Extending the Strand Space Method with Timestamps: 
Part II Application to Kerberos V*

Yongjian Li¹,², Jun Pang³

¹Chinese Academy of Sciences, Institute of Software Laboratory of Computer Science, Beijing, China
²The State Key Laboratory of Information Security, Beijing, China
³University 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 show how to use the novel extended strand space method to verify Kerberos V. First, we formally model novel semantical features in Kerberos V such as timestamps and protocol mixture in this new framework. Second, we apply unsolicited authentication test to prove its secrecy and authentication goals of Kerberos V. Our formalization and proof in this case study have been mechanized using Isabelle/HOL.

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

1. Introduction

The first version of Kerberos protocol was developed in the mid eighties as part of project Athena at MIT [1]. 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) [2]. 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.

As we introduced in the previous paper [3], there are two novel semantic features in Kerberos V protocol. First, it uses timestamps to prevent replay attacks, so this deficiency of the strand space theory makes it difficult to analyze these protocols. Second, it is divided into three causally related multiple phases: authentication, authorization, and service protocol phases. 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, Kerberos uses 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 both the timestamps and protocol mixture in a semantic framework.

The aim of this paper is practical. We hope to apply the extended theory in [3] to the analysis of Kerberos V protocol. Kerberos V is appropriate as our case study because it covers both timestamps and protocol mixture semantical features.

Structure of the Paper: Section 2 briefly introduces the overview of Kerberos V. Section 3 presents the formalization of Kerberos V. Sections 4 and 5 prove its secrecy and authentication goals. We discuss related work and conclude the paper in Section 6.

2. An Overview of Kerberos V

The protocol’s layout and its message exchanging are presented in Figure 1 and Figure 2 separately. In the infrastructure of the Kerberos V protocol, there is a unique authentication server, and some (not necessarily only one) ticket granting servers. The latter assumption is different from that in [4], where only a unique ticket granting server exists.

*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.
In order to access some network service, the client needs to communicate with two trusted 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 clients and application servers. The full protocol needs to communicate with two trusted servers Kas and Tgs.

Detailed explanation about Kerberos V is delayed to Section 2, where the protocol is formalized in strand space model with our extensions. Here we only give an overview of the general principles to guarantee recency, secrecy and authentication in the design of Kerberos V.

For recency,

- A regular sender should attach a timestamp to indicate the time when the message is issued; usually such a message is of the form $\{[t]\}_{K}$, where $t$ is the time, $K$ may be either a session key or long-term key.

- When a regular receiver checks the recency of the message by comparing the timestamp $t$ with the reception time. More formally, if the receiving node is $n$, then $\text{time}(n)$ should be no later than $\text{cracktime}(K) + t$, meaning that this message cannot be cracked at $\text{time}(n)$, which in turn indicates that the message $\{[t]\}_{K}$ is recent.

For an encrypted message $\{[h]\}_{K}$, the secrecy of a part of the plain message $h$ also comes from both the secrecy of $K$ and the recency of the message $\{[h]\}_{K}$ itself. That is to say, when a regular receives $\{[h]\}_{K}$ at time $t$, it must be ensured that the aforementioned two conditions must be guaranteed until $t$. From this, we can see that recency and secrecy are closely related with each other in a timed protocol framework.

Unsolicited tests are the main mechanism to guarantee authentication. Because a guarantee of the existence of a regular node can be drawn from an unsolicited test, a regular agent uses unsolicited test to authenticate its regular protocol participant in Kerberos V.

Now let us briefly review the main theoretical results in [3], which will be used in this work. For interesting readers, refer to [3] for preliminary definitions.

If an agent is not a penetrator then his shared key cannot be penetrated, which is formalized as follows:

**Axiom 1** If $A \not\equiv \text{Bad}$, then $K_{A} \not\in K_{P}$.

Lemma 1 is the main technique used to reason about authentication guarantee of a node $n$ which is an unsolicited test for an encrypted term of the form $\{[h]\}_{K}$ (e.g., the tickets $\{[A, Tgs, authK, T_{A}]\}_{K_{Tgs}}$, $\{[A, Tgs, authK, T_{A}]\}_{K_{Tgs}}$, and so on). That is to say, regular agents can use an unsolicited test with other properties of the protocol to guarantee that the agent who originates the term $\{[h]\}_{K}$ should be an intended regular agent.

**Lemma 1 (Unsolicited authentication test)** $B$ is a given bundle. Let $n$ be an unsolicited test for $\{[h]\}_{K}$. Then there exists a positive regular node $m$ in $B$ such that $m \succeq_{B} n$ and $\{[h]\}_{K} \sqsubseteq \text{term}(m)$ and $\{[h]\}_{K} \sqsubseteq \text{term}(m)$ for any node $m$ such that $m \succeq_{B} m$.

Let $a$ be an atomic message that uniquely originates at some node $n$, $m$ be a positive penetrator node in a bundle $B$ such that and $a \sqsubseteq \text{term}(m)$. Suppose $M$ is a test suite for $a$ w.r.t. $m$ in the bundle $B$. A strand's
receiving nodes get messages which are all in \( \text{synth}(M) \), but a new message, which is not in \( \text{synth}(M) \), is sent in the strand, then the strand must be regular because a penetrator strand can not create such a term.

**Lemma 2** Let \( m \) be a positive node in a bundle \( B \), \( a \) be an atomic message that uniquely originates at a regular node \( n \), \( M \) be a message set such that \( \text{suite}(M,a,m,n,B) \), and \( \text{term}(m) \in \text{synth}(M) \) for any node such that \( m \rightarrow^* m \), and \( \text{term}(m) \notin \text{synth}(M) \), then \( m \) is regular.

We will illustrate these general principles in detail in the next sections when we formalize the semantics and prove secrecy properties of Kerberos V.

### 3. Formalizing Kerberos V

To model the penetrator to break a message encrypted by a long-term shared key or a session key, we define two constants \( \text{shrKcracktime} \) and \( \text{sessionKcrktime} \). The crack time of any regular agent’s long-term shared key is the constant \( \text{shrKcracktime} \), and the crack time of any session key originated by an agent \( A \) in Kerberos V is

**Axiom 2** \( \text{cracktime}(K_a) = \text{shrKcracktime}, \) for any regular agent \( A \) in Kerberos V.

The crack time of any session key originated by an authentication server is the constant \( \text{sessionKcrktime} \),

**Axiom 3** \( \text{cracktime}(\text{authK}) = \text{sessionKcrktime}, \) for any session key \( \text{authK} \) originated by \( \text{Kas} \).

The trace \( tr \) specifications of the regular strands of Kerberos V (see Figure 2) are defined as predicates:

1) Part I (Authentication Phase)

- **Ag-I** \([i_1, A, \text{Tgs}, \text{authK}, T_a, \text{authTicket}, t_0, t_1] \) iff

\[
tr(i_1) = \begin{cases} 
(t_0, +, \{ A, \text{Tgs} \}) \\
(t_1, -, \{ A, \text{authTicket} \}) \\
\{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}} 
\end{cases}
\]

where \( \text{Tgs} \in \text{TGSs} \) and \( t_1 - T_a \leq \text{shrKcracktime} \).

- **AS** \([a_s, A, \text{Tgs}, \text{authK}, T_a, t_0, t_1] \) iff

\[
tr(as) = \begin{cases} 
(t_0, -, \{ A, \text{Tgs} \}) \\
(t_1, +, \{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}}) \\
\{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}} 
\end{cases}
\]

where \( \text{Tgs} \in \text{TGSs} \).

In the first phase, when \( \text{Kas} \) issues the second message

\[
\{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}}, \{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}}
\]

\( \text{authK} \) is the session key that will be used for the client \( A \) to communicate with a ticket grant server \( \text{Tgs} \), \( \text{Kas} \) attaches \( T_a \) with the message to indicate when this message is sent; if \( A \) receives this message at time \( t_1 \), \( A \) will check the condition \( t_1 - T_a \leq \text{shrKcracktime} \) to ensure the recency of this message. At the end of this phase, \( A \) obtains a ticket \( \text{authTicket} \) and the session key \( \text{authK} \) to communicate with \( \text{Tgs} \).

2) Part II (Authorization Phase)

- **Ag-II** \([i_2, A, \text{authK}, \text{authTicket}, B, \text{servK}, T_s, \text{servTicket}, t_2, t_1] \) iff \( \exists i_1, \text{Tgs}, t_0, t_1, i_1 \Rightarrow i_1 \land \)

- **Ag-I** \([i_1, A, \text{Tgs}, \text{authK}, T_a, \text{authTicket}, i_0, t_1] \land \)

\[
tr(i_1) = \begin{cases} 
(t_0, +, \{ A, \text{authTicket} \}) \\
(t_1, +, \{ A, \text{authK}, B \}) \\
\{ A, B, \text{servK}, T_s \}_{\text{authK}} 
\end{cases}
\]

where \( \text{Tgs} \in \text{TGSs} \) and \( t_1 - T_a \leq \text{shrKcracktime} \) and \( t_1 - T_s \leq \text{sessionKcrktime} \).

- **TGS** \([tgs, A, \text{Tgs}, \text{authK}, B, \text{servK}, T_a, i_0, t_0, t_1] \) iff

\[
tr(tgs) = \begin{cases} 
(t_0, -, \{ A, \text{Tgs}, \text{authK}, T_a \}_{\text{authK}}) \\
(t_1, +, \{ A, B, \text{servK}, T_s \}_{\text{authK}}) \\
\{ A, B, \text{servK}, T_s \}_{\text{authK}} 
\end{cases}
\]

where \( \text{Tgs} \in \text{TGSs}, \ B \notin \text{TGSs}, t_1 + \text{sessionKcrktime} \leq T_s + \text{shrKcracktime} \).

In the second phase, the situation is more complex. Both \( \text{Tgs} \) and \( A \) need to check whether their received messages are recent by the same mechanism. Furthermore, \( \text{Tgs} \) also need ensure a side condition that

\[
t_1 + \text{sessionKcrktime} \leq T_s + \text{shrKcracktime}
\]
to guarantee that the application server \( B \) only receives a recent service ticket. Informally speaking, this condition means that \( \text{Tgs} \) can guarantee any \( \text{authK} \) that he receives can only be compromised later than \( \text{servK} \) which is associated with the \( \text{authK} \). We will comment this side condition in analysis in the third phase. At the end of this phase, \( A \) obtains a ticket \( \text{servTicket} \) and the session key \( \text{servK} \) to communicate with \( B \).
3) Part III (Service Phase)

- **Ag-III** \([i_A, A, servK, servTicket, t_4, t_2]\) iff
  \[
  \exists i_T, \text{tgs}, authK, T_s, t_2, i_1, i_2, authTicket, B, T_s, t_2, t_3.
  \]
  \[
  tr(i_T) = \left[ \left(t_4, +, \left\{ \left\{ A, T_s \right\}_{servK} \right\} \right) \right] \wedge \left[ \left(t_2, -, \left\{ \left\{ T_s \right\}_{servK} \right\} \right) \right].
  \]

- **Ag-I** \([i_A, A, Tgs, authK, T_s, authTicket, t_3, t_1]\)

- **Ag-II** \([i_A, A, authK, authTicket, B, servK, T_s, servTicket, t_2, t_3]\) \(\land\) \(i_1 \leftrightarrow i_2 \land i_2 \rightarrow i_3\)
  where \(Tgs \in TGSs\), \(t_3 - T_S \leq \text{shrink Time} \) and \(t_4 - T_S \leq \text{session Ker Ker Time}\).

- **Apps** \([\text{apps}, A, B, servK, T_s, T_4, t_3, t_1]\) iff
  \[
  tr(\text{apps}) = \left[ \left(t_0, -, \left\{ \left\{ A, B, servK, T_s \right\}_{k_B} \right\} \right) \right] \wedge \left[ \left(t_1, +, \left\{ \left\{ A, T_s \right\}_{servK} \right\} \right) \right],
  \]

where \(i_0 - T_0 \leq \text{session Ker Ker Time}\).

In the last phase, it is subtle for the application server \(B\) to check the recency of the message \(\left\{ \left\{ A, B, servK, T_s \right\}_{k_B} \right\} \wedge \left\{ \left\{ A, T_s \right\}_{servK} \right\}\) from the ticket \(\left\{ \left\{ A, B, servK, T_s \right\}_{k_B} \right\} \), \(B\) knows that \(Tgs\) must have issued \(\left\{ \left\{ A, B, servK, T_s \right\}_{k_B} \right\} \wedge \left\{ \left\{ A, B, servK, T_s \right\}_{authK} \right\}\) at time \(T_s\). The potential compromise of \(servK\) is from the message \(\left\{ \left\{ A, B, servK, T_s \right\}_{authK} \right\}\). A penetrator can either directly break \(\left\{ \left\{ A, B, servK, T_s \right\}_{authK} \right\}\) to obtain \(servK\), or have \(authK\) first then decrypt the message \(\left\{ \left\{ A, B, servK, T_s \right\}_{authK} \right\}\) to obtain \(servK\). Since \(authK\) is also a session key which is originated by \(Kas\) in an earlier time than \(T_s\), the guarantee for the confidentiality of \(authK\) is of extreme importance. The corresponding ticket \(\left\{ \left\{ A, Tgs, authK, T_s \right\}_{k_tgs} \right\}\) is not available for \(B\), \(B\) cannot know the creation time of \(authK\). So \(B\) cannot check whether \(authK\) has been compromised. Fortunately, if \(Tgs\) can guarantee that any \(authK\) which it receives will be compromised later than \(servK\), associated with the \(authK\), then it is enough for \(B\) to check \(t_0 - T_S \leq \text{session Ker Ker Time}\) to ensure that the \(authK\) has not been compromised. At the end of this stage, \(A\) and \(B\) authenticate each other, and \(A\) can access the service provided by \(B\).

The authentication server \(Kas\) must obey the following principles to generate a session key \(authK\):
- \(authK\) must never be known initially to a penetrator, i.e., \(authK \notin K_B\);
- \(authK\) must be uniquely originated;
- \(authK\) is a symmetric key;
- \(authK\) must not be the same as an agent’s long-term shared key.

We summarize these principles as the following axiom:

**Axiom 4** For any authentication server strand as such that \(AS[as, A, Tgs, authK, t_3, t_1]\), we have
\(authK \notin K_B\), \(authK\) uniquely originates in \((as, 1)\), \(authK = authK '-1\), and \(authK \notin K_B\).

A ticket grant server creates the session key \(servK\) by three principles, which are similar to those which the authentication server obeys to create the session key \(authK\).

**Axiom 5** For any ticket grant server strand \(tgs\) such that \(TGS[tgs, A, Tgs, authK, servK, B, T_s, T_3, t_0, t_1]\), \(servK \notin K_B\), \(servK\) uniquely originates in \(tgs, 1\), \(servK = servK '-1\), and \(servK \notin K_B\) for any agent \(B\).

In the following two subsections, we verify the secrecy and authentication properties of Kerberos V. We use similar ways for representing these security properties as in [5]. However, we may need formulate secrecy properties with temporal restrictions when we discuss them in a timed framework. A value \(v\) is secret for a protocol if for every bundle \(C\) of the protocol the penetrator cannot receive \(v\) in cleartext until some time \(t\); that is, there is no node \(n\) in \(C\) such that \(\text{term}(n) = v\) and \(\text{time}(n) \leq t\). For Kerberos V, we mainly discuss the secrecy of a long-term key of a regular agent, and \(authK\), \(servK\) issued by servers.

Authentication properties are specified as usual: for a protocol if for every bundle \(e\) acting as an agent, for a certain vector of parameters \(\bar{x}\), if each time principal \(B\) completes a run of the protocol as a responder using \(\bar{x}\) supposed with \(A\), then there is a run of the protocol with \(A\) acting as an initiator using \(\bar{x}\) supposed with \(B\). And this is formalized as follows: there is a responder strand \(Resp(\bar{x})\) and the \(i\)-th node of the strand is in a bundle \(C\), then there is an initiator strand \(Init(\bar{x})\) and some \(j\)-th node of the initiator strand is in \(C\).

In order to prove the secrecy of a long-term key \(K_{A,}\), we only need use the well-founded induction principle on bundles. But the knowledge closure property on penetrators is needed when we prove the secrecy of some session key \(authK\) or \(servK\). For instance, in order to prove the secrecy of \(authK\), we construct a set
\[
M = \#\left\{ \left\{ A, Tgs, authK, T_s \right\}_{k_{tgs}}, \left\{ A, Tgs, authK, T_s \right\}_{k_{tgs}} \right\} \cup \left\{ t | authK \not\subset t \right\}.
\]
We will show that for any node \( m \) in a Kerberos bundle \( B \), if \( \text{authK} \sqsupset \text{term}(m) \) and \( \text{time}(m) \leq T_a + \text{shrKcracktime} \), then \( \text{term}(m) \) must be in \( \text{synth}(M) \). Intuitively, this fact holds because both the penetrator and regular strands can only emit a message which is in \( \text{synth}(M) \). The penetrator can never decrypt or crack the messages

\[
\text{\{[\langle A, Tgs, \text{authK}, T_a \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs}
\]

until time \( T_a + \text{shrKcracktime} \), so it can only synthesize any messages which is in \( \text{synth}(M) \); except a unique authentication server strand, any other regular strand can not emit any message which has \( \text{authK} \) as a subterm until that time. But for the authentication server strand, he can only emit

\[
\text{\{[\langle A, Tgs, \text{authK}, t_i \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs}
\]

which is still in \( \text{synth}(M) \). Our formal proof is by contradiction. If not so, by the well-founded induction principle on \( B \), we have a minimal element \( m \) such that \( \text{authK} \sqsupset \text{term}(m) \) and \( \text{term}(m) \notin \text{synth}(M) \). By the knowledge closure property, we can exclude the cases when \( m \) is in a penetrator strand. By case analysis on the form of the trace of regular strands, we can also exclude the case when \( m \) is in a regular strand. Thus, a contradiction is concluded.

In the following two sections, we give the detailed proof on the secrecy and authentication properties to show how to apply the proof techniques aforementioned. Note that we also have formalized all the proofs in Isabelle/HOL, and the proof scripts can be obtained at [6]. The paper proof here can be viewed as a text account of the mechanical proof scripts at [6].

4. Proving Secrecy Goals

In Kerberos V, a long-term key of a regular agent is never sent in the network, so it cannot be compromised. Let \( B \) be a bundle of Kerberos V. For any node in the bundle, the long-term key of a regular agent cannot be a part of the term of the node. In order to prove this lemma, we only need the well-founded induction principle on bundles.

**Lemma 3** Let \( n \in B \). If \( n \notin \text{Bad} \), then \( \text{term}(n) \neq K_{\alpha} \).

**Proof.** Let

\[
P = \{ x \mid x \in B \land K_{\alpha} \sqsupset \text{term}(x) \}
\]

We show that \( P \) is empty by contradiction. If there is a node \( n \in P \), then by the well-foundedness of a bundle, there exists a node \( m \) such that \( m \) is minimal in \( P \). Namely, \( m \in B \), \( K_{\alpha} \sqsupset \text{term}(m) \), and for all \( m' \in B \), if \( m' \preceq m \) then \( K_{\alpha} \sqsupset \text{term}(m') \).

We prove that the sign of \( m \) is positive. If \( \text{sign}(m) = - \), then by upward-closed property of a bundle there must be another node \( m'' \) in the bundle \( B \) such that \( \text{sign}(m'') = + \) and \( m'' \rightarrow m \). This contradicts with the minimality of \( m \). Then \( m \) is either in a regular strand or in a penetrator strand.

- **CASE 1:** \( m \) is in a regular strand.

There are six cases. Here we only analyze the cases when \( m \) is in an authentication server strand such as \( \text{as} \in \text{AS} \) \( \{[\langle A, Tgs, \text{authK}, t_i \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs} \) or \( m \) is in a client strand \( i \in \text{Ag-I} \{[\langle i, A, \text{authK}, \text{authTicket}, B, \text{servK}, T_a, \text{servTicket}, t_i \rangle \}_{g, K}^{Kgs} \}. \) The other cases are either straightforward or can be analyzed in a similarly.

If \( m \) is in an authentication server strand such that \( \text{as} \in \text{AS} \{[\langle A, Tgs, \text{authK}, t_i \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs} \), by inspection on the trace form of the strand, we have \( m = (a, 1) \), \( K_{\alpha} \sqsupset \text{term}(a, 1) \), and

\[
\text{term}(a, 1) = \text{\{[\langle A, Tgs, \text{authK}, t_i \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs}
\]

then

\[
K_{\alpha} \sqsupset \text{\{[\langle A, Tgs, \text{authK}, t_i \rangle \}_{g, K}^{K}, \text{term}(m) \}_{g, K}^{Kgs}
\]

In both cases, we can conclude that \( K_{\alpha} \sqsupset \text{authK} \). But this contradicts with Axiom 4. If \( m \) is in a client strand such that \( i \in \text{Ag-II} \{[\langle i, A, \text{authK}, \text{authTicket}, B, \text{servK}, T_a, \text{servTicket}, t_i \rangle \}_{g, K}^{Kgs} \}. \) By inspection on the trace form of the client strand, we have \( m = (i, 0) \), \( K_{\alpha} \sqsupset \text{term}(i, 0) \), and

\[
\text{term}(i, 0) = \text{\{[\text{authTicket}, \langle A, t_i \rangle \}_{\text{authK}}, B \}
\]

then \( K_{\alpha} \sqsupset \text{authTicket} \). But by the definition of the client strand, there exists some client strand \( i, \) such that \( i \mapsto i \) and \( \text{Ag-I} \{[\langle i, A, Tgs, \text{authK}, T_a, \text{authTicket}, t_i \rangle \}_{g, K}^{Kgs} \}. \) From the definition of the strand, we have \( \text{authTicket} \sqsupset \text{term}(i, 1) \). From this and \( K_{\alpha} \sqsupset \text{authTicket} \), we have \( (1) \text{ } K_{\alpha} \sqsupset \text{term}(i, 1) \). From \( i \mapsto i \), we have \( (2) \text{ } (i, 0) \mapsto (i, 0) \). From \( (1) \text{ and } (2) \), we can conclude that \( m \) is not minimal in \( P \). This contradicts with the minimality of \( m \).

- **CASE 2:** \( m \) is in a penetrator strand \( p \).

Here we only analyze the cases when \( p \) is either \( K_{\alpha} \) (key strand) or \( C_{g,h} \) (concatenation). Other cases are either straightforward or can be analyzed in a similar way.

- \( p \) is \( K_{\alpha} \). We have \( m = (p, 0) \) and \( K_{\alpha} \sqsupset K \). Then \( K_{\alpha} \sqsupset K \in \text{K}_p \). This contradicts with Axiom 1.

- \( p \) is \( C_{g,h} \). We have \( m = (p, 2) \) and \( K_{\alpha} \sqsupset \text{g.h} \).

By the definition of \( \sqsupset \), we have \( K_{\alpha} \sqsupset g \), or \( K_{\alpha} \sqsupset h \). If \( K_{\alpha} \sqsupset g \), then \( K_{\alpha} \sqsupset \text{term}(p, 0) \). This contradicts with the minimality of \( m \). The case when \( K_{\alpha} \sqsupset h \) can be analyzed similarly.
If an authentication ticket $[[A, Tgs, authK, T_t]]_{K_{M}}$ or $[[A, Tgs, authK, T_{a}]]_{K_{M}}$ occurs as a subterm of a node in $B$, $A$ is not compromised, and $Tgs$ is a ticket granting server, then it can be guaranteed that there must be an authentication server strand as in which $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ and $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ originate at time $T_{a}$. Therefore, $T_{a}$ is the earliest time when $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ and $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ occur in $B$. With the specification of the origin of the key $authK$ by $Kas$ (formulated by Axiom 4), we also are ensured that $T_{a}$ is the earliest time when $authK$ occurs in $B$. The minimal property of $T_{a}$ will be used in the proof of Lemma 5.

**Lemma 4** Let $n \in B \cup \{A \neq Bad\}$, and $Tgs \in TGSs$. If $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(n)$, then there exists an authentication server strand as such that $\text{AS}[[as, A, Tgs, authK, t_{o}, T_{a}]]$ for some $t_{o}$, $(as,1) \in B$, and $T_{a} \leq \text{time}(n)$.

**Proof.**

Here we only prove the case $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(n)$. The other case can be proved in a similar way. First we prove that (1) $n$ is an unsolicited test for the term $[[A, Tgs, authK, T_{a}]]_{K_{a}}$. We only need prove that $K_{a}$ must be regular w.r.t. $n$. By Lemma 3, there is no node $m \in B$ such that $\text{term}(m) = K_{a}$, so $K_{a}$ must be regular w.r.t. $n$.

From (1), by Lemma 1, there exists a positive regular node $n$ in $B$ such that $m \leq_{g} n$ and $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(n)$ and $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(m)$ for any node $m$ such that $m \leq_{g} n$.

From $m \leq_{g} n$ and $B$ is a bundle, we can easily conclude $\text{time}(m) \leq \text{time}(n)$ and $m \in B$.

Now we prove that $m$ must be in an authentication server strand. From the fact that $m$ is regular, then we have six cases, here we select two cases when $m$ is in an authentication server strand as such that $\text{AS}[[as, A, Tgs, authK, t_{o}, T_{a}]]$ or in an ticket granting server strand $Tgs$ such that $\text{TGS}[tgst, A', Tg', authK', servK, B', T_{a}', t_{0}', t_{1}']$. By inspection on the form of the strand, $m = (as,1)$ because $m$ is positive. Obviously

\[
\text{term}(m) = \left[ [[A', Tgs', authK', T_{a}]]_{K_{M}} \cup [[A', Tgs', authK', t_{1}]]_{K_{M}} \right]
\]

By $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(m)$, we have either (2) $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset [[A', Tgs', authK', T_{a}]]_{K_{M}}$ or (3) $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset [[A', Tgs', authK', t_{1}]]_{K_{M}}$. From (2), we have $A = A'$ and $Tgs = Tgs'$ and $authK = authK'$ and $T_{a} = t_{1}$, so $\text{AS}[[as, A, Tgs, authK, t_{o}, T_{a}]]$. Case (3) can be proved similarly.

* $m$ is in an ticket granting server strand such that $TGS[[A, Tgs, authK', servK, B, T_{a}, T_{o}, t_{0}, t_{1}]]$. By inspection on the form of the strand, $m = (Tgs,1)$ because $m$ is a positive node. Obviously

\[
\text{term}(m) = \left[ [[A, Tgs, servK, T_{a}]]_{K_{M}} \cup [[A, Tgs, servK, t_{1}]]_{authK} \right]
\]

From $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(m)$, we have either (2) $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset [[A', Tgs', authK', T_{a}]]_{K_{M}}$ or (3) $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset [[A', Tgs', authK', t_{1}]]_{K_{M}}$. From (2), we can prove that $Tgs = B$, then by the assumption $Tgs \in TGSs$, we have $B \in TGSs$. But by the definition of the ticket granting server, we have $B \neq TGSs$. Therefore a contradiction is obtained. Case (3) can be proved similarly.

Once the authentication tickets $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ or $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ are created by the authentication server $Kas$ at $t_{a}$, then the session key $authK$ will not be compromised until the time $T_{a} + shrKcracktime$.

**Lemma 5** Let $n \in B$, $A \neq Bad$, and $Tgs \in TGSs$. If $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ or $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(n)$, then for any node $m \in B$ such that $\text{time}(m) \leq T_{a} + shrKcracktime$, $\text{term}(m) \neq authK$.

**Proof.** First we define two sets.

\[
M = \{ [[A, Tgs, authK, T_{a}]]_{K_{a}} \cup \{ \text{term}(T_{a}) \} \}
\]

\[
P = \{ m \in B \wedge \text{time}(m) \leq T_{a} + shrKcracktime \wedge \text{term}(m) \not\in \text{synth}(M) \}
\]

We show that for any node $m \in B$ such that $\text{time}(m) \leq T_{a} + shrKcracktime$, $\text{term}(m) \in \text{synth}(M)$. In order to prove this, we only need show $P$ is empty. We prove the assertion by contradiction. If $P$ is not empty, then by the well-foundedness of a bundle, (1) there exists a positive node $m$ such that $m \in B$, $\text{term}(m) \not\in \text{synth}(M)$, $\text{time}(m) \leq T_{a} + shrKcracktime$, and for all $m \in B$, if $m \neq m$, then $\text{term}(m) \in \text{synth}(M)$.

First from the fact that $[[A, Tgs, authK, T_{a}]]_{K_{a}}$ or $[[A, Tgs, authK, T_{a}]]_{K_{a}} \sqsubset \text{term}(n)$ by the Lemma 4, then
there exists an authentication server strand \( as \) such that \( AS[as, A_{Tgs}, authK, t_0, T_a] \) for some \( t_0, (as,1) \in B \), and \( T_a \leq time(n) \). From the definition of \( AS \) and Axiom 4, we have (2) \( authK \) uniquely originates at \((as,1)\) and \( time(as,1) = T_a \).

Next we prove that (3) \( \text{suite}(M, authK, m, (as,1), B) \). Here we need show both \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \) and \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \) are components in \( B \). From \( A \not\in Bad \) and \( Tgs \in TGSSs \), by Lemma 3, we have that neither \( K_{Tgs} \) nor \( K_{Tgs} \) is compromised, and they are symmetry, therefore \( \text{regular}(K_{Tgs}^{-1}, m, B) \) and \( \text{regular}(K_{Tgs}^{-1}, m, B) \); furthermore from \( time(m) \leq T_a \), shrKcracktime, and by Axiom 2, \( cracktime(K_{Tgs}) = \text{shrKcracktime} \), with (2), we have \( time(m) \leq time(as,1) + cracktime(K_{Tgs}) \), similarly we have \( time(m) \leq time(as,1) + cracktime(K_{Tgs}) \), so (3) is proved.

From (1), we have for any \( m' \) such that \( m' \Rightarrow^* m \), \( \text{term} (m') \in \text{synth}(M) \). With (2)(3), by Lemma 2, we have \( m \) must be in a regular strand \( i \), then there exist six cases. Here we analyze the cases when \( AS[i, A, Tgs', authK, t_0, t_i] \), other cases are more simpler. If \( m \) is in an authentication server strand \( AS[i, A, Tgs', authK, t_0, t_i] \). By inspection on the form of the strand, \( m = (i,1) \) because \( m \) is positive. Obviously \( \text{term} (m) = \{[A, Tgs', authK', t_i]_{K_{Tgs}}, [A, Tgs', authK', t_i]_{K_{Tgs}}\} \).

Obviously \( authK \not\subseteq \text{term} (m) \), otherwise \( time(m) \in M \). Therefore \( authK \not\subseteq \{A, Tgs', authK', t_i\}_{K_{Tgs}} \) or \( authK \not\subseteq \{A, Tgs', authK', t_i\}_{K_{Tgs}} \), then \( authK = authK' \).

From the definition of Axiom 4, we have \( authK \) uniquely originates from the strand \( i \). Combining with (2), we have \( as = i \), then \( A = A \), \( Tgs = Tgs', \) \( t_i = T_a \), so

\( \text{term} (m) = \{[A, Tgs, authK, T_a]_{K_{Tgs}}, [A, Tgs, authK, T_a]_{K_{Tgs}}\} \) \in \text{synth}(M) \)

This contradicts with the fact \( \text{term} (m) \not\in \text{synth}(M) \).

Therefore for any node \( m \in B \) such that \( time(m) \leq T_a + shrKcracktime \), \( \text{term}(m) \in \text{synth}(M) \). Next we only need prove that \( authK \in \text{synth}(M) \). We prove by contradiction, if \( authK \in \text{synth}(M) \), by the rule inversion of definition of \( \text{synth} \), we have \( authK \in M \), this contradicts with the definition of \( M \).

In order to prove the conclusion of Lemma 5, we need the conclusion of Lemma 4, which ensures us that a penetrator cannot crack the term \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \) or \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \) to obtain \( authK \). Because the earliest time when \( authK \) occurs in \( B \) is \( T_a \) and \( authK \) can only occur in \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \) the penetrator cannot crack such a term until \( T_a + shrKcracktime \), and what he can only do is to synthesize some term from \( M \). Therefore, \( authK \) must be safe until that time. Furthermore, the intermediate result of this proof tells us that \( \text{term}(m) \) must be in \( \text{synth}(M) \) for any node \( m \in B \) such that \( time(m) \leq T_a + shrKcracktime \).

If both the tickets

\( \{A, B, servK, T_s\}_{authK} \) and \( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \)

occur as a part of the term of a node in \( B \), \( A \) and \( B \) are not compromised, and \( B \) is not a ticket grant server, and \( authK \) is still not compromised at the time when the above two tickets occur, then it can be guaranteed that \( A \) must have passed the first and second phases of the protocol, and a ticket grant server strand \( Tgs \) must exist in \( B \), where two tickets

\( \{A, B, servK, T_s\}_{authK} \) and \( \{A, B, servK, T_s\}_{authK} \)

are issued for some session key \( authK \). Similar to Lemma 4, this lemma ensures us that \( T_s \) is the earliest time when \( servK \) occurs in \( B \), and this minimal property is needed in the proof of Lemma 7.

**Lemma 6** Let \( m, n \in B \), \( A \not\in Bad \), \( Tgs \in TGSSs \). If both

\( \{A, Tgs, authK, T_a\}_{K_{Tgs}} \not\subseteq \text{term}(m) \),

and

\( \{A, B, servK, T_s\}_{authK} \not\subseteq \text{term}(n) \)

and \( time(n) \leq T_a + shrKcracktime \), then there exists a ticket granting server strand \( Tgs \) such that \( TGS\{Tgs, A, Tgs, authK, servK, B, T_s, T_a, t_0, t_i\} \) for some \( T_a, t_0, (Tgs,1) \in B \) and \( T_a \leq time(n) \).

Here we only give the proof sketch of this lemma. First we need show that (1) \( n \) is an unsolicited test for \( \{A, B, servK, T_s\}_{authK} \) in \( B \). We need prove \( \text{regular}(authK, n, B) \). This can be ensured by Lemma 5. Because \( time(n) \leq T_a + shrKcracktime \), we have \( \text{term}(n) \not= authK \) for any node \( n' \) such that \( time(n') \leq time(n) \). From (1), we can show that there is a regular node \( n' \) such that \( n' \leq n \) and

\( \{A, B, servK, T_s\}_{authK} \not\subseteq \text{term}(n') \) and
\[
\{[A,B,\text{servK},T_5]\sub{\text{authK}} \in \text{term(m')}\} \quad \text{for any node } m' \text{ such that } m' \geq n. \quad \text{By the case analysis on the form of regular strands, we can prove that } n \text{ must be in a ticket granting server strand tgs such that } TGS[\text{tgs, A, Tgs, authK, servK, B, T_6, t_0, t_1}] \text{ for some } T_6, \quad t_0, \quad (\text{tgs,1}) = n \quad \text{and } T_5 = \text{time(n)}.
\]

Moreover, by the fact a ticket \( \{[A,B,\text{servK},T_5]\sub{\text{authK}} \) is originated at time \( T_5 \), then the session key \( \text{servK} \) will not be compromised until the time \( T_5 + \text{sessionKcrktime} \).

Because during the interval from \( T_5 \) to \( T_5 + \text{sessionKcrktime} \), neither \( \{[A,B,\text{servK},T_5]\sub{\text{authK}} \) will be cracked, nor the session key \( \text{authK} \) can be obtained by a penetrator to decrypt the ticket \( \{[A,B,\text{servK},T_5]\sub{\text{authK}} \).

**Lemma 7** Let \( m, n \in B, A, B \notin \text{Bad}, Tgs \in TGSs \). If both \( \{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(m)} \) and \( \{[A,B,\text{servK},T_5]\sub{\text{authK}} \subseteq \text{term(n)} \),

and \( \text{time(n)} \leq T_5 + \text{shrKcrktime} \), then for any node \( m \in B \) such that \( \text{time(m)} \leq T_5 + \text{sessionKcrktime} \), \( \text{term(m)} \neq \text{servK} \).

**Proof.** First we define:

\[
M = \{[A,B,\text{servK},T_5]\sub{\text{authK}} \} \cup \{[t] \mid \text{servK} \subseteq t\}.
\]

We will show that for any node \( m \in B \) such that \( \text{time(m)} \leq T_5 + \text{sessionKcrktime} \), \( \text{term(m)} \subseteq \text{synth}(M) \). We prove the assertion by contradiction.

Let \( P = \{m, m \in B \land \text{time(m)} \leq T_5 + \text{sessionKcrktime} \land \text{term(m)} \subseteq \text{synth}(M)\}. \) If \( P \) is not empty, then by the well-foundedness of a bundle (1), there exists a positive node \( m \) such that \( m \in B \), \( \text{time(m)} \leq T_5 + \text{sessionKcrktime} \), \( \text{term(m)} \subseteq \text{synth}(M) \), and for all \( m' \in B \), if \( m' \leq a m \) then \( \text{term(m')} \subseteq \text{synth}(M) \).

From the fact \( \{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(m)} \}, \) by Lemma 4, there exists an authentication server strand \( \text{as} \) such that \( \text{AS}[\text{as},A,Tgs,\text{authK},t_0,T_6] \) for some \( t_0, \quad (\text{as,1}) \in B \). From the definition of \( \text{AS} \), we know (2) \( \text{authK} \) uniquely originates at \( (\text{as,1}) \) and \( \text{time(as,1)} = T_5 \).

From the fact that \( \{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(m)} \} \) and \( \{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(n)} \}, \) by Lemma 6, then there exists a ticket granting server strand tgs such that \( TGS[\text{tgs, A, Tgs, authK, servK, B, T_6, t_0, t_1}] \) for some \( T_6, \quad t_0, \quad t_1 \). From the definition of \( \text{TGS} \) and Axiom 5, we have (3) \( \text{servK} \) uniquely originates at \( (\text{tgs,1}), \quad \text{time(tgs,1)} = T_5 \),

\[
\{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(tgs,0)} \}, \quad \text{and } T_5 + \text{sessionKcrktime} \leq T_5 + \text{shrKcrktime} \). From \( \{[A,Tgs,\text{authK},T_5]\sub{\text{authK}} \subseteq \text{term(tgs,0)} \}, \) by Lemma 4, we can easily conclude that \( T_5 \leq \text{time(tgs,0)} \), then (4) \( T_5 \leq \text{time(tgs,1)} \).

Next we prove that (5) \( \text{suite}(M,\text{servK},m,(\text{tgs,1}),B) \). Here we need work both \( \{[A,Tgs,\text{servK},T_5]\sub{\text{authK}} \} \) and \( \{[A,B,\text{servK},T_5]\sub{\text{authK}} \} \) are components in \( B \). From \( B \in \text{Bad} \), by Lemma 3, we have \( K_g \) are never compromised, similar to counterpart in Lemma 5, we can prove that \( \text{regular(K}_{g_1},m,B) \). From \( T_5 + \text{sessionKcrktime} \leq T_5 + \text{shrKcrktime} \), and \( \text{time(m)} \leq T_5 + \text{sessionKcrktime} \), we have (6) \( \text{time(m)} \leq T_5 + \text{shrKcrktime} \). From \( \text{time(m)} \leq T_5 + \text{sessionKcrktime} \), and by Axiom 4, we have \( \text{cracktime( authK )} = \text{sessionKcrktime} \), and \( \text{time(m)} \leq T_5 + \text{cracktime( authK )} \), with (3), we have \( \text{time(m)} \leq \text{time(tgs,1)} + \text{cracktime( authK )} \). Therefore (5) is proved.

From (1), we have for any \( m' \) such that \( m' \Rightarrow m \), \( \text{term(m')} \subseteq \text{synth}(M) \). With (2),(5), by Lemma 2, we have \( m \) must be in a regular strand \( \text{as} \), then there exists six cases. Here we analyze the cases when \( \text{AS}[i',A',Tgs',\text{authK}',i_0',t_1'] \) or \( \text{TGS}[\text{as}',A',Tgs',\text{authK}',\text{servK}',B',T_6',t_0',t_1'] \), other cases are more simpler.

If \( \text{AS}[i',A',Tgs',\text{authK}',i_0',t_1'] \), then by inspection on the form of the strand, \( m = (i,1) \) because \( m \) is positive. Obviously

\[
\text{term(m)} = \{[A,Tgs',\text{authK}',t_1']_{\text{as}',i_0'},[A,Tgs',\text{authK}',t_1']_{\text{as}},[A,Tgs',\text{authK}',t_1']_{\text{as}}\}.
\]

Obviously \( \text{servK} \subseteq \text{term(m)} \), otherwise \( \text{term(m)} \in M \). Therefore \( \text{servK} \subseteq \{[A,Tgs',\text{authK}',T_5]\sub{\text{authK}} \} \) or

\[
\text{servK} \subseteq \{[A,Tgs',\text{authK}',T_5]\sub{\text{authK}} \} \quad \text{for any node} \quad m \quad \text{such that} \quad \text{time(m)} \leq \text{time(tgs,1)} + \text{cracktime( authK )} \quad \text{Therefore (5) is proved}.
\]
strand, obviously this is a contradiction.

If $TGS(i, A, Tgs', authK', servK', B, T_g, t_o, t')$ , then by similar argument, we can prove that $servK = servK'$ from Axiom 5, we have $servK$ uniquely originates from $i$ , with (3), we have $i = tgs$ , we can prove that
term $(m) = \left\{ [A, B, servK, T_s]_{SGK}, [A, B, servK, T_s]_{authK} \right\}$

then term $(m) \in synth(M)$ this contradicts with (1).

At last we only need prove that term $(m) \in synth(M)$ implies that term $(m) \neq servK$ , this is similar to counterpart in Lemma 5.

Both Lemma 6 and Lemma 7 have the assumption that $\left\{ [A, Tgs, authK, T_s]_{KGK} \right\}$ is a subterm of the node $n$ , which can guarantee that authK must be a session key originated by an authentication server strand. The assumption $time(n) \leq T_s + shrKcracktime$ is used to guarantee that authK is still safe at $time(n)$ . Besides, the two terms $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ and $\left\{ [A, Tgs, authK, T_s]_{KGK} \right\}$ are intelligible for the client $A$ , so these two lemmas are secrecy properties in the view of $A$ .

In Lemmas 6 and 7, both $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ and $\left\{ [A, Tgs, authK, T_s]_{KGK} \right\}$ are unintelligible for an application server $B$ because authK and $K_G$ cannot be known by $B$ . So the two properties are not in $B$ 's view. $B$ can only receive a message such as $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ , can it be ensured that servK is confidential when $B$ receives the message $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ ? The following two lemmas are about the confidential information inferred from the message $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ . They are secrecy properties in $B$ 's view.

Once a server ticket such as $\left\{ [A, B, servK, T_s]_{SGK} \right\}$ occurs in a bundle, where $A$ and $B$ are not compromised, and $B$ is not a ticket granting server, then conclusions similar to those in Lemma 6 and Lemma 7 can be drawn.

**Lemma 8** Let $n \in B \backslash A$ , $B \notin TGS$ . If $\left\{ [A, B, servK, T_s]_{SGK} \right\} \subset term(n)$ , then there exists a ticket grant server strand $tgs$ such that $TGS[tgs, A, Tgs, authK, servK, B, T_g, t_o, t', T_s]$ for some $Tgs, authK, T_g, t_o, t', (tgs, 1) \in B$ and $T_s \leq time(n)$ .

**Lemma 9** Let $n \in B \backslash A$ , $B \notin TGS$ . If $\left\{ [A, B, servK, T_s]_{SGK} \right\} \subset term(m)$ , then for any node $m \in B$ such that $\left\{ [A, T_s, T_s + sessionKcracktime] \right\}$ , term $(m) \neq servK$ .

Here we summarize the main ideas used in the above proof of secrecy properties.

* For a long-term key of a regular agent, its secrecy is easily inferred because it is never sent as a part of a message. We only need the well-founded induction principle on bundles to prove this.

* But for a short session key authK or servK, the cases are more complex because they are sent as a part in a message such as

$$\left\{ [A, T_s, authK, T_s] \right\}_{KGK} \text{ or } \left\{ [A, B, servK, T_s] \right\}_{authK} .$$

In kerberos V, a session key such as authK (servK ) occurs as a part of a term of node $n$ which is of the form $\left\{ [h]_{K_0} \right\}$ , where $K_0$ can be either a long-term key or another short session key, and $h$ also contains a timestamp $t$ such as $T_s(T_g)$ , which indicates the time when $\left\{ [h]_{K_0} \right\}$ is $t$ . As mentioned before, both secrecy of $K$ and recency of $\left\{ [h]_{K_0} \right\}$ should be guaranteed. Secrecy of $K$ can be directly drawn from other lemmas on $K$ . But for recency checking, firstly we need prove that the timestamp $t$ indeed indicates the time when $\left\{ [h]_{K_0} \right\}$ is originated. Lemmas 4, 6, 9 play a role in guaranteeing that $t$ is the first time when authK (servK ) is originated. From this and the assumption that $time(n) \leq t + cracktime( K)$ , the recency of $\left\{ [h]_{K_0} \right\}$ can be proved.

5. **Proving Authentication Goals**

For convenience, we call that a strand $i$ uses a term $\left\{ [h]_{K_0} \right\}$ as an unsolicited test if there is a node $n$ is in the strand $i$ and is an unsolicited test for $\left\{ [h]_{K_0} \right\}$ in a bundle $B$ . Because a guarantee of the existence of a regular node can be drawn from an unsolicited test, a regular agent uses unsolicited test to authenticate its regular protocol participant in Kerberos V.

The client strand in the authentication phase receives $\left\{ [A, Tgs, authK, T_s] \right\}_{KGK}$ as an unsolicited test that authenticates the positive node of the authentication server strand. The intuition behind this authentication is quite straightforward. By case analysis on the form of $i$ , we have $\left\{ [A, Tgs, authK, T_s] \right\}_{KGK} \subset term(i,1)$ , combining with the assumption that $A$ 's is not compromised, by Lemma 4, we have $\left\{ [A, Tgs, authK, T_s] \right\}_{KGK}$ can only be originated by an authentication server. For the sake of brevity, in the following discussion we use $P(x,*,...,y)$ to denote $\exists x.P(x,x',...,y) . B$ is a bundle of Kerberos V.

**Lemma 10** Let $A \notin B$ . If $i$ is a client strand in the authentication phase such that $Ag-I[i, A, Tgs, authK, T_g, authTicket, t_o, t']$ and $(i,1) \in B$ , then there exists an authentication server strand as such as $AS[as, A, Tgs, authK,.*, T_s]$ , $(as,1) \in B$ .

The ticket grant server strand uses $\left\{ [A, T_s] \right\}_{authK}$ as an unsolicited test to authenticate the client strand in the authorization phase. This guarantee is ensured from the secrecy of authK , which is in turn guaranteed by the ticket $\left\{ [A, authK, T_s] \right\}_{KGK}$ . By the trace specification of a ticket grant server strand, we have that
\[
\left\{ \left[ A, B, \text{auth}K, T_a \right], \left[ A, T_a \right]_{\text{auth}K} \right\}_{K_{\text{tg}S}} \supset \left[ A, T_a \right]_{\text{auth}K}
\]
is received by Tgs earlier than the time

\[ T_a + \text{sessionKcrktime}, \]

by Lemma 5, authK is safe at that time.

**Lemma 11** Let \( A \not\in \text{Bad} \), Tgs ∈ TGSs. If tgs is a ticket granting server strand such that TGS[tgs,A,Tgs,authK,\text{servK},B,T_a,T_i,0,t_i]\ (and \( (0,0) \in B \)), then there exists a client strand \( i \) in the authorization phase such that \( (i,0) \in B \) and Ag-II[\( i, A, \text{auth}K \), \( i, T_a \), \( i, t_0 \), \( i, t_1 \)].

**Proof.** By analysis on the form of tgs strand, we have (1) \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset (\text{tg}S,0) \), (2) \( \left[ A, B, \text{auth}K, T_a \right]_{K_{\text{tg}S}} \sqsubset \text{term}(tgs,0) \) and \( \text{time}(tgs,0) \leq T_a + \text{shrkKcrktime} \).

From (2), by Lemma 5, we have that \( \text{term}(m) \neq \text{auth}K \) for any \( m \in B \) such that \( \text{time}(m) \leq \text{time}(tgs,0) \), therefore regular(\text{auth}K,(tgs,0),B) . With (1), by Lemma 1, we have (3) there is a positive regular node \( m \) such that \( m \leq \text{term}(tgs,0) \) and \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \text{term}(n) \) and \( \left[ A, T_a \right]_{\text{auth}K} \sqsubset \text{term}(m) \) for any node \( m \) such that \( m \leq \text{term}(tgs,0) \). Obviously \( m \in B \).

Now we need prove that \( m \) must be in a client strand \( i \) in the authorization phase. From the fact that \( m \) is regular, then we have six cases, here we select two cases when (4) \( m \) is in a strand \( i \) such that Ag-II[\( i, A, \text{auth}K \), \( i, T_a \), \( i, t_0 \), \( i, t_1 \)] or (5) \( m \) is in a strand \( i \) such that Ag-III[\( i, A, \text{servK}, \text{servTicket}, \text{term} i, t_0 \)]. Other cases are more simpler.

If (4) holds, then \( m = (i,0) \) because \( m \) is positive. From \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \text{term}(n) \) and \( \text{term}(m) = \left[ \text{authTicket} \left', \left[ A, T_a \right]_{\text{auth}K} \right] \right \), we have either (6) \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \text{authTicket} \left' \right \) or (7) \( \left[ A, T_a \right]_{\text{auth}K} \sqsubset \left[ A, t_2 \right]_{\text{auth}K} \).

If (6) holds, then by the definition of the client strand, there exists some client strand \( i \) such that \( i \mapsto i \) and Ag-I[\( i, A, \text{tgs} \), \( i, \text{auth}K \), \( i, T_a \), \( i, \text{authTicket} \), \( i, t_0 \), \( i, t_1 \)]. From the definition of the strand, we have

\[ \text{term}(i,1) = \left[ \text{authTicket} \left', \left[ A, \text{tgs} \right]_{\text{auth}K} \right] \right \].

From this and (6), we have (8) \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \text{term}(i,1) \). From \( i \mapsto i \), \( (i,1) \equiv (0,0) \), then (9) \( (i,1) \geq_{G} (i,0) \). But (8) and (9) contradicts with (3). If (7) holds, then \( A = A \), \( T_a = t_2 \), authK = authK. So the conclusion is obtained.

If (5) holds, then similar to the counterpart of the argument for case (4), we have either (10) \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \text{servTicket} \) or (11) \( \left[ A, T_0 \right]_{\text{auth}K} \sqsubset \left[ A, t_0 \right]_{\text{auth}K} \). For case (10), its proof is similar to that of case (6). If (11) holds, then (12) \( A = A \), \( T_a = t_2 \), authK = servK. By the definition of Ag-III, (13) there is a client strand \( i, T_a \) such that \( i \mapsto i \) and Ag-II[\( i, A, \text{auth}K \), \( i, \text{authTicket} \), \( B, \text{servK} \), \( T_s, \text{servTicket} \), \( t_2 \), \( t_3 \)] and Ag-I[\( i, A, \text{tgs} \), \( i, \text{auth}K \), \( i, T_a \), \( i, \text{authTicket} \), \( i, t_0 \), \( i, t_1 \)] for some Tgs ∈ TGSs. Obviously,

\[ \left[ A, T_s, \text{auth}K, T_a \right]_{K_s} \sqsubset \text{term}(i,1), \]

\[ \left[ A, B, \text{servK}, T_s \right]_{\text{auth}K} \sqsubset \text{term}(i,1), \]

and \( \text{time}(i,1) = T_a + \text{shrkKcrktime} \). From \( i \mapsto i \) and \( i \mapsto i \) and \( (i,0) \in B \), we have \( (i,1) \in B \) and \( (i,1) \in B \). From (12), and the assumption \( A \not\in \text{Bad} \), by Lemma 6, (14) there is a ticket granting server strand tgs such that TGS[tgs, A, Tgs, authK, \text{servK}, B, T_a, t_0, t_1] for some Tgs, t_0. But from (2), by Lemma 4, we have (15) there is an authentication server as such that AS[\( A, \text{tgs}, \text{auth}K \), \( i, T_a, \) for some \( t_0 \). But from (12) and Axioms 4,5, we have tgs = as because authK (\text{servK}) uniquely originates from a strand, obviously this is a contradiction.

A client strand \( i_2 \) in the authorization phase receives \( \left[ A, B, \text{servK}, T_s \right]_{\text{auth}K} \) as an unsolicited test. Note that \( \left[ A, B, \text{servK}, T_s \right]_{\text{auth}K} \) is received in the second node in the client strand; furthermore, from the definition of Ag-II, we have that there exists a client strand \( i_1 \) in the authentication phase such that \( i_1 \mapsto i_2 \), and the ticket \( \left[ A, T_s, \text{auth}K, T_a \right]_{K_s} \) must be received at the second node of \( i_1 \); from the definition of Ag-II, \( \left[ A, B, \text{servK}, T_s \right]_{\text{auth}K} \) must have been received at an earlier time than \( i_1 \mapsto \text{shrkKcrktime} \); then by Lemma 5, it can be guaranteed that authK must be safe at the time when the client strand receives \( \left[ A, B, \text{servK}, T_s \right]_{\text{auth}K} \).

**Lemma 12** Let \( A, B \not\in \text{Bad} \). If \( i \) is a client strand in the authorization phase such that Ag-II[\( i, A, \text{auth}K, T_a \), \( i, \text{authTicket}, B, \text{servK}, T_s, \text{servTicket}, t_2, t_3 \)] and \( (i,1) \in B \), then there exists a client strand \( i_0 \) in the authentication phase, and a ticket granting server strand tgs, and some Tgs such that \( i_0 \mapsto i \) and Ag-I[\( i_0, A, \text{tgs}, \text{auth}K, T_a, \text{authTicket}, * \) and TGS[tgs, A, Tgs, authK, \text{servK}, B, T_a, * ]] (and \( (t,1) \in B \), and \( B \not\in \text{TGSs} \)).

The application server B receives \( \left[ A, T_i \right]_{\text{servK}} \), which is an unsolicited test to guarantee that the first received message must be from a client strand in the service phase. This guarantee is ensured from the secrecy of servK, which is in turn guaranteed by the ticket \( \left[ A, B, \text{servK}, T_s \right]_{K_B} \). By the trace specification of an application server strand, we have that

\[ \left[ \left[ A, B, \text{servK}, T_s \right]_{K_B}, \left[ A, T_i \right]_{\text{servK}} \right] \]
is received by B earlier than the time
\[ T_S + \text{sessionKerberos} \].

By Lemma 9, servK is safe at that time.

**Lemma 13** Let \( A, B \notin \text{Bad} \), \( B \in \text{TGSs} \). If \( B \) is an application server strand such that \( \text{Apps}[b, A, \text{servK}, T_S, t_s, t_i, t_i] \), and \( (b, 0) \in B \), there are two client strands \( i_2 \), \( i_1 \) and some servTicket such that \( i_2 \rightarrow i_1 \) and \( (i_1, 0) \in B \) and \( \text{Ag-II}[i_2, A, *, *, B, \text{servK}, T_S, \text{servTicket}, *, *] \) and \( \text{Ag-III}[i_1, A, \text{servK}, \text{servTicket}, T_S, t_i] \).

The client strand in the service phase uses \( \{T_s\}_{\text{servK}} \) as an unsolicited test to authenticate the application server strand. This guarantee is also ensured from the secrecy of servK, which is in turn guaranteed by the ticket \( \{A, B, \text{servK}, T_S\}_{\text{authK}} \) and \( \{A, \text{Tgs}, \text{authK}, T_S\}_{\text{authK}} \). By the trace specification of an application server strand, we have that \( \{T_s\}_{\text{servK}} \) is received by A earlier than the time \( T_S + \text{sessionKerberos} \) and \( T_S + \text{shrKerberos} \).

By Lemma 7, \( \text{servK} \) is safe at that time.

**Lemma 14** Let \( A, B \notin \text{Bad} \), \( B \in \text{TGSs} \). If \( i_1 \) is a client strand in a service phase such that \( \text{Ag-III}[i_1, A, \text{servK}, \text{servTicket}, t_s, *, *,] \) and \( (1, 1) \in B \), and \( i_2 \) is a client strand in the authorization phase such that \( \text{Ag-II}[i_2, A, \text{authK}, \text{authTicket}, B, \text{servK}, T_S, \text{servTicket}, t_s, t_i] \) and \( i_2 \rightarrow i_1 \), then there exists an application server strand \( b \) such that \( \text{Apps}[b, A, \text{servK}, T_S, t_s, t_i, *, *,] \) and \( (b, 1) \in B \).

**Proof.** By analysis on the form of strand \( i_1 \) and \( i_2 \), we have (1) \( \text{term}(i_1, 1) = [t_1]_{\text{servK}} \subseteq \text{term}(i_1, 1) \). By unfolding the definition of \( \text{Ag-II} \), there exists a client strand \( i_1 \) such that \( i_1 \rightarrow i_2 \) and \( \text{Ag-I}[i_1, A, \text{Tgs}, \text{authK}, T_S, \text{authTicket}, t_s, t_i] \) and \( \text{Tgs} \in \text{TGSs} \) for some \( \text{Tgs}, T_a, t_a, t_i \). Obviously, (3) \( \{A, \text{Tgs}, \text{authK}, T_S\}_{\text{authK}} \subseteq \text{term}(i_1, 1) \). From \( i_1 \rightarrow i_2 \) and \( i_1 \rightarrow i_1 \) we can easily conclude that \( (i_1, 1) \Rightarrow^* (i_1, 1) \) and \( (i_1, 1) \Rightarrow^* (i_1, 1) \). With \( (i_1, 1) \in B \), we have \( (i_1, 1) \in B \) and \( (i_1, 1) \in B \).

With (2)(3), by Lemma 7, (4) \( \text{term}(m) \neq \text{servK} \) for any node \( m \) such that \( \text{time}(m) = T_S + \text{sessionKerberos} \).

By the definition of \( \text{Ag-III} \), we have (5) \( \text{time}(i_1, 1) \leq T_S + \text{sessionKerberos} \). From (4)(5), we have \( \text{term}(m) \neq \text{servK} \) for any node \( m \) such that \( \text{time}(m) \leq \text{time}(i_1, 1) \), therefore regular \( \text{servK} \in B \). So \( (i_1, 1) \) is an unsolicited test for \( \{T_s\}_{\text{servK}} \) in B. By Lemma 1, (5) there is a regular positive node \( m \) such that \( m \sim \text{time}(i_1, 1) \) and \( \{t_i\}_{\text{servK}} \subseteq \text{term}(m) \) and \( \{t_i\}_{\text{authK}} \subseteq \text{term}(m) \) for any node \( m \) such that \( m \sim \text{time}(i_1, 1) \). Obviously \( m \in B \). By simple case analysis, we have that \( m \) must be in an application server \( b \) such that \( \text{Apps}[b, A, B, \text{servK}, T_S, t_s, *, *,] \), \( m = (b, 1) \).

**6. Conclusions and Related Work**

Our main aim is to extend and mechanize the strand space theory to analyze Kerberos V, since mechanization in a theorem prover not only helps us model protocols rigorously and specify protocol goals without any ambiguity, it also guarantees a formal proof. Besides the
essential inherence from the classic strand space method, our work is deeply inspired by Paulson and Bella’s work. We have directly used their formalization of message algebra, and have learned a lot about the semantics of timestamps and replay attacks from [4]. However, we model and analyze protocols in strand space theory rather than in Paulson’s trace theory. In detail, we model behaviors of all the agents by strands, and mainly use the well-founded induction principle to prove properties. So in our Isabelle formalization, main efforts have been devoted to definitions and lemmas about strand space theory. e.g., we formalize strands, bundles, unique originality, the well-founded principle on bundles, and use this principle to prove important results such as unsolicited authentication test and regularity of keys.

In [4], the ability of a penetrator to crack a stale encrypted message is modelled by the Oops rule in the inductive definition of a trace, and the trace definition depends on the protocol under study. However, in the strand space theory, a penetrator’s abilities are modelled to be independent of the protocol, that is the main reason why we relate a key with a crack time, and model a penetrator’s ability of cracking a stale encrypted message by a new key cracking strand. The advantage of our method is that modelling a penetrator’s behavior remains independent and results such as the unsolicited authentication tests can be generalized.

Regarding verification of the Kerberos protocols, Mitchell et al. [7] analyzed a simplified version of the protocol by model checking, and Butler et al. [8] analyzed the Kerberos V protocol using MSR [9]. But they did not include timestamps and replay attacks in their model, in fact the former work ignored both nonces and timestamps, and the latter only considered the implementation of the Kerberos protocol basing on nonce.

7. References