Formal Specification Requirements for Group key Management and Distribution 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

July, 2006

Abstract

Design and verification of cryptographic protocols has been under investigation for quite long time. However, there is not much attention paid for the class of protocols that deals with group key management and distribution, mainly because of their dynamic characteristics. In this paper, we provide formal specification requirements for group key distribution and management and distribution protocols. This can help in the correct specification of the behavior of the protocol which is necessary in the design and verification process. We define a formal model based on these specifications and we use the idea of rank functions in order provide a solution for the verification problem of this class of protocols using theorem proving.

1 Introduction

Cryptographic protocols provide security services for communicating entities. The protocol involves a precise interaction between the entities in order to achieve the required service such as authentication of origin, establishing session keys between communicating entities, and secrecy of information. Because of the sensitivity of these services, it is very important to verify the operation of protocols and that they are not subtle to attacks. There are examples of protocols that have been used extensively before it turns out that an attack can be taken against them, even though the protocol received intensive analysis, but the flaw could not be addressed. This triggered people towards using formal methods to design and verify protocols. There are many approaches and tools that have been developed in this direction, some succeeded in identifying new flaws in protocols.

There are different kinds of environments that protocols must interoperate with, networks also 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 like availability and fairness take more important rules in some protocols like in commercial systems. This requires that complexity of the cryptographic protocol should be increased. In addition, new cryptographic primitives are being adopted; in group key management protocols of unbounded size, it is not obvious how to reason about them with existing protocol analysis systems. This of course, makes both verification and implementation more difficult. It also requires the search for new modeling and verification approaches for cryptographic protocols [14].

Group communication applications use encryption methods in order to limit access to information for legitimate members only. They are also dynamic with regard to principals participating in the group. So messages are protected by encryption using a chosen key, which in the context of group communication is called the group key. Only those who know the group key are able to recover the original message. In addition, the group may require that membership changes cause the group key to be refreshed. Changing the group key prevents a new member from decoding messages exchanged before they joined the group or after they leave. If a new key is distributed to the group when a new member joins, the new member cannot decipher previous messages even if it has recorded earlier messages encrypted with the old key.

However, distributing the group key to legitimate members is a complex problem. Although re-keying a group, by sending the new group key to the old group members encrypted with the old group key, before the join of a new member is trivial, re-keying the group after a member leaves is far more complicated. The old key cannot be used to distribute a new one, because the leaving member knows the old key. Therefore, a group key distributor must provide another scalable mechanism to distribute keys to the group.

So, there should be a mechanism to create secret keys and distribute them among legitimate principals that guarantees the secrecy of these keys. There are three different classes for group key management [16]: *Centralized group key management protocols*, where a single entity is responsible for controlling the whole group, hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server sides. *Decentralized architectures*, where the management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place. Finally, *distributed key management protocols*, where there is no explicit key distribution center, and the members themselves do the key generation. All members can perform access control and the generation of the key can be either contributory, meaning that all members contribute some information to generate the group key, or done by one of the members.

Talking about protocols involving two or three parties, the general requirements for these protocols are well understood, however, the case is different with group key distribution protocols, where the key can be distributed among large number of member who may join or leave the group at an arbitrary time. So security properties that are well defined in normal two parties protocols have different

meaning and different interpretation in group key distribution protocols, and so they require a more precise definition before we talk about how to verify them. An example of such properties is secrecy property, which deals with the fact that secret data should remain secret and not compromised. However, talking about group key distribution protocols, this property has a further dimension since there are long terms secret keys, short term secret keys, in addition to present, future, and past keys; where a principal who just joined the group and learned the present key should not be able to have enough information to deduce any previous keys, or similarly a principal who just left the group should not have enough information to deduct any future keys.

There are many protocols design approaches suggested to handle the .

There are many suggested protocols to handle the group key distribution problem, examples are Hydra [15], Enclaves [4], key graphs [15], and Internet Group Management Protocol (IGMP) [15]. Rafaeli and Hutchison [16] provided a survey on current approaches and protocols for key distribution. The correctness of these protocols is an important issue that has not been investigated enough yet. Therefore, in this paper we discuss our approach to provide a verification methodology for this class of protocols. We first give a formal specification requirements for group distribution protocols, and then discuss the steps of a verification methodology which is based on our requirements definition and utilizing previous techniques and existing verification tools.

We intend to use theorem proving techniques as verification method. So we will be concerned with the fact that certain messages should not occur, or should occur only under particular conditions. In theorem proving, this means providing theories for establishing the impossibility of particular combinations of events. In general, security properties are concerned with conditions under which the intruder can learn specific facts. We require that facts are obtained by protocol principals only be possible after some other fact (like authentication) has been occurred. In order to establish a prove that a fact is not available to the intruder, we need to show that this specific fact has a characterizing property that enables its generation, and the intruder do not have that property. To achieve this, we assign value or *rank* to each fact, these ranks will depend on the protocol itself, the initial knowledge and capabilities of the intruder, and finally the facts that we want to reveal to the intruder [17].

The rest of this paper is organized as follows, Section 2 provides related work to ours. In Section 3, we provide our formal specification requirements for group

key management protocols. In Section 4 we provide an application of the requirements to verify properties for group key management protocols. Finally Section 5 concludes the work with future work hints.

2 Related Work

In this section we discuss approaches for modeling and verification of group key management and distribution protocols that are closely related to our work. There is a limited effort has been done on modeling and verification protocols that involve more than two parties. In addition, there are very few trials to discuss the general formal requirements for reasoning about cryptographic protocols, which, once developed, can be applied at the design stage of new protocols as well as existing ones.

Syverson and Meadows [19] presented a the formal requirements for authentication in key distribution protocols. They tried to provide a single set of requirements to specify a whole class of protocols, which can be fine-tuned for the particular application. There were two main problems in their approach; first, the requirements they provide was for a single property, which is authentication, which is a property that has no different meanings in different protocols. Second, the requirements are defined as a security property, whereas the definition should include the whole protocol requirements and how they can be interpreted and then applied into a specific protocol.

The work in [9, 8] used a combination of three different approaches in order to prove the correctness of Intrusion-tolerant Enclaves protocol. They used Murphi model checker in order to verify authentication property, and PVS (Prototype Verification System) [12] to verify safety and liveness properties such as proper agreement, agreement termination, and integrity, finally, and Random Oracle model to manually prove robustness and unpredictability properties. The choice of the techniques was driven by the nature of the correctness arguments in each module of the protocol, by the environment assumptions and the capability of the verification approaches. This example shows that its difficult to verify and analyze this class of protocols. There was no single formalism where the protocol can fit and its verification is feasible. However, they achieved a promising success in verifying a complex protocol such as Enclaves, the results could be improved further first by using the rank functions to partition the message space and mechanically prove properties for the protocol [17]. Also by providing correctness

proofs for the consistency group membership when members leave and join the group. Finally one of the biggest challenges is to perform the analysis of the group key management module in PVS which requires the elaboration of some general purpose theories that deal with probabilities which are not yet available in theorem provers.

In addition to the previously mentioned work in [9], Meadows and Syverson [10] used the NPATRL language, a temporal requirement specification language for use with the NRL Protocol Analyzer, in order to specify the the Group Domain of Interpretation (GDOI) key management protocol. In a later stage Meadows *et al.* [11] 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. However, the problem with this approach is that there is no general set of requirements for protocols requirements which can be applied on specific protocol, or can be used for the refinement of protocol specifications in the design stage. In a related approach, Denker and Millen [3] used multiset term rewriting in order to model group communication protocols.

Bresson *et al.* [1, 2] presented a security model Group Diffie-Hellman protocols for Authenticated Key Exchange (AKE) and use it to precisely define AKE and the entity-authentication goal as well. They then define in this model the execution of an authenticated group Diffie-Hellman scheme and prove its security. Other work by Kim *et al.* [6, 7] investigates a novel group key agreement approach which blends key trees with Diffie-Hellman key exchange for the purpose of designing group key agreement protocols. But they do not provide any proof of the security of the protocol. A complementary step would be providing an approach to verify this class of protocols based on the same approach to design them.

In a recent work, Pereira [13] 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 suit. In a recent work, Sun and Lin [18] extended the strand space theory to analyze the dynamic security of GKAP.

There has been recent trials to provide design approaches for group key management protocols. Wong *et al.* [20] presented three strategies for securely distributing re-key messages after a join/leave operation and specified protocols for joining and leaving a secure group. Kikuchi [5] presented a new group key dis-

tribution scheme based on the Rabin public-key cryptosystem. Another approach based on tree-key for group key management protocols construction is found in [7]. Rafaeli and Hutchison [15] presented a decentralized architecture to create and distribute symmetric cryptographic keys to large multicast-based groups. These design approaches are important to us because the formal model we intend to present should be scalable to different protocols and therefore should consider the design approaches used to develop group protocols.

Ryan and Schneider [17] proposed the idea of rank functions for protocols verification by defining suitable rank functions for the CSP (Communication Sequential Process). In this project, we take advantage of this idea and use it for group key distribution protocols by applying the ranks functions on our protocol model and providing a mechanized verification approach.

We noticed that there are general common problems in many of these approaches discussed above. There is no single formalism to model the protocols and reason about their security properties. There is no formal link between the informal specification and the provided protocol models and their security properties. All approaches concentrate on the trivial secrecy and authentication properties. There are no trials to reason about complex features of key distribution properties such as key hierarchies which is not easy to reason about. There is no mechanized verification methodology to relay on to prove the correctness of a specific protocol. Finally, the design process of new protocols doesn't relay on a well defined specification requirements, which will reduce the possibility of introducing errors into the protocol in the design stage. This justifies the need of formal specification requirements of group key distribution protocols, which is discussed in the next section.

3 Formal Specification Requirements for Group Key Distribution Protocols

In this section we give specification requirement of group key management protocols. Since there are many different approaches in literature to design these protocols, specially keys generation and distribution, the specification of these protocols and their properties are informal. So we try to provide a common formal model for these specifications where most of these currently designed protocols fit. We need these formal specification requirements for many reasons: first, to fill the gap between the informal protocols descriptions on one hand and the formal

protocol models and their implementations on the other hand. Second, to integrate formal analysis in the design process of cryptographic protocols and specifically group key distribution protocols [10]. Finally, to give a better understanding for the verification problem and suggest a verification method based on these requirements.

Freshness requirement imposes that when a principal receives a piece of information, such as keys, then this information must have been fresh and currently valid. In this sense, freshness is similar to those that have been defined for two parties protocols. Group secrecy should guarantee that its computationally infeasible to discover any group key. Forward secrecy should guarantee that knowing a subset of old group keys will not lead to the computation of any subsequent group key. Backward secrecy should guarantee that knowing a contiguous subset group keys will not lead to the computation of a preceding group key.

Joining group or leaving group events are operations that result in creating a new group with a new group key out of an existing group. Any protocol should guarantee the above properties should apply on joining/leaving group members. Merge group event is the operation where portioned subgroups need to be merged back into a single group, a new group key is computed and distributed to every member of the new group, and group keys for subgroups are considered old or preceding keys. Split group event is the operation of creating two subgroups out of a single group, where every subgroup has an independent group key.

The formal definition of these protocols requirements is necessary first for understanding the protocol, and second for the protocol verification. Therefore, we are concerned with providing a formal definition for these requirements as a first step. We will consider different protocol design approaches, explore how these requirements are provided in these protocols, and formally model them accordingly.

We introduce the notation of time in order to be able to model specific events and properties in the protocols. This is mainly because the definition of a specific property at one time is different from another time, which can be a future time or a past time. This is necessary when we talk about forward secrecy and backward secrecy.

In order to provide a formal definition of these requirements, we introduce here the notation we use throughout the report.

P: honest principals who are willing to communicate.

I: a dishonest member.

G_t: current group, which cab be formally defined as a set of principals who share a secret key, or information that can be used to calculate the secret key.

G_{t+n}: a group that can share a secret key at future time.

G_{t-n}: a group that previously in time shared a secret key.

F: a set of facts, i.e. nounces, secret keys, etc.

E: the set of dynamic operations, i.e. join, leave, merge, and split.

τ a trace of events that can take place during a protocol run, like encryption, decryption, send, receive, etc.

K_{G_t}: the group session key: which is the key generated for the current session. Equivalently, it can be the set of information that can be used to calculate the key.

$K_{G_{t+n}}$: a group session key that can be generated and used sometime in the future.

$K_{G_{t-n}}$: a group session key that was generated and used previously sometime in time.

$\in$ group membership: we define membership as follows: if a principal P knows a key K_{G_t}, then $P \in G_t$.

3.1 Authentication

This property is similar to authentication in two parties protocol, and it has received intensive analysis and so formally modeled in many frameworks [10, 17, 19]. However, in group key distribution, it can be intended for authentication of group members to each others, in addition to the authentication of a new user to the group.

3.2 Secrecy

Only members of the group should have access to keys. the important issue here, is wither we want to allow users who just joint the group to have access to previously used keys, also wither we want to allow users who just left the group to have access to keys that will be generated hence after. To insure the secrecy of old and new keys, every protocol uses a mechanism for keys generation to guarantee that they cannot be calculated using the current group session information including the key itself.

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 G_t, and dishonest principal I who knows a set of facts F, there is no trace τ that he can execute in order to obtain the current group session key K_{G_t}.*

Definition 3.2. *Forward secrecy requires that a session key cannot be calculated from 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. In order for a protocol to satisfy this property, there should be no trace of events that can lead to generating a previously used keys by a user who was not part of the group at the time when the key was generated.

We formally model forward secrecy requirement as follows:

For any current group G_t, and a dishonest principal I who knows $K_{G_{t-n}}$, there is no trace τ that he can execute in order to obtain the current group session key K_{G_t}.

Definition 3.3. *Backward secrecy requires that a session key cannot be calculated from 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. The main concern here is that a user who decides to leave the group, should not be able to use the information he learned in order to calculate keys that maybe used after he left.

We formally model backward secrecy requirement as follows: For any current group G_t, and a dishonest principal I who knows K_{G_t}, there is no trace τ that he can execute in order to obtain a previous group session key $K_{G_{t-n}}$.

3.3 Freshness

This is the requirement that, if a principal receives information, then it must be current at some specified time. When a principal requests a piece of information, he expects to receive a current information. So when a protocol responds to a member's request with for a key, it must be sure that this is a new request, not a reply of an old one. Therefore, this freshness requirement can only be guaranteed for honest members, as we cannot prevent dishonest users from replying old requests to obtain some information.

Definition 3.4. *Freshness requirement: a principal P_i can execute a trace τ to obtain a fact F, however, there is no principal P_j that can execute the same trace tau in order to obtain the fact F.*

In order to reason about freshness, we have to consider time, and assign time-stamps to every piece of information, including the traces that can be executed by principals, and the set of facts the principal is willing to obtain.

3.4 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*. Leave occurs when a member wants to leave the group or forced to. Join occurs when a member wants to have access to current group. Although, protocols may impose an agreement criteria on joining and leaving groups, the effect of executing the event should result in a new group settings, in case the event is executed successfully, i.e. the member is granted access to the group, or released from it.

Definition 3.5. *A principal P joins the group if for a current group settings G_t, $P \notin G_t$, and there is a trace τ that P can execute such that $P \in G_{t+n}$ and $K_{G_{t+n}} \neq K_{G_t}$.*

For this definition of *join* event, there is a time delay of n, which should be less than the max join delay imposed by the protocol.

Definition 3.6. *A principal P leaves the group if for a current group settings G_t, $P \in G_t$, and there is a trace τ that P can execute such that $P \notin G_{t+n}$ and $K_{G_{t+n}} \neq K_{G_t}$.*

3.5 Merging and Splitting Groups

Merging and splitting groups are considered as multiple members operations, some protocols relay on distributing security management among distributed servers rather than one single server, this is obtained by having multiple groups. However, sometimes there is a need to merge two groups (or more) or to split a current group into two groups. These events affect the current groups settings and result in new settings that should maintain all the security requirements of the protocol.

Merge event occurs when two groups with two different settings execute a trace of events that result in every member of either group is a member of a new group that has new settings. Whereas split event occurs when one group executes a trace of events that result in two new different groups, where every member of current group is a member of one and only one of the new groups.

Definition 3.7. *A group $G1_t$ merges with group $G2_t$, if there is a trace τ that both $G1_t$ and $G2_t$ can execute such that $\forall\ P\ \in\ G1_t \cup G2_t\ P\ \in\ G_{t+n}$, $K_{G_{t+n}}\ \neq K_{G1_t}$ and $K_{G_{t+n}}\ \neq K_{G2_t}$.*

Definition 3.8. *A group G_t splits into groups $G1_t$ and $G2_t$, if there is a trace τ that G_t can execute such that $\forall\ P\ \in\ G_t\ either\ P\ \in\ G1_{t+n}\ or\ P\ \in\ G2_{t+n}$, $K_{G_t}\ \neq K_{G1_{t+n}}$ and $K_{G_t}\ \neq K_{G2_{t+n}}$.*

These definitions can are used in order to define rank theorems for the properties we wish to verify.

4 Verification Approach Based on the Formal Requirements

Our verification approach is based on the specification requirements model provided in the previous section. The main idea is to map the requirements into ranks, this map is based on a predefined function called *the rank function* [17]. Then, we will define rank theorems that provide conditions satisfied by this rank function in order to conclude that the model satisfies the security property under investigation. The proof establishment will be mechanized in PVS theorem proving based on embedding the rank functions into PVS and establishing proof of correctness for the rank theorems in PVS.

We will define a verification approach based on the model we presented, this approach should have the following features:

- It can prove the correctness of the protocol under investigation.
- It can verify security properties provided by the protocol, mainly safe key distribution.

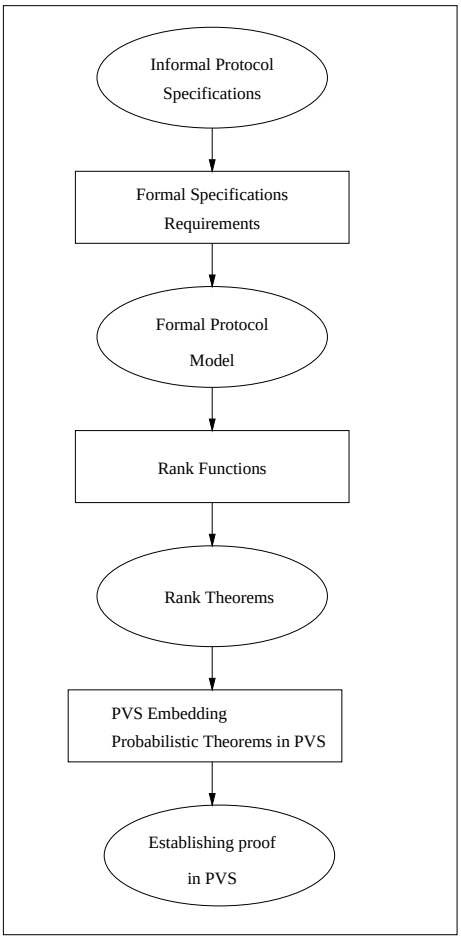

Figure 1: Verification Methodology

- It can be efficiently implemented, which means there is an appropriate tool support.

In our approach, we can consider establishing the proof on the different levels of abstraction of the protocol at two levels: the protocol level and the encryption level. Working at the protocol level, embedding of the rank functions and rank theorems into PVS will make verification feasible, however, working out the proof at the encryption level, will require the definition of probabilistic theorems in PVS which are not available yet.

Figure 1 shows the steps of our verification methodology. The first step, which was discussed in the previous section, will help to eliminate the gap between the informal protocol specification and the formal model and provide a well defined protocol specification that can be directly integrated into the verification methodology. In the second step, we define a map function between the set of facts we

know, or we can learn about the protocol and the set of integers. This mapping function will be useful in partitioning the message space and enabling mechanized proof of security protocols properties. After this map, we obtain *rank theorems*, which are the set of properties and protocol specifications modeled using the rank functions we defined. In order to prove the correctness of a specific property, we need to show that its corresponding rank theorem is correct with regard to the protocol model. The final step, is to mechanize the proof through PVS theorem prover.

4.1 Rank Functions for Protocol Models

Rank functions were first introduced in [17] for the purpose of proving CSP using PVS. 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 we wish to show cannot be obtained by the intruder cannot have positive rank. The ranks that are assigned will depend on the protocol itself, the initial knowledge and capabilities of the intruder, and the property we want to prove. In our approach, we will define suitable rank functions that map our formal specification requirements in order to obtain rank theorems, which are the properties we wish to prove, the key result that provides the basis of the verification approach is that if these requirements all hold, then no fact of non-positive rank can be generated by the system. Which means that these facts cannot be leaked to dishonest users.

The definition of the rank function is provided in [17] as follows:

Definition 4.1. *(Rank Function) A rank function ρ is a map function $\rho : Fact \cup Signal \rightarrow \mathbb{Z}$ which maps facts and signals into integers.*

In order to ensure that facts and signals of positive rank can be generated, it is necessary to verify that each participant cannot introduce anything of non-positive rank to the system. In other words, intruder initial knowledge must be of positive rank, and only facts of positive ranks can be generated from sets of facts of positive rank. We illustrate this idea by defining a rank function for group secrecy requirement defined in the previous section.

$$\rho(f) = 1, \text{where } f \in F$$

$$\rho(key) = \begin{cases} 0 & if\ key = K_{G_t} \\ 1 & otherwise \end{cases}$$

$$\rho(m) = \begin{cases} 0 & if\ \{m\} = K_{G_t} \\ 1 & otherwise \end{cases}$$

$$\rho(m1.m2) = min\{\rho(m1), \rho(m2)\}$$

$$\rho(claim\_secret.k) = \begin{cases} 0 & if\ k = K_{G_t} \\ 1 & otherwise \end{cases}$$

The rank theorem in the above map is $\rho(K_{G_t}) > 1$ which we wish to proof correct. All previous maps define facts about the protocol in general and maps them into integer values. However, for a specific protocol, the rank function maybe a little bit different depending on the nature of the protocol. In addition, we can define similar rank functions for all the properties we defined in the previous section.

So, to define the property we want to proof for a specific protocol, like $\rho(K_{G_t}) > 1$, we say: For all events ε and trace tau a principal P can execute on G_t, $\rho(K_{G_t}) > 1$. $\forall\ \varepsilon, \tau\ P \parallel G_t \Rightarrow \rho(K_{G_t}) > 1$. Where, $\varepsilon \in E$ and $\parallel$ represents executing the protocol by a principal.

The rank function ρ can also be appliedd on forward and back ward secrecy requirments. The rank function for forawrd secrecy is defined as:

$$\rho(key) = \begin{cases} 0 & if\ key = K_{G_t} or\ key = K_{G_{t-n}} \\ 1 & if\ key = K_{G_{t+n}} \end{cases}$$

and the rank function that maps backward secrecy is defined as follows:

$$\rho(key) = \begin{cases} 0 & if\ key = K_{G_t} or\ key = K_{G_{t+n}} \\ 1 & if\ key = K_{G_{t-n}} \end{cases}$$

Similarily, we want to show for both forward and backward secrecy that $\rho(K_{G_t}) > 1$ holds:

$\forall\, \varepsilon, \tau\ P \parallel G_t \ \Rightarrow\ \rho(K_{G_t}) > 1$

4.2 Verification in Theorem Proving

In the next step, we plan to mechanize the correctness proof using the general purpose theorem prover PVS. For our case, we need to provide a PVS embedding for the formal requirements, the rank functions we defined, and the rank theorems. In addition, we need to consider timing since security properties we deal with are time dependent (such as forward and backward secrecy), and protocols messages may contain time-stamps as part of the message. For this purpose, we will use the general PVS theory about time automata developed by Layouni [8].

In a later stage of this project, if we plan to conduct correctness proofs for the protocols at the encryption level, we will need to define probabilistic theories in PVS in order to be able to reason about encryption and decryption algorithms. However, this remains a future direction of this project which depends on the results we can achieve using the current approach in addition to the time needed.

5 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 our proposal, we provided a literature survey of the state of the art approaches to use formal methods in the design and verification of cryptographic protocols. We also illustrated the need for a verification methodology for a class of protocols that deal with group key distribution, because most of the approaches in the literature targets cryptographic properties for two parties protocols, while the verification problem is more challenging for group key distribution protocols.

The contributions of this paper are providing a state of the art survey on modeling and verification of group key distribution protocols, then the formal specification requirements of group key distribution protocols, and finally defining rank functions for our requirements and proposing a verification approach to enable and mechanize the verification procedure of this class of protocols. We use rank functions that map the formal protocol model into a set of integers in order to obtain rank theorems. As future work, we will embed our model, rank functions

and rank theorems in PVS in order to be able to construct the proof of the claimed security properties. This requires providing PVS model for the protocol under investigation.

References

[1] E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange - the Dynamic Case. In *Proc. of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, Lecture Notes in Computer Science, pages 290–309. Springer-Verlag, 2001.

[2] E. Bresson, O. Chevassut, and D. Pointcheval. The Group Diffie-Hellman Problems. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, page 325338. Springer-Verlag, August 2002.

[3] G. Denker and J. Millen. Modeling Group Communication Protocols using Multiset Term Rewriting. In *Rewriting Logic and its Applications*, volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002.

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

[5] H. Kikuchi. Rabin Tree and its Application to Group Key Distribution. In Farn Wang, editor, *Automated Technology for Verification and Analysis, ATVA 2004*, volume 3299 of *Lecture Notes in Computer Science*, pages 384–391. Springer-Verlag, November 2004.

[6] Y. Kim, A. Perrig, and G. Tsudik. Communication-Efficient Group Key Agreement. In *IFIP/Sec '01: Proceedings of the IFIP TC11 Sixteenth Annual Working Conference on Information Security*, pages 229–244. Kluwer Academic Publisher, 2001.

[7] Y. Kim, A. Perrig, and G. Tsudik. Tree-based Group Key Agreement. *ACM Transactions on Information and Systems Security*, 7(1):60–96, 2004.

[8] M. Layouni. On the Formal Verification of an Intrusion-Tolerant Group Communications Protocol. Master's thesis, Concordia University, Montreal, Canada, September 2003.

[9] 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 *Lecture Notes in Computer Science*, pages 231–246. Springer-Verlag, 2003.

[10] 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.

[11] 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.

[12] S. Owre, J.M. Rushby, and N. Shankar. PVS: A Prototype Verification System. In *Automated Deduction*, volume 607 of *Lecture Notes in Computer Science*, pages 748–752. Springer Verlag, 1992.

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

[14] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proc. IEEE Symposium on Security and Prvacy*, pages 56–73, Washington, DC, USA, May 2000.

[15] S. Rafaeli and D. Hutchison. Hydra: A Decentralised Group Key Management. In *Proceedings of the 11th IEEE International Workshops on Enabling Technologies*, pages 62–67. IEEE Computer Society Press, 2002.

[16] S. Rafaeli and D. Hutchison. A Survey of Key Management for Secure Group Communication. *ACM Computing Surveys*, 35(3):309–329, 2003.

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

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

[19] P. Syverson and C. Meadows. Formal Requirements for Key Distribution Protocols. In *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 320–331. Springer-Verlag, 1995.

[20] C. Wong, M. Gouda, and S. Lam. Secure Group Communications using Key Graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, 2000.