Rank Theorems for Forward Secrecy in Group Key Management Protocols

Amjad Gawanmeh and Sofiène Tahar
Department of Electrical and Computer Engineering
Concordia University, 1455 de Maisonneuve West
Montreal, Quebec H3G 1M8
{amjad,tahar}@ece.concordia.ca

Abstract

Design and verification of cryptographic protocols has been under investigation for quite sometime. However, little has been done for the class of protocols that deals with group key management and distribution, and have special security properties, such as forward secrecy. In this paper, we define a formal model and establish rank theorems for forward properties based on a set of generic formal specification requirements for group key management and distribution protocols. Rank theorems imply the validity of the security property to be proved, and are deducted from a set of rank functions we define over the protocol. The above formalizations and rank theorems were implemented using the PVS theorem prover. We illustrate our approach on the verification of forward secrecy for the Enclaves protocol designed at SRI.

1 Introduction

Cryptographic protocols provide security services for communicating entities. They involve precise interactions in order to achieve the required security services, therefore, it is very important to verify that the protocol operations are not vulnerable to attacks. Besides, networks handle more and more tasks in a potentially hostile environment. Therefore, cryptographic protocols should take more responsibilities in order to capture these new requirements. Some security properties such as availability and fairness take more important roles in some protocols like in commercial systems. This requires that the complexity of the cryptographic protocol should be increased.

The general requirements for protocols involving two or three parties are well understood, however, the case is different with group key distribution protocols, where the key can be distributed among a larger number of members who may join or leave the group at arbitrary times. Therefore, security properties that are well defined in normal two-party protocols have different meanings and different interpretations in group key distribution protocols, and so they require a more precise definition before we look at how to verify them. Therefore, systems designed for two-party protocols may not be able to model a group protocol, or its intended security properties because such tools require an abstraction to a group of fixed size to be made before the automated analysis takes place. This can eliminate chances of finding attacks on the protocol.

Our verification methodology is based on the notion of rank theorems we present in this paper utilizing the *rank functions* first proposed by Ryan and Schneider [10]. We map the requirements into ranks, this map is based on a predefined function, the rank function. For this map, we have to find the appropriate rank functions for the protocol events, traces and properties. This rank function is tailored for the security property we intend to verify, in particular, forward secrecy and backward secrecy. Based on the above rank functions, we define a set of rank theorems for forward secrecy. The proof establishment is mechanized in the PVS (Prototype Verification System) theorem prover [8]. Rank theorems, protocol events and traces of execution have been defined in PVS. We apply the implemented proof environment on the Enclaves protocol from SRI [3] in order to verify related forward secrecy.

The rest of the paper is organized as follows, Section 2 provides related work to ours. In Section 3, we present the overall verification methodology and the formal specification requirements model. In Section 4, we define rank theorems, then prove the theorem for forward secrecy property. In Section 5, we describe the details of our implementation of the formal specifications and theorems in PVS. Finally, Section 7 concludes the paper with future work hints.

2 Related Work

Syverson and Meadows [13] presented the formal requirements for authentication in key distribution protocols. The requirements they provide was for a single property, authentication, which is similar in different protocols, whereas other properties may have different semantics in different classes of protocols, like secrecy property for example. Layouni *et al.* [7] used a combination of model checking (to verify authentication property), theorem proving (to verify safety and liveness properties such as proper agreement), and a Random Oracle Model (to manually prove robustness and unpredictability properties). This example shows how difficult it is to verify and analyze this class of protocols. Meadows and Syverson [5] used the NPATRL language, a temporal requirement specification language for use with the NRL Protocol Analyzer, in order to specify the Group Domain of Interpretation (GDOI) key management protocol. In a later stage Meadows *et al.* [6] gave a detailed specification of the requirements for GDOI and provided a formal analysis of the protocol with respect to these requirements using the NRL Protocol Analyzer. In a related approach, Denker and Millen [2] used multiset term rewriting in order to model group communication protocols. They show the mechanisms used in key distribution and provide an analysis of group protocols complexity in terms of key distribution. Archer [1] provided a mechanized correctness proof of the basic TESLA protocol based on establishing a sequence of invariants for the protocol using TAME. The model of the protocol is rather simple, and the proof was made under a strong assumption stating that the adversary has no initial knowledge, and can only use facts revealed by users.

In a more recent work, Pereira [9] proposed a systematic approach to analyze protocol suites extending the Diffie-Hellman key-exchange scheme to a group setting. He pointed out several unpublished attacks against the main security properties claimed in the definition of these protocols. The method provided is essentially manual and applicable only on Group Diffie-Hellman (GDH) protocols. In a similar work, Sun and Lin [12] extended the strand space theory to analyze the dynamic security of Group Key Agreement Protocols (GKAP) and discussed the conditions of the security retention in the dynamic cases of the protocol, this work treats the analysis dynamic aspects of the protocol with no reasoning about the correctness of the protocol under these dynamic events. A related work by Steel *et al.* [11] model a group key protocol by posing inductive conjectures about the trace of messages exchanged in order to investigate novel properties of the protocol and whether it results in an agreement on a single key. The method, however, is applicable on limited groups of two or three members only. Recently, Truderung [14] presents selecting theories, which extend the standard non-recursive term rewriting model and allow participants to compare and store arbitrary messages. This formalism can model recursive protocols, however, it cannot be applied on non–recursive protocols such as GDH or the Enclaves.

In [10], Ryan and Schneider proposed the idea of rank functions for the verification of CSP (Communication Sequential Process). Dutertre and Schneider [4] used an embedding of CSP in PVS in order to verify the authentication property of Needham Shroeder public key protocol. However, the work in [10] did not present a method that can be applied on security properties in other classes of protocols, like group key protocols.

3 Verification Methodology

From the above account on related work, we noticed the lack of a single formalism to model the protocols and reason about their security properties. In this section, we propose a verification methodology based on rank theorems. We use a rank function to map facts about the protocol into ranks, and define for every security property a theorem that implies the validly of the property with respect to the protocol. Figure 1 provides a summary of the steps of our verification methodology. The first step consists of providing a formal model and precise definition for group protocols properties and events. This will help eliminating the gap between the informal protocol specification and the formal model. It will also provide a well defined protocol specification that can be directly integrated into the verification methodology. In the second step, we define map functions between the set of facts and the set of integers. The set of facts include protocol events, protocol execution traces and the security property. Then we define rank theorems that provide conditions satisfied by a given rank function in order to conclude that the security property satisfies its protocol model. We define for every security property a theorem that implies the validly of the property with respect to the protocol. Rank theorems imply the correctness of the security property it models. In order to prove the correctness of a specific property, we need to prove that its corresponding rank theorem is correct with respect to the protocol model. In the final step of our approach, we implement the rank theorems in PVS and establish their proof of correctness.

In order to make sure of the soundness of our approach, we have to show the formal link between the constructed rank theorem and the formal model of the property. This is carried out by proving that the correctness of the rank theorem implies the correctness of the security property. This way, we can argue that verifying the property at the implementation level guarantees the correctness of the property in the model. In the rest of this section and throughout this paper, we will use the following notations:
$\mathbb{M}$: set of all possible messages, (messages space).
P: a honest principal who is willing to communicate.
$\mathbb{P}$: the set of knowledge of member P, $\mathbb{P} \subseteq \mathbb{M}$. $\mathbb{S}$: secret messages space, the set of all secret messages, $\mathbb{S} \subset \mathbb{M}$,

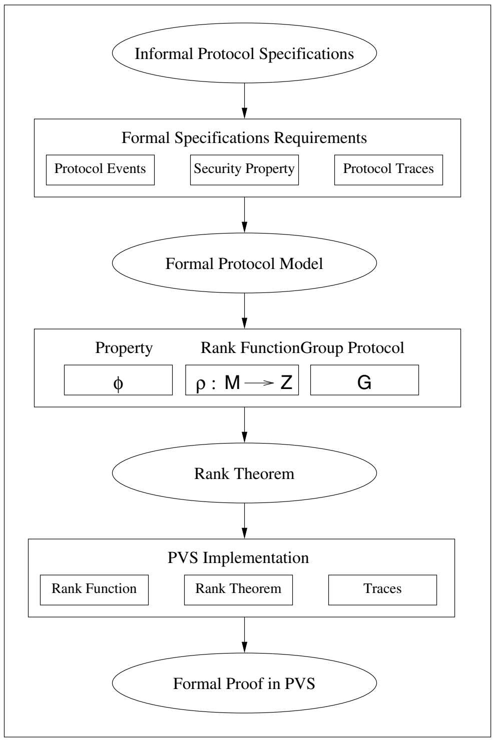

Figure 1. Verification Methodology

which we want to keep hidden from the intruder. These messages are defined by the protocol.

I: a dishonest member. We assume that the intruder is a dishonest member who is trying to find an attack in the protocol by using his/her unlimited resources and computational power.

$\mathbb{E}$: set of all events, or dynamic operations, i.e., join, leave, merge, and split. An event is a term from the message space to the message space, $\mathbb{E} : \mathbb{M} \to \mathbb{M}$.

$\mathbb{T}$: the set of all possible traces, where a trace of events is the execution of the sequence of these events. We use $\tau \in \mathbb{T}$, such that $\tau : \mathbb{E} \times \mathbb{M} \to \mathbb{M}$, then we write $m = \tau(\mathbb{E}, \mathbb{M})$ to say that a message $m \in \mathbb{M}$ is generated by the trace τ by executing the set of events $\mathbb{E}$ on the set of messages $\mathbb{M}$. We also write $\tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m$ to represent a predicate formula that evaluates to true if and only if $m = \tau(\mathbb{E}, \mathbb{M})$.

$\mathbb{K}_0$: set of initial knowledge of the intruder, where $\mathbb{K}_0 \subset \mathbb{M}$. The initial knowledge of the intruder is basically the information he/she can collect before the start of the execution of the protocol events. $\forall m \in \mathbb{M} : m \in \mathbb{S} \Rightarrow m \notin \mathbb{K}_0$. $\mathbb{K}$: the set of knowledge of the intruder. The intruder updates this knowledge by executing events starting with the initial set of knowledge, $\mathbb{K}_0 \subseteq \mathbb{K}$ and $\mathbb{K} \subseteq \mathbb{M}$.

$\mathbb{G}_t$: current group, which can be formally defined as a set of principals who share a secret key, or information that can be used to calculate the secret key. $\mathbb{G}_{t+i}$: a group that can share a secret key at future time. $\mathbb{G}_{t-i}$: a group that previously in time shared a secret key. For group membership, we say $K_{\mathbb{G}_t} \in \mathbb{P} \Rightarrow P \in \mathbb{G}_t$, which means a principal P is a member of the group $\mathbb{G}_t$ at this time, t, if the group key $K_{\mathbb{G}_t}$ is in his/her set of principal P's knowledge $\mathbb{P}$. $K_{\mathbb{G}_t}$: the group session key: the key generated for the current session. Equivalently, it can be the set of information that can be used to calculate the key. $I \notin \mathbb{G}_t \Rightarrow K_{\mathbb{G}_t} \in \mathbb{S}$. $K_{\mathbb{G}_{t+i}}$: a group session key for the group $\mathbb{G}_{t+i}$ that can be generated and used sometime in the future, $I \notin \mathbb{G}_{t+i} \Rightarrow K_{\mathbb{G}_{t+i}} \in \mathbb{S}$. $K_{\mathbb{G}_{t-i}}$: a group session key that was generated and used previously in time. $I \notin \mathbb{G}_{t-i} \Rightarrow K_{\mathbb{G}_{t-i}} \in \mathbb{S}$.

Secrecy. In the following, we give the formal definition of group secrecy, forward secrecy and backward secrecy.

Definition 3.1. ***Group key secrecy**: for any current group $\mathbb{G}_t$, and a dishonest principal I who knows a set of initial knowledge $\mathbb{K}_0$, there is no trace $\tau \in \mathbb{T}$ that he/she can execute in order to obtain the current group session key $K_{\mathbb{G}_t}$.*
$I \notin \mathbb{G}_t \Rightarrow \neg\exists\tau \in \mathbb{T}$*:* $K_{\mathbb{G}_t} = \tau(\mathbb{E}, \mathbb{M})$.

Forward secrecy requires that a session key cannot be calculated from keys and information that are generated after this key in time. Which means that compromising sessions keys does not compromise previous session keys that were established for previous protocol runs.

Definition 3.2. ***Forward secrecy**: for a group $\mathbb{G}_t$, and a dishonest principal I, where $I \in \mathbb{G}_t$ (I knows $K_{\mathbb{G}_t}$), there is no trace $\tau \in \mathbb{T}$ that he/she can execute in order to obtain a previous group session key $K_{\mathbb{G}_{t-i}}$, where $0 < i < t$.*
$I \in \mathbb{G}_t \Rightarrow \neg\exists\tau \in \mathbb{T} : K_{\mathbb{G}_{t-i}} = \tau(\mathbb{E}, \mathbb{M})$*, where* $I \notin \mathbb{G}_{t-i}$*, and* $0 < i < t$.

Backward secrecy requires that a session key cannot be calculated from keys and information that are generated before this key in time. Which means that compromising sessions keys does not compromise keys for future sessions.

Definition 3.3. ***Backward secrecy**: for any current group $\mathbb{G}_t$, and a dishonest principal I, $I \in \mathbb{G}_t$ (I knows $K_{\mathbb{G}_t}$), there is no trace $\tau \in \mathbb{T}$ that he/she can execute in order to obtain a previous group session key $K_{\mathbb{G}_{t+i}}$, where $i > 0$.*
$I \in \mathbb{G}_t \Rightarrow \neg\exists\tau \in \mathbb{T} : K_{\mathbb{G}_{t+i}} = \tau(\mathbb{E}, \mathbb{M})$*, where* $I \notin \mathbb{G}_{t+i}$ *and* $i > 0$.

Joining and Leaving Groups. Any group key distribution protocol must handle adjustments to group secrets subsequent to all membership change operations. Single member operations include member *join* or *leave*.

Definition 3.4. *A principal P joins the group $\mathbb{G}_t$ if $P \notin \mathbb{G}_t$, and there exists a trace $\tau \in \mathbb{T}$ that P can execute, where $K_{\mathbb{G}_{t+i}} = \tau(E^n, M^p)$ such that $K_{\mathbb{G}_{t+i}} \in \mathbb{P}$ (or $P \in \mathbb{G}_{t+i}$) and $K_{\mathbb{G}_{t+i}} \neq K_{\mathbb{G}_n}$.*

Definition 3.5. *A principal P leaves the group $\mathbb{G}_t$ if $P \in \mathbb{G}_t$, and there exists a trace $\tau \in \mathbb{T}$ that P can execute such that $K_{\mathbb{G}_{t+i}} \notin \mathbb{P}$ (or $P \notin \mathbb{G}_{t+i}$).*

Merging and Splitting Groups. Merging and splitting groups are considered as multiple members operations. A merge event occurs when two groups with two different settings execute a trace of events that result in a new group setting, where every member of each of the two groups is a member of a new group. Whereas a split event occurs when one group executes a trace of events that result in two new different groups, where every member of the current group is a member of one and only one of the new groups.

Definition 3.6. *A group $\mathbb{G}1_t$ merges with group $\mathbb{G}2_t$, if there is a trace $\tau \in \mathbb{T}$ that both $\mathbb{G}1_t$ and $\mathbb{G}2_t$ can execute such that $\mathbb{G}_{t+i} = \mathbb{G}1_t \cup \mathbb{G}2_t$ (which implies that $\forall\ P \in \mathbb{G}_{t+i}$, $K_{\mathbb{G}_{t+i}} \in \mathbb{P}$), where $K_{\mathbb{G}_{t+i}} = \tau(\mathbb{E}, \mathbb{M})$, $K_{\mathbb{G}_{t+i}} \neq K_{\mathbb{G}1_t}$ and $K_{\mathbb{G}_{t+i}} \neq K_{\mathbb{G}2_t}$.*

Definition 3.7. *A group $\mathbb{G}_t$ splits into groups $\mathbb{G}1_{t+i}$ and $\mathbb{G}2_{t+i}$, if there exists a trace $\tau \in \mathbb{T}$ that $\mathbb{G}_t$ can execute such that $\mathbb{G}_t = \mathbb{G}1_{t+i} \cup \mathbb{G}2_{t+i}$ and $\mathbb{G}1_{t+i} \cap \mathbb{G}2_{t+i} = \phi$, where $K_{\mathbb{G}_t} \neq K_{\mathbb{G}1_{t+i}}$ and $K_{\mathbb{G}_t} \neq K_{\mathbb{G}2_{t+i}}$.*

4 Rank Theorems for Protocol Models

Rank functions were first introduced in [10]. For the purpose of establishing the proof that a specific fact will not be available to the intruder, we assign a value or *rank* to each fact, such that, facts that can be generated by the system have positive rank, and facts that cannot be obtained by the intruder cannot have positive rank. The definition of the rank function is as follows:

Definition 4.1. *A rank function ρ is a function $\rho : \mathbb{M} \rightarrow \mathbb{Z}$ that maps the set of all messages into integers.*

The rank function should obey specific rules in order to be sound. First there are no negative ranks generated by the system. It is necessary to verify that each participant cannot introduce anything of non-positive rank to the system. In other words, the intruder initial knowledge must be of positive rank, and only facts of positive ranks can be generated from sets of facts of positive rank. All messages that are supposed to be secret and unknown to the intruder are mapped to zero rank. When executing an event, the rank of the generated message is a bounded function of the rank of the parameters of the event.

We define a property ϕ for a given group protocol $\mathbb{G}$. This property states that a dishonest user I cannot execute a trace in $\mathbb{T}$ in order to discover a secret in $\mathbb{S}$, and is formally modeled as follows: $\phi = \forall \tau \in \mathbb{T},\ \tau(E^n, M^p) \rightsquigarrow m \Rightarrow m \notin \mathbb{S}$. If this property is correct for the protocol $\mathbb{G}$ then we can write $\mathbb{G} \models \phi$. This is a general secrecy property that will be used to define and proof the rank theorem. The target security property to be verified, i.e., forward secrecy, will be concretely defined later in this section. Now, we define and prove a general *rank theorem* for this property as follows:

Theorem. *$\forall m \in \mathbb{K}, \rho(m) > 0 \Rightarrow \mathbb{G}_t \models \phi$, where $m = \tau(\mathbb{E}, \mathbb{M})$ and $\tau \in \mathbb{T}$*

This means that for all traces $\tau \in \mathbb{T}$, a dishonest principal I can execute on a group protocol $\mathbb{G}_t$. We say that the protocol satisfies a security property ϕ, $\mathbb{G}_t \models \phi$, if the protocol can maintain a positive rank for the messages that can be generated by the intruder.

Proof. We assume there exists $m \in \mathbb{K}$ such that $\rho(m) = 0$, and we show that the property ϕ is invalid. $\rho(m) = 0 \Rightarrow m \in \mathbb{S}$; $\mathbb{S}$ is a closed set, only messages in $\mathbb{S}$ have rank zero. $m \in \mathbb{K}$ and $m \notin \mathbb{K}_0 \Rightarrow \exists \tau \in \mathbb{T} : m = \tau(\mathbb{E}, \mathbb{M})$, therefore, $\tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m$ is valid. Then we can write $\exists \tau \in \mathbb{T} : \tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m \Rightarrow \rho(m) = 0$, and so $\exists \tau \in \mathbb{T} : \tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m \Rightarrow m \in \mathbb{S}$, this means $\rho(m) = 0 \Rightarrow \neg\phi$, and so $\rho(m) = 0 \Rightarrow \mathbb{G}_t \not\models \phi$ □

Now we define our forward secrecy property, φ, based on the formal specifications model presented previously as follows: $\varphi = \forall m \in \mathbb{S}, I \in \mathbb{G}_t \Rightarrow \neg\exists \tau \in \mathbb{T} : \tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m$. We use the above rank theorem in order to define a rank theorem for the forward secrecy property φ (and similarly for backward secrecy). However, we should define the rank function ρ_ϕ that maps the set of all messages to the appropriate ranks. ρ_φ is defined as follows, where $\forall i \in \mathbb{Z} : t+i \geq 0$ and $t - i \geq 0$.

$$\rho_\varphi(m) = \begin{cases} 0, & if\ m \in \mathbb{S}\ \vee\ m = K_{\mathbb{G}_{t-i}} \\ 1, & if\ m \in \mathbb{K}_0\ \vee\ m = K_{\mathbb{G}_t}\ \vee \\ & \quad (m = K_{\mathbb{G}_{t+i}}\ \wedge\ I \in \mathbb{G}_{t-i}) \end{cases}$$

This means that for the validity of forward secrecy, we give the rank zero to all messages in the set of secret messages $\mathbb{S}$, such as secrets shared between users and servers, and all the groups keys that were generated before the current group key. However, for the keys generated after the assumed dishonest user joined the group are mapped to a positive rank because they are in his/her initial set of knowledge. Now we can write the above theorem for *forward secrecy property* as follows:

Theorem. *$\forall m \in \mathbb{K}$, $\rho_\varphi(m) > 0 \Rightarrow \mathbb{G}_t \models \varphi$, where $m = \tau(\mathbb{E}, \mathbb{M})$ and $\tau \in \mathbb{T}$.*

One of the advantages of introducing such theorems, is that, first, it is protocol independent, which means that we can apply it on different protocols as well as on the same protocols at different levels of abstraction. Second, it is implementation independent, which gives more freedom to verification tool choice without any modification on the previous steps of our methodology.

5 PVS Implementation and Application

The last step of our methodology is to mechanize the proof of the rank theorem using a verification tool, for this purpose we choose the PVS theorem prover. Our model in the PVS includes a definition of the formal requirements that we defined for the events and traces of execution, the rank functions and the rank theorem of the security property. Then we prove in PVS that the rank theorem maintains a positive rank, which implies the correctness of the property with respect to the protocol.

In our implementation, we first formalized the general requirement, rank function and its lemmas, and the rank theorem. First we show the type declarations we used for our model, this includes the types of messages, events, key, a subtype of messages, and users, traces, and groups.

```
   MESSAGE : TYPE
   EVENT : TYPE = MESSAGE, MESSAGE -> MESSAGE
   KEY : TYPE FROM MESSAGE
   USER : TYPE
   TRACE : TYPE = set[EVENT]
   GROUP : TYPE
```

Then we defined the prototypes for the events protocols can execute. This includes the normal events, like send, receive, encrypt, and decrypts, in addition to the dynamic events such as join, leave, merge split. These events of the protocol are represented in PVS as a data type in order to be sure that all actions are syntactically different.

```
Event : DATATYPE
   BEGIN
      send(msg_send, m_recv: USER,
              s_msg: MESSAGE): send?
      recv(m_recv, m_send: USER,
              r_msg: MESSAGE) : recv?
      join(user: USER, group : GROUP) : join?
      leave(user: USER, group : GROUP) :leave?
      merge(x_group, y_group : GROUP): merge?
      split(group: GROUP) : split?
   END event
```

In order to define the rank function for forward secrecy property we use the predicate $inSet$ which tells if a given message belongs to a specific set of messages. This predicate is defined as follows:

```
inSet: [set[MESSAGE], MESSAGE -> bool] =
  (LAMBDA (p: set[MESSAGE], m: MESSAGE): p(m)
```

We define the rank function for forward secrecy property that initializes every message in the intruders initial set of knowledge and all the messages in the set of secret messages, this definition represents the initialization of the ranks of the messages in the initial state of the protocol, when executing the protocol, every new generated message will have a specific rank that is calculated depending on the events executed. We also show how we update ranks of newly generated messages from events in PVS as follows:

```
rank(msg:MESSAGE) : NAT =
   IF msg = id THEN 1
   ELSEIF msg = nonce THEN 1
   ELSEIF inSet(secretKey,m) THEN 0
   ELSEIF inSet(intInitKnldg,m) = THEN 1
   ENDIF
updateRank(event,m1,m2, key, u1,u2) : nat =
   CASES event OF
      concat(m1,m2)   : MIN(rank(m1),rank(m2)
      encr(m1,key)    : rank(m1)+1
      decr(m1,key)    : rank(m1)-1
      send(u1,u2,m1)  : rank(m1)
      recv(u1,u2,m1)  : rank(m1)
   ENDCASES
```

At this point we encoded our forward secrecy property and our rank theorem for this property as follows:

```
forward_secrecy : THEORY
  BEGIN
      msg  : VAR MESSAGE
      fwd_secrecy_property: LEMMA
         FORALL msg: inSet(intKnldg,msg)
            IMPLIES rank(msg) > 0
  END forward_secrecy
```

This is the basic lemma we proved based on previous definition. The set *intKnldg* is updated by the intruder, and for ever update we calculate the new rank as shown above. So the proof means that the intruder who executes any of the above defined events for the protocol cannot obtain a message with rank zero.

Enclaves Protocol. Enclaves [3] is a protocol that enables users to share information and collaborate securely through insecure networks, such as the Internet, and provides services for building and managing groups of users. Authorized users can dynamically join, leave, and rejoin an active group. The group communication service relies on a secure multicasting channel that ensures integrity and confidentiality of group communication. The group-management service consists of user authentication, access control, and group-key distribution. We apply our approach on this protocol and show the correctness of its forward secrecy property.

Then we define a preposition to show the possession of a key by a specific user after executing the necessary steps. Next, we describe the connection state of a user which indicates that a user is connected to the group if all the given premises are valid. At this point, we can instantiate our rank theorem for forward secrecy and check its validity in the protocol states. This means that starting from the initial step in the protocol, and applying any trace, the rank of any message in the intruder knowledge is positive.

```
session_key_prop: PROPOSITION
    Reachable(step_00, T)(q) AND q`users(A0) =
      Joined(N, Ka) => InUse(Ka, q)
joined_states: PROPOSITION
    Reachable(step_00, T)(q) AND
    Joined?(q`users(A0)) AND
    Joined?(q`leader(A0))
    => EXISTS Ka, Na: q`users(A0) =
      Joined(Na, Ka) AND q`leader(A0) =
      Joined(Na, Ka)

forward_secrecy : THEORY
  BEGIN
      fwd_secrecy: LEMMA
         FORALL m,T: Reachable(step_00, T)
                 AND inSet(intKnldg,m)
                     IMPLIES rank(m) > 0
  END forward_secrecy
```

Using the features of PVS, we have proved that the protocol satisfies forward secrecy property by establishing the correctness of the above theorem *rank theorem for forward secrecy Property* in PVS. The proof was conducted using the set of general requirements in addition to the protocol model, the implementation of the proof took around three months. The proof of backward secrecy can be derived in a similar fashion, and in much shorter time, given the experience gained.

6 Conclusion

The correctness of security protocols in communication systems remains a great challenge because of the sensitivity of the services provided. Formal methods have been used widely in this area to perform protocol verification and analysis. In this paper, we illustrated the need for a verification methodology for a class of protocols that deal with group key distribution. While most approaches in the literature target cryptographic properties for two parties protocols, the verification problem for group key distribution protocols is more challenging. In addition, properties like forward and backward secrecy are very important for protocols correctness, however, they did not receive enough attention in the literature.

In this paper, we provided a set of generic requirements of group key distribution protocols and established their formal specifications, then defined a formal model for this class of protocols, and finally presented rank theorems to enable and mechanize the verification procedure of this class of protocols. We defined our model, rank functions and rank theorems in PVS in order to construct the proof of the claimed security properties. We proved the soundness of our approach by proving the correctness of the rank theorem. We applied our methodology on the Enclaves group management protocol, and constructed the correctness proof for forward secrecy property for this protocol in PVS. We are in the process of extending the above approach to prove other properties for the same protocol, like backward secrecy property, authentication, and group consistency under protocol events.

References

[1] M. Archer. Proving Correctness of the Basic TESLA Multicast Stream Authentication Protocol with TAME. In *Workshop on Issues in the Theory of Security*, January 2002.

[2] G. Denker and J. Millen. Modeling Group Communication Protocols using Multiset Term Rewriting. In *Rewriting Logic and its Applications*, volume 71 of ENTCS. Elsevier, 2002.

[3] B. Dutertre, V. Crettaz, and V. Stavridou. Intrusion-tolerant enclaves. In *Proc. IEEE International Symposium on Security and Privacy*, pages 216–224, May 2002.

[4] B. Dutertre and S. Schneider. Using a PVS Embedding of CSP to Verify Authentication Protocols. In *Theorem Proving in Higher Order Logics*, volume 1275 of *LNCS*, pages 121–136. Springer-Verlag, 1997.

[5] C. Meadows and P. Syverson. Formalizing GDOI Group Key Management Requirements in NPATRL. In *Proc. ACM Conference on Computer and Communications Security*, pages 235–244, November 2001.

[6] C. Meadows, P. Syverson, and I. Cervesato. Formal Specification and Analysis of the Group Domain of Interpretation Protocol using NPATRL and the NRL Protocol Analyzer. *Journal of Computer Security*, 12(6):893–932, 2004.

[7] M.Layouni, J. Hooman, and S.Tahar. On the Correctness of an Intrusion-Tolerant Group Communication Protocol. In *Correct Hardware Design and Verification Methods*, volume 2860 of *LNCS*, pages 231–246. Springer-Verlag, 2003.

[8] S. Owre, J. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Proc. Intl. Conf. on Automated Deduction*, volume 607 of *LNCS*, pages 748–752. Springer Verlag, 1992.

[9] O. Pereira and J. Quisquater. Some Attacks upon Authenticated Group Key Agreement Protocols. *Journal of Computer Security*, 11(4):555–580, 2004.

[10] P. Ryan and S. Schneider. *The Modelling and Analysis of Security Protocols: The CSP Approach*. Addison-Wesley, 2001.

[11] G. Steel, A. Bundy, and M. Maidl. Attacking a Protocol for Group Key Agreement by Refuting Incorrect Inductive Conjectures. In *Proc. Intl. Joint Conf. on Automated Reasoning*, volume 3097 of *LNCS*, pages 137–151. Springer-Verlag, 2004.

[12] H. Sun and D. Lin. Dynamic Security Analysis of Group Key Agreement Protocol. *IEEE Transactions on Communication*, 152(2):134 – 137, April 2005.

[13] P. Syverson and C. Meadows. Formal Requirements for Key Distribution Protocols. In *EUROCRYPT*, volume 950 of *LNCS*, pages 320–331. Springer-Verlag, 1995.

[14] T. Truderung. Selecting Theories and Recursive Protocols. In *Proc. Concurrency Theory*, volume 3653 of *LNCS*, pages 217–232. Springer-Verlag, 2005.