Theorem Proving based Framework for Verification of Group Key Protocols

Amjad Gawanmeh

Department of Electrical and Computer Engineering,
Concordia University, Montreal, Quebec, H3G 1M8 Canada
Email:amjad@ece.concordia.ca

Abstract. The correctness of group key protocols in communication systems remains a great challenge because of dynamic characteristics of group key construction as we deal with open number of group members. In this paper, we present a combination of three different theorem-proving methods to verify security properties for group oriented protocols. In the first method, rank theorems for forward properties are established 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 deduced from a set of rank functions we define over the protocol. In the second, we provide a sound and complete inference system to detect attacks in group key management protocols. The inference system provides an elegant and natural proof strategy for such protocols compared to existing approaches. Finally, in the third method, we use an event-B first-order proving system to provide invariant checking for group key secrecy property. In our framework, we applied each method on a different group protocol from the literature illustrating the features of each method.

1 Introduction

Security protocols are used to establish secure channels between communicating systems. Great care needs to be taken in developing and implementing robust protocols. The complexity of security-protocol interactions can hide security weaknesses that normal analysis methods cannot reveal, mainly because 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.

There are different kinds of environments that protocols must interoperate with, 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. This of course, makes both modeling and verification more difficult. It also requires the search for new modeling and verification approaches for cryptographic protocols.

Group key protocols allow group members to exchange or establish keys to encrypt and authenticate messages within the group. At the same time, individuals outside of the group cannot eavesdrop on group communication or inject messages. In fact, group key management protocols need security retention in the case of dynamic member actions, such as leaving the group for an existing member, or joining the group for a new member. It should be also guaranteed that all authorized members are able to access the group, at the same time unauthorized ones are unable to have this access.

Therefore, security properties that are well defined in normal two-party protocols have different meanings and different interpretations in group key distribution protocols. An example of such properties is secrecy, which deals with the fact that secret data should remain secret and not compromised. However, for group key distribution protocols, this property has a further dimension since there are long-term 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. 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 these protocols.

In this paper, we describe a framework for the verification of group key management protocols. The framework is based on three approaches: a rank theorems based approach, a rank functions based inference system, and and event-B first-order logic theorem proving approach. In the first two methods we adapted the idea of rank functions introduced by Schneider *et al.* [12, 21] in order to be able to verify security properties for group oriented protocols. In our method, we first define a set of sound rank functions that satisfy specific requirements, and prove the correctness of every rank function with these requirement. Then, we use this set of rank functions in two directions, in the first to define rank theorems that guarantee the correctness of the security property. The implementation of the rank theorems methods, from our experiences, requires a considerable amount of effort and time in theorem proving. To enhance the performance of this technique we combine the rank function into the protocol events. This resulted in the second method, a rank functions based inference system, which requires the same level of higher-order logics to reason about messages and events. The inference system is composed of a set of inference rules over rank functions, where, every rule can be applied in order to generate new knowledge and assign new ranks to these generated messages. A special rule called *Attack* is defined to represent the bottom of the system, and, when executed, illustrates an attack in the protocol. We formally prove the soundness and the completeness of the rank theorem and the inference system for a sound rank functions.

Implementing the inference system in higher-order logic theorem proving required a lot of effort and time, in addition, verifying properties is achieved interactively with the theorem proving tool because of the decidability problem on higher-order logic. In the third method, which is complimentary to the above two, we define a formal link between event-B semantic and security protocols, the use of event-B first-order logic prover to verify secrecy property as an event-B invariant. This allows us to avoid the user interaction with the theorem proving tool, and reduce the time required to verify the security property. This is not a straightforward task, and should be based on a correct semantical link between the two models, and finally the application of a group protocol (that has join/leave events) making use of first-order logic automatic prover, while abstracting some features of the protocol.

The above methods were applied on different protocols from the literature. First we implemented the rank theorem proof environment on the Enclaves protocol from SRI [11] in order to verify related forward secrecy using the PVS (Prototype Verification System) theorem prover [18]. Then we implemented the verification of a generic Group Diffie-Hellman (GDH) protocol [24, 10] in PVS based on the proposed inference system approach. Finally, we applied the event-B based approach on a tree based Group Diffie-Hellman (TGDH) protocol and used invariant checking to verify the correctness of key construction. Protocol events and traces are modeled as event-B operations and messages as sets. Then, group secrecy property was defined and verified as an event-B invariant in tool Click'n'Prove [3].

The rest of the paper is organized as follows. Section 2 discusses related work to ours. In Section 3, we overview the general framework and preliminary definitions and notations we use. In Section 4, we define the concept of rank theorems, and we show how to define theorems for forward and backward secrecy properties. In Section 5, we introduce the details of our rank functions based inference system. In Section 6, we present the event-B based approach. Finally, Section 7 concludes the paper with future work hints.

2 Related Work

In this section we discuss work related to ours in the literature to the best of our knowledge.

Pereira and Quisquater [19] proposed a systematic approach to analyze protocol suites extending the Diffie-Hellman key-exchange scheme to a group setting. They 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 more recent work, Pereira and Quisquater [20] provided a systematic way to derive an attack against any Authenticated GDH-type (A-GDH) protocol with at least four participants and exhibit protocols with two and three participants. They provided a generic insecurity results concerning authentication protocols. In their work, the authors did not attempt to address group related properties such as forward and backward secrecy.

In another related work, Steel *et al.* [23] modeled a group key protocol by posing inductive conjectures about the trace of messages exchanged in order to investigate novel properties of the protocol, such as tolerance to disruption, 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 [26] presented a formalism, called selecting theories, which extends the standard non-recursive term rewriting model and allows participants to compare and store arbitrary messages. This formalism can model recursive protocols, where participants, in each protocol step, are able to send a number of messages unbounded w.r.t. size of the protocol. This modeling, however, cannot be applied on non–recursive protocols such as GDH or the Enclaves. In addition, the model provided is not readable and very complex to construct. There are many other efforts in the literature that deal with the formal analysis for GDH style protocols, for more details see [14].

Dutertre and Schneider [12] used an embedding of CSP (Communication Sequential Process) in PVS in order to verify the authentication property of the Needham-Shroeder public key protocol. They proposed the idea of rank functions in order to enable CSP verification of the Needham Shroeder protocol. Later, Ryan and Schneider [21, 22] used the idea of rank functions for the verification of CSP. The work did not present a method that can be applied on security properties in other classes of protocols, specifically, group key protocols. In fact, the method, as is, may not be applied on secrecy property for group key management protocols. Even though rank functions were introduced and used by Schneider *et al.* [12, 22, 21, 8, 9] in different directions, in this paper, we use their definition in order to precisely define a set of sound rank functions and prove their correctness. We then propose a rank functions based inference system for the verification of group key distribution protocols and prove the soundness and completeness of the system using the defined set of rank functions.

Layouni *et al.* [17] used a combination of model checking, theorem proving, and a Random Oracle Model to verify authentication property, safety and liveness properties such as proper agreement, and robustness and unpredictability properties, respectively. The verified protocol is a complex protocol developed for group key agreement under multiple leaders scheme, and is called the Enclaves protocol [11]. This example shows how difficult it is to verify and analyze this class of protocols. While the authors achieved a promising success in verifying a complex protocol such as Enclaves, they failed to accomplish the formal proof of the three components in a single formalism.

Events-based verification of security protocols was used by Crazzolara and Winskel [6, 7] for mappings between process algebra, Petri nets, strand spaces and inductive models. The authors established precise relationships between the Petri nets semantics and transition semantics, strand spaces, inductive rules, and trace languages and event structures. They show how event-based models can be structured in a compositional way and so used to give a formal semantics to security protocols which support proofs of the correctness of these protocols. They demonstrated the usefulness of their Petri nets semantics in deriving proof principles for security protocols and apply them to prove an authentication property. The method is applicable to authentication property insufficient for capturing forward and backward secrecy properties.

Stouls and Potet [25] proposed a method to automatically enforce an abstract security policy on a network. They used the B refinement process to build a formal link between concrete and abstract terms, which is dynamically computed from the environment data. They applied their method on a case study modeling a network monitor. A different approach to achieve a similar objective was proposed in [4], where the authors addressed the proof-based development of system models satisfying a security policy. They used OrBAC [1] models to express the security policies in order to state permissions and prohibitions on actions. An abstract B model [2] is derived from the OrBAC specification of the security policy and then the model is refined to introduce properties that can be expressed in OrBAC. The refinement guarantees that the resulting B model satisfies the security policy.

In this paper we address security property for group oriented protocols, taking into account features such as the concept of group secrecy and dynamic group events. In addition, we consider events that are specific for group protocols that were not treated by the B method based work of Butler [5]. The verification is tackled using theorem proving in higher-order and first-order logic models.

3 Verification Methodology

Figure 1 displays a general view of the proposed verification methodology. Three major techniques are used: in the first one, we use rank theorems to define the required security property based on the concept of rank functions. Rank theorem are implemented in PVS higher-order logic theorem prover. In the second approach, we provide a sound and complete inference system to detect attacks in group key management protocols. The inference system is also implemented in PVS. Finally, abstract our group model to fit it to the first-order logic theorem proving in event-B. We define a well-formed formal link between the group protocol model and the event-B counterpart model. Based on this link, we propose a solution to verify the required security properties, in particular secrecy properties, using the event-B invariant checking tool Click'n'Prove. First we present our formal model and the notations that will be used throughout this paper.

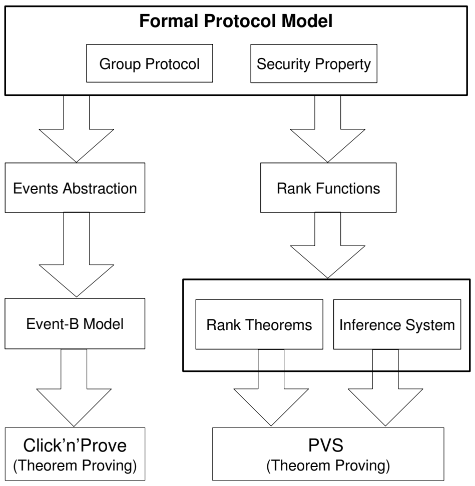

Fig. 1. Verification Methodology

$\mathbb{M}$: set of all possible messages (messages space).
P: a honest principal who is willing to communicate.
$\mathbb{P}$: set of knowledge of member P, $\mathbb{P} \subseteq \mathbb{M}$.
$\mathbb{S}$: secret messages space, the set of all secret messages, $\mathbb{S} \subset \mathbb{M}$. These are the messages we want to keep hidden from the intruder. They 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 unlimited resources and computational power. However, we state normal assumptions about the intruder such as being able to encrypt or decrypt a message only if he knows the appropriate key, or the ability to block or read any message in the system.
$\mathbb{E}$: 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, $\mathbb{E} : \mathbb{M} \rightarrow \mathbb{M}$. It represents an action the user can perform in order to obtain extra information and update his own set of knowledge.
$\mathbb{T}$: 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 $\tau : \mathbb{E} \times \mathbb{M} \rightarrow \mathbb{M}$, $m \in \mathbb{M}$, then we write $m = \tau(\mathbb{E}, \mathbb{M})$ to say that a message m is generated by the trace τ by executing the vector of events E on the set of messages M, we also write $\tau(\mathbb{E}, \mathbb{M}) \rightsquigarrow m$ to represent a predict 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 executing the protocol events. This information is usually public and known, so there are no secret information that is in the intruders initial set of knowledge. In other words $\forall m \in \mathbb{M} : m \in \mathbb{S} \Rightarrow m \notin \mathbb{K}_0$
$\mathbb{K}$: set of knowledge of the intruder. The intruder updates this knowledge by executing events. The intruder starts with the initial set of knowledge and the set of events, then, by executing a sequence of events, he/she updates this set. $\mathbb{K}_0 \subseteq \mathbb{K}$ and $\mathbb{K} \subseteq \mathbb{M}$.
attack: we define attack w.r.t. confidentiality as the ability of the intruder to have a message in the set of secret messages in his own set of knowledge, $attack \equiv m \in \mathbb{K}$ and $m \in \mathbb{S}$. The notion of attack can be seen differently depending on the nature of the security property under investigation.

Traces are defined since they will be used to prove the soundness of the inference system. t represents a single execution of one event or inference rule that updates the intruder set of knowledge. For a given events $e_1, e_2, ..., e_n \in \mathbb{E}$, we use $\mathbb{M}$ to represent the messages generated by executing each event: $m_1 = e_1(\mathbb{M}_0), m_2 = e_2(\mathbb{M}), ..., m_n =$

$e_n(\mathbb{M})$, where $\mathbb{M}_0 \in \mathbb{K}_0$ is a set of messages in the initial set of knowledge of the intruder, and $M^p \in \mathbb{K}_0$ is a vector of p messages in the updated set of knowledge of the intruder. Now we can define $t_1, t_2, ..., t_n$ as follows :

$t_1 : m_1 = e_1(\mathbb{K}_0),\ \rho(m_1) = c_1,\ \mathbb{K}_1 = \mathbb{K}_0 \cup \{m_1\}$
$t_2 : m_2 = e_2(\mathbb{K}_1),\ \rho(m_2) = c_2,\ \mathbb{K}_2 = \mathbb{K}_1 \cup \{m_2\}$
$t_i :\ m_i = e_i(\mathbb{K}_i),\ \rho(m_i) = c_i,\ \mathbb{K}_i = \mathbb{K}_{i-1} \cup \{m_i\}$
$\vdots$
$t_n : m_n = e_n(\mathbb{K}_n),\ \rho(m_n) = c_n,\ \mathbb{K} = \mathbb{K} \cup \{m_n\}$

We define a trace $T_n \in \mathbb{T}$ as the sequence of executing $t_1, t_2, ..., t_n$ in order.

$T_n : t_1, t_2, ...t_n;\ \mathbb{K} = \mathbb{K}_0 \cup \{m_1, m_2, ..., m_n\};\ \rho(m_n) = c_n$. We say that $m_n = T_n(\mathbb{E}, \mathbb{M})$ which means that the trace T_n generates the message m_n of rank c_n.

Rank functions were first introduced in [21]. 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 1. *A rank function ρ is a function $\rho\ :\ \mathbb{M}\ \rightarrow\ \mathbb{Z}$ that maps the set of all messages into integers.*

It is necessary to verify that protocol participants cannot generate non-positive ranks. The appropriate rank function we choose to apply on the protocol should be sound. We define a set of rank functions with a number of requirements, which will be used to prove the correctness of the rank function.

Definition 2. *a rank function is **sound** if it satisfies the following condition: applying an event on the intruder's initial set of knowledge will not generate a zero rank:*
$\forall m \in \mathbb{K}_0, e\ \in \mathbb{E}, \rho(m) > 0$ *and* $\rho(e(m)) > 0$.

This intuitively means that the rank function will not generate or add any inconsistency to the group model. Therefore, it will guarantee that zero ranks can only be generated as a result of problems in the protocol.

Definition 3. *a rank function is **initially sound** if it satisfies these three requirements:*

1. $\forall m \in \mathbb{M},\ \rho(m) >= 0$, there are no negative ranks generated by the system.
2. $\forall m \in \mathbb{K}_0,\ \rho(m) > 0$, intruder initial knowledge must be of positive rank.
3. $\forall m \in \mathbb{S},\ \rho(m) = 0$, all secret messages must have a zero rank.

Definition 4. *Two events are **invertible** if each one is the inverse of the other.*
$e_2(e_1(m_1)) = m_1$, *where* $e_1, e_2 \in \mathbb{E}$, *and* $m_1 \in \mathbb{M}$.

Definition 5. *a rank function is **invertible** for all invertible events of inference rules.* $e_2(e_1(m_1)) = m_1 \Rightarrow\ \rho(e_2(e_1(m_1))) = \rho(m_1)$, *where* $e_1, e_2 \in \mathbb{E}$, *and* $m_1, m_2 \in \mathbb{M}$, *and any user of the system cannot apply an invertible event unless he/she is able to apply the inverse.*

Definition 6. *a rank function is **bounded** if* $\rho(m) - 1 \leq \rho(e(m)) \leq \rho(m) + 1$, *where* $e \in \mathbb{E}$, *and* $m \in \mathbb{M}$.

Theorem 1. *Rank Function Soundness*
*A rank function is **sound**, if it is initially sound, invertible and bounded.*

The theorem states that a rank function with the above specifications is consistent. It ensures that the zero rank cannot be generated by the initial knowledge of the intruder, or by the definition of the rank function for the events. In other words, applying each single event separately on the set of intruders initial knowledge will not generate a zero rank, simply because a secret is not revealed to the intruder. Formally, $\forall m \in \mathbb{K}_0,\ \rho(m) > 0$ and $\forall m \in \mathbb{K}_0, e \in \mathbb{E},\ \rho(e(m)) > 0$. We use $\mathbb{R}$ to represent the set of all sound rank functions. More details about the theorem and its proof can be found in [14]. In the next sections we talk in details about each of the three methods of the framework.

4 Rank Theorems

We briefly present the steps of our rank theorems based verification methodology as described Figure 2. 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.

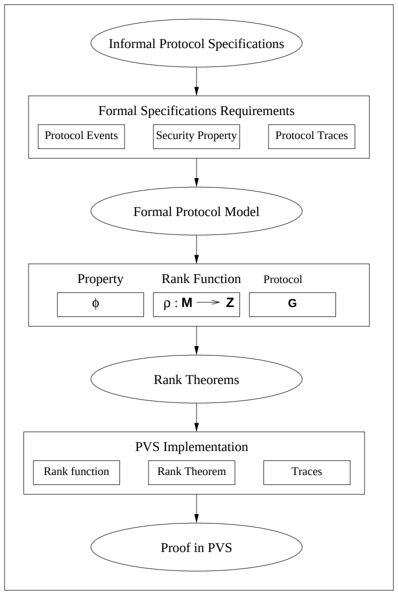

Fig. 2. Rank Theorem Based Approach

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.

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.

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 $t - i \geq 0$.

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

This means that for the validity of forward secrecy, we give rank zero to all messages in the set of secret messages $\mathbb{S}$, such as secrets shared between users and servers, and all 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* $\tau \in \mathbb{T}$.

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.

This approach was illustrated on forward secrecy property for the Enclaves protocol. For further details see [15]

5 Rank Functions based Inference System

Our inference system consists of a set of inference rules. Every rule represents an event in the protocol. Rules have a precondition that has to be satisfied before they are applied. We define the pair $\langle m, c \rangle$ to represent a message m and its rank c. A special rule, *Attack*, is defined with a precondition, such that, when executed by the intruder, it indicates the occurrence of an attack by reaching the bottom of the system $\perp$. The intruder, by executing these rules on the set of knowledge $\mathbb{K}$, generates new knowledge with new ranks and updates his/her set. For this system to work, we assume the fairness of executing these rules, i.e., the intruder will not keep using the rule *compose* forever, but other rules will have their chance to be executed, specially the rule *Attack*. The set of rules in the inference system are shown below.

Rule1: *Encryption:* $\frac{\mathbb{K}\cup\{\langle m,\rho_1\rangle,\langle k,\rho_2\rangle\}}{\mathbb{K}\cup\{\langle m,\rho_1\rangle,\langle k,\rho_2\rangle\}\cup\{\langle \{m\}_k,\rho_1+1\rangle\}}$

where $\{m\}_k = Encr(m,k)$

Rule2*: Decryption:* $\frac{\mathbb{K}\cup\{\langle Encr(m,k),\rho_1\rangle,\langle k,\rho_2\rangle\}}{\mathbb{K}\cup\{\langle \{m\}_k,\rho_1\rangle,\langle k,\rho_2\rangle\}\cup\{\langle m,\rho_1-1\rangle\}}$

where $m = Decr(\{m\}_k, k)$

Rule3*: Compose:* $\frac{\mathbb{K}\cup\{\langle m_1,\rho_1\rangle,\langle m_2,\rho_2\rangle\}}{\mathbb{K}\cup\{\langle m_1,\rho_1\rangle,\langle m_2,\rho_2\rangle\}\cup\{\langle Comp(m_1,m_2),min(\rho_1,\rho_2)\rangle\}}$

Rule4*: Decompose:* $\frac{\mathbb{K}\cup\{\langle Comp(m_1,m_2),\rho_1\rangle\}}{\mathbb{K}\cup\{\langle Comp(m_1,m_2),\rho_1\rangle\}\cup\{\langle m_1,\rho_1\rangle,\langle m_2,\rho_1\rangle\}}$

Rule5*: Expo:* $\frac{\mathbb{K}\cup\{\langle expo(m_1,m_2),\rho_1\rangle\}}{\mathbb{K}\cup\{\langle Comp(m_1,m_2),\rho_1\rangle\}\cup\{\langle {m_1}^{m_2},\rho_1-1\rangle\}}$

Rule6*: Attack:* $\frac{\mathbb{K}\cup\{\langle m,0\rangle\}}{\bot}$

For this inference system, we use the event $Enc(m,k)$ to represent a message m that is encrypted w.r.t. a symmetric encryption algorithm with the key k. The event $Dec(Enc(m,k),k)$ represents decrypting a message that is already encrypted, where the same key used for the encryption is to be used for the decryption event. These two events are $invertible$, therefore $m = Dec(Enc(m,k),k)$. The event $Comp(m1,m2)$ represents two composed messages by concatenation. The function min gives the minimum rank from two given ranks.

In the following we define theorems for the soundness and completeness of the approach. The first theorem states that if we can find a message in the set of knowledge of the intruder that has the rank zero, or equivalently, if the rule *attack* of the inference system is applied, then there is an *attack* in the system. We assume *fairness* in applying the inference rules.

Theorem 2. *Soundness*

Let $\mathbb{P}$ be a security protocol, let ρ be a sound rank function, and let $\mathbb{K}_0$ be the set of the initial knowledge of the intruder. Then, the protocol $\mathbb{P}$ has an attack if the inference rule $attack$ can be applied in a fair inference system.

$\exists m \in \mathbb{K} : \rho(m) = 0$ *and* $\rho \in \mathbb{R} \Rightarrow \exists attack$ *in* $\mathbb{P}$.

The following corollary is deduced from the above theorem and states that if there is no attack in the system, then a sound rank function will be greater than zero. It can be viewed as the complementary case of the above theorem.

Corollary 1. *Absence of Attack*

Assuming the same conditions as Theorem 2, if the protocol $\mathbb{P}$ has no attack, then the rule $attack$ will never be applied in a fair inference system.

$\nexists attack$ *in* $\mathbb{P} \Rightarrow \forall m \in \mathbb{K} : \rho(m) > 0$.

The second theorem states that if there is no message in the set of knowledge of the intruder that has the rank zero, or equivalently, if the rule *attack* of the inference system can never be applied, assuming *fairness* of the strategy application of inference rules, then there is no *attack* in the system.

Theorem 3. *Completeness*

Assuming the same conditions as Theorem 2, if the rule $attack$ cannot be applied in a fair inference system, then the protocol $\mathbb{P}$ has no attack.

$\forall m \in \mathbb{K} : \rho(m) > 0,\ \rho \in \mathbb{R} \Rightarrow \nexists attack$ *in* $\mathbb{P}$.

Corollary 2. *Detecting Attacks*

Assuming the same conditions as Theorem 2, if the rule $attack$ can be applied in a fair inference system, then the protocol $\mathbb{P}$ has an attack.

$\exists m \in \mathbb{K} : \rho(m) = 0$ *and* $\rho \in \mathbb{R} \Rightarrow \exists attack$ *in* $\mathbb{P}$.

This corollary states that when the rank function evaluates to zero, then there exists an attack in the protocol. Theorems 2 and 3 and Corollaries 1 and 2 provide the formal link between the protocol model and the implementation model in PVS. The proofs of these theorems can be found in [14].

The inference system we defined can prove that an attack exists in the protocol using the Theorem 2. However, the limitation of the Soundness Theorem comes in the type of implementation that will be used. In case we want to prove that there is no attack in the protocol, the inference system can diverge in case we apply the Theorem 3. The inference system may terminate with a result, depending on the nature of the protocol and the strategies used while conducting theorem proving.

We still can reason about absence of attacks in protocols using Theorem 3. An *indirect* proof can be generated by proving that the strategy is fair and the application of our inference system diverges. This *indirect* proof can be achieved by generating partial proofs that affirm that the inference system will diverge when the applied strategy is fair. We believe that in order to be apply this approach, we have to provide an implementation for the inference system that allows partial proofs based on divergence. This later issue will be considered for further study.

In order to illustrate this method, we consider the Group Diffie-Hellman (GDH) protocol [24], which is a basic group key management protocol widely studied in the literature. In the first part, we showed how to manually detect the attack in a step by step application of the inference system. Then we use the PVS theorem prover in order to implement the inference system and apply it on the protocol.

We applied our approach on the example protocol in the existence of an active adversary. The case of a passive adversary is more restricted than an active one. In addition, it has been shown that a protocol that is secure in the passive setting can be considered secure in the active case [16]. Therefore, we believe it is adequate to handle the active adversary case for the case study, and consider the passive adversary case as restricted special case which may be considered for further investigations [14].

6 Event-B Semantics based Verification Methodology

It is a practical solution to verify a security property using model checking tools, when applicable. However, it is inconvenient because of two reasons: the state space explosion problem of model checking, and the limited expressiveness of proposition logic based-tools. Treating the problem at the first-order logic level requires applying a valid abstraction on the protocol in order to fit to the proving system. This abstraction should be based on a correct semantical link between the protocol model and the target model. We tackle this problem by using event-B as the target first-order logic model benefiting from the automation and the expressiveness of the logic and the availability of supporting tools.

In order to model a group protocol in event-B first-order logic, the semantics of the event-B language should be formally related to the protocol model. We define well-formed conditions to guarantee that the event-B invariant is equivalent to the security property in the group protocol model. These conditions are particular to the group protocol model, and are essential to establish the equivalent event-B model. We show how an event-B model can be structured from group protocols model and then used to give a formal semantics to protocols which support proofs of their correctness. More precisely, we give a map from our protocol model to the event-B language. The event-B language allows the definition of invariant properties and provides an automatic model-checking based proof.

The proposed verification methodology consists of a number of steps as shown in Figure 3. In the first step, the group key protocol is specified formally using the model proposed in [14] in order to obtain precise protocol specifications. In addition, the secrecy property expected to be checked by the system is described informally. In the second step, the obtained specification is translated into event-B specification using mapping relations presented in Figure 4. From this mapping we obtain an event-B model that captures the features of the group protocol mode. Next, the secrecy property ϕ is specified as an invariant of the resulting event-B model I. Messages can be defined as a set with an enumeration of all possible secret and known messages. The intruder initial knowledge, $\mathbb{K}_0$, is directly defined as variable or set in the event-B *initialization* list. Secret messages are defined similarly. Protocol initial constraints, such as $\mathbb{K}_0 \subset \mathbb{M}$ and $\mathbb{S} \subset \mathbb{M}$, are defined as properties that will be included in the invariant. Protocol join or leave events are defined as event-B operations that update the intruder's knowledge and the set of secret messages, including the new generated key. Finally, the property is checked from the obtained global system specification using the event-B invariant checking tool Click'n'Prove.

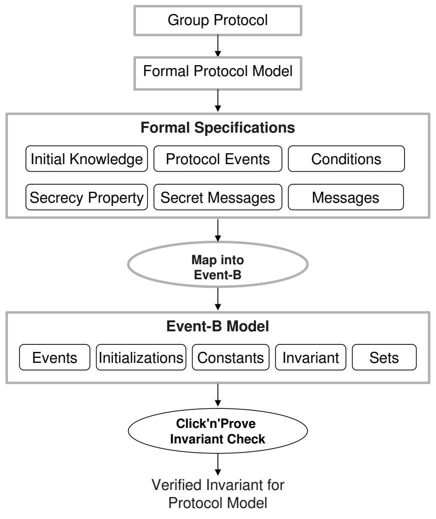

Fig. 3. Verification Methodology

Protocol events and execution traces are mapped into event-B events, messages generation conditions are mapped into events guards, and messages sets are used to generate event-B model constants properties. The initial knowledge is defined as event-B initializations, messages are mapped directly into sets, and finally the secrecy property is defined as an invariant for the event-B model. The generation of the target event-B model requires treating three parts: the static part which includes initializations and the constant properties of the protocol, the dynamic part that represents events of the protocol, and finally, enriching the resulting model with invariants describing the required secrecy properties.

The event-B semantics is close to the protocol model semantics. This relationship is demonstrated by establishing a well-formed link between the semantics of both models. To achieve this link, we are interested in showing that if the invariant I holds for event-B machine M, then the safety property ϕ must hold for the group protocol model $\mathbb{G}$. Formally, $(M \models I) \Rightarrow (\mathbb{G} \models \phi)$. In terms of equivalence between the two models, we can say that a protocol model $\mathbb{G}$ is equivalent to an event-B model M, with regards to the security property, if the property ϕ holds in the model $\mathbb{G}$, and the invariant I holds in the model M. To illustrate this equivalence, we need to show that $I \Rightarrow \phi$. Therefore, it is enough to show that the invariant I, with regards to M, implies the safety property ϕ, with regard to $\mathbb{G}$. The full details of the semantical link can be found in [13].

Under certain conditions, we guarantee that when the invariant holds in event-B model, the security property definition holds for the group protocol model. These predicates should be considered carefully when providing the event-B implementation. Properties that can be expressed as invariants are verified using the translation process and event-B tool. Other properties could be verified directly with model checking. This is illustrated in Theorem 4 shown below.

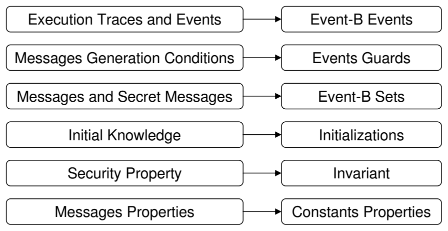

Fig. 4. Mapping protocol primitives into event-B

Theorem 4. *Secrecy Soundness. A group protocol, G, satisfies its secrecy property, ϕ, if there is an equivalent event-B model, M ($G \triangleq M$), that satisfies an event-B invariant, I, that implies the property ϕ. Let $(G \triangleq M)$, $(M \models I)$, and $(I \Rightarrow \phi)$ be correct lemmas, then*
$((G \triangleq M) \wedge (M \models I) \wedge (I \Rightarrow \phi) \Rightarrow (G \models \phi)$.

This method was applied on a Tree-based Group Diffie-Hellman protocol that generates a key in a distrusted group. We show how the conditions defined for the correctness of the above model can be concretely applied on a real protocol. The intended secrecy property, along with its conditions, are efficiently defined and checked as event-B invariant [13].

7 Conclusion

The correctness of group key protocols in communication systems remains a great challenge because of the sensitivity of the services provided. In this paper, we illustrated the need for a verification methodology for a class of protocols that deal with group key distribution.

A framework for modeling and verification of group key management protocols that combine three approaches is proposed. We defined a formal model for group key protocols based on a set of generic requirements of group key distribution protocols. In addition we defined a set of rank function that were used in our framework and provided the proof of soundness of these functions. The first method in our framework is based on rank theorems to enable and mechanize the verification procedure of security properties for group oriented protocols in PVS theorem proving. Secondly, we provided an inference system defined over rank functions. The approach is based on an elegant and natural proof strategy for the verification of group key protocols. Finally, we introduced an event-B based method based on a formal link between the semantics of group protocols model and event-B. We defined a well-formed connection between event-B invariant and the security property based on conditions to guarantee that the invariant verified in event-B is equivalent to the security property. This latter method is based on an abstracted first-order logic model in order to enable automatic invariant checking for security properties.

Even though we used three different approaches in our framework, we believe that there are some limitations for our framework. In general our methods are based on perfect cryptography conditions. Also, in our methods we tried to target the non-algebraic nature of protocols and analyzed their distributive nature. In addition, the application of the inference system was illustrated on a group protocol in the existence of an active adversary, it will be interesting to investigate the application of the method on similar protocols in the existence of a passive adversary. Also, even the event-B approach is more automated, only invariant properties can be modeled and verified. This is due to the target model and verification tool, namely, event-B and Click'n'Prove.

As future work, we intend to extend some features of our framework in order to be able to apply it on more challenging properties such as key independence (or collusion). Another direction is to provide an implementation of the inference system itself, rather than defining it in a theorem prover. This will provide more flexibility for modeling different protocols, however, If we implement our inference system, then we need to implement some strategies in order to guarantee the success of the verification process. If necessary the user can help the system in order to find an attack. Another open issue is to provide an implementation for the inference system that allows partial proofs based on divergence. For the event-B based approach, it will be more interesting to consider more dynamic properties: forward and backward secrecy, and the most interesting case is key independence. However, in order to achieve this, major modifications of the approach are required to support the distributive nature of the protocol because of the limitations of event-B tools.

References

1. A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, and G. Trouessin. Organization Based Access Control. In *International Workshop on Policies for Distributed Systems and Networks*, pages 120–131. IEEE Computer Society Press, June 2003.
2. J. Abrial. *The B-Book: Assigning Programs to Meanings.* Cambbridge University Press, 1996.
3. J. Abrial and D. Cansell. Click'n'Prove: Interactive Proofs within Set Theory. In *Theorem Proving in Higher Order Logics*, volume 2758 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, 2003.

4. N. Benaissa, D. Cansell, and D. Méry. Integration of Security Policy into System Modeling. In *Formal Specification and Development in B*, volume 4355 of *Lecture Notes in Computer Science*, pages 232–247. Springer-Verlag, January 2007.
5. M. Butler. On the Use of Data Refinement in the Development of Secure Communications Systems. *Formal Aspects of Computing*, 14(1):2–34, 2002.
6. F. Crazzolara. *Language, Semantics, and Methods for Security Protocols*. PhD thesis, BRICS, Denmark, May 2003.
7. F. Crazzolara and G. Winskel. Events in Security Protocols. In *ACM conference on Computer and Communications Security*, pages 96–105. ACM Press, 2001.
8. R. Delicata and S. Schneider. A Formal Model of Diffie-Hellman using CSP and Rank Functions. Technical Report CSD-TR-03-05, Department of Computer Science, Royal Holloway, University of London, 2003.
9. R. Delicata and S. Schneider. Temporal Rank Functions for Forward Secrecy. In *Computer Security Foundations Workshop*, pages 126–139, Washington DC, USA, June 2005. IEEE Computer Society Press.
10. W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
11. B. Dutertre, V. Crettaz, and V. Stavridou. Intrusion-Tolerant Enclaves. In *Proc. IEEE International Symposium on Security and Privacy*, pages 216–224, May 2002.
12. 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 *Lecture Notes in Computer Science*, pages 121–136. Springer-Verlag, 1997.
13. A. Gawanmeh, L. J. Ben Ayed, and S. Tahar. Event-B based Invariant Checking of Secrecy in Group Key Protocols. In *Conference on Local Computer Networks*. IEEE Computer Society Press, To Appear in October, 2008.
14. A. Gawanmeh, A. Bouhoula, and S. Tahar. Rank Functions based Inference System for Group Key Management Protocols Verification. *International Journal of Network Security*, 8(2):207–218, 2009.
15. A. Gawanmeh and S. Tahar. Rank Theorems for Forward Secrecy in Group Key Management Protocols. In *21st International Conference on Advanced Information Networking and Applications*, pages 18–23. IEEE Computer Society Press, 2007.
16. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *Advances in Cryptology*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer-Verlag, 2003.
17. M. Layouni, J. Hooman, and S. Tahar. Formal Specification and Verification of the Intrusion-Tolerant Enclaves Protocol. *International Journal of Network Security*, 5(3):288–298, 2007.
18. 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.
19. O. Pereira and J. Quisquater. Some Attacks upon Authenticated Group Key Agreement Protocols. *Journal of Computer Security*, 11(4):555–580, 2004.
20. O. Pereira and J. Quisquater. On the Impossibility of Building Secure Cliques-Type Authenticated Group Key Agreement Protocols. *Journal of Computer Security*, 14(2):197–246, 2006.
21. P. Ryan and S. Schneider. *The Modelling and Analysis of Security Protocols: The CSP Approach*. Addison-Wesley, 2001.
22. S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, September 1998.
23. G. Steel, A. Bundy, and M. Maidl. Attacking a Protocol for Group Key Agreement by Refuting Incorrect Inductive Conjectures. In *Automated Reasoning*, volume 3097 of *Lecture Notes in Computer Science*, pages 137–151. Springer-Verlag, 2004.
24. M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.
25. N. Stouls and M. Potet. Security Policy Enforcement Through Refinement Process. In *Formal Specification and Development in B*, volume 4355 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 2007.
26. T. Truderung. Selecting Theories and Recursive Protocols. In *Concurrency Theory*, volume 3653 of *Lecture Notes in Computer Science*, pages 217–232. Springer-Verlag, 2005.