysis," *Proceedings of* 7*th Conference on Parallel and Distributed Computing*, 2006, pp. 509-514.

[14] Y. Li, "The Inductive Approach to Strand Space," *Proceedings of* 25*th IFIP Conference on Formal Techniques for Networked and Distributed Systems*, LNCS 3731, 2005, pp. 547-552.

[15] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR," *Proceedings of* 2*nd International Conference on Tools and Algorithms for the* 10 *Construction and Analysis of Systems*, LNCS 1055, pages 147-166, 1996.

[16] J. Heather and S. A. Schneider, "Toward Automatic Verification of Authentication Protocols on an Unbounded Network," *Proceedings of* 13*th IEEE Computer Security Foundations Workshop*, 2000, pp. 132-143.

[17] F. Butler, I. Cervesato, A. Jaggard and A. Scedrov, "A Formal Analysis of Some Properties of Kerberos 5 Using MSR," *Proceedings of* 15*th IEEE Computer Security Foundations Workshop*, 2002, 175-190.

[18] L. Bozga, C. Ene and Y. Lakhnech, "A Symbolic Decision Procedure for Cryptographic Protocols with Time Stamps," *Journal of Logic and Algebraic Programming*, Vol. 65, No. 1, 2005, pp. 1-35.

[19] C. Meadows, "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer," *Proceedings of* 12*th IEEE Computer Security Foundations Workshop*, 1999, pp. 216-231.

[20] C. J. F. Cremers, "Feasibility of Multi-Protocol Attacks," *Proceedings of* 1*st Conference on Availability, Reliability and Security*, 2006, pp. 287-294.