Rank Functions based Inference System for Group Key Management Protocols Verification

Amjad Gawanmeh[1], Adel Bouhoula[2], and Sofiène Tahar[1]

[1]Department of Electrical and Computer Engineering,

Concordia University, Montreal, Canada

Email: {amjad, tahar}@ece.concordia.ca

[2]Ecole Supérieure des Communications de Tunis

Ariana, Tunisia

Email:adel.bouhoula@supcom.rnu.tn

March 2008

Abstract

Design and verification of cryptographic protocols has been under investigation for quite sometime. However, most of the attention has been paid for two parties protocols. In group key management and distribution protocols, keys are computed dynamically through cooperation of all protocol participants. Therefore regular approaches for two parties protocols verification cannot be applied on group key protocols. In this paper, we present a framework for formally verifying of group key management and distribution protocols based on the concept of rank functions. We define a class of rank functions that satisfy specific requirements and prove the soundness of these rank functions. Based on the set of sound rank functions, 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. The above formalizations and rank theorems were implemented using the PVS theorem prover. We illustrate our approach by applying the inference system on a generic Diffie-Hellman group protocol and prove it in PVS.

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. 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. Some security properties like 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. 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. 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. We should also guarantee that all authorized members are able to access the group, at the same time unauthorized ones are unable to have this access.

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. 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 protocol. Also in group key protocols, the key should be computed through cooperation of all protocol participants. This makes the verification problem for group key distribution protocols more challenging. In addition, some protocol may contain unbounded number of data fields or unbounded number of sessions [13].

There are some trials to address modeling and verification of protocols that involve more than two parties, these are discussed in the next section. In this paper, we suggest an approach for the verification of group key management protocols. We define an inference system based on the idea of rank functions, which was used by Schneider *et al.* [7, 10, 23, 22]. 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 define an inference system that 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. We also define a special rule called *Attack* that represents the bottom of the system, and, when executed, illustrates an attack in the protocol. We formally prove the soundness and the completeness of the inference system for a sound rank function. We implement the verification of a generic Diffie-Hellman group protocol [25, 8] in the PVS theorem prover [19] based on the inference system approach. Although there is a considerable amount of work on GDH protocol verification, little effort has been put on machine assisted verification, specifically on theorem proving. The advantages of our framework are mainly its applicability for a class of protocols that is difficult to tackle. In addition, the inference system provides a natural and elegant proof strategy compared to existing approaches in the literature. The results we achieved are promising and can be applied on similar protocols.

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 [12]. 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.

We believe the contributions of this paper are: (1) The definition of a set of sound rank functions and proof of their soundness; (2) The combination of rank functions with basic cryptographic protocols operation to define primitive rules. These rules are combined together to obtain an inference system. The primitive rules are enough to model regular protocol operations; and (3) The system can be extended to support special and more complex operations that may exist in some protocols.

The rest of the paper is organized as follows. Section 2 discusses related work to ours. In Section 3, we overview preliminary definitions and notations we use. In Section 4, we define and prove a consistent set of rank functions that satisfy specific requirements for group key protocols. In Section 5, we describe the details of our rank functions based inference system, and prove the soundness and completeness of the system. Section 6 illustrates our approach by applying the inference system on the case study of the Group Diffie-Hellman protocol and provide an implementation in PVS for the verification of the protocol. 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. We discuss two main directions: (1) Group key protocols verification; and (2) Rank functions and theorem proving

Group Key Protocols Verification. Meadows *et al.* [17] provided a detailed specification of the requirements for Group Domain Of Interpretation and then formally analyzed the protocol with respect to these requirements using the NRL Protocol Analyzer. However, the problem with this approach is that no general set of requirements for protocols requirements can be applied on a specific protocol, or can be used for the refinement of protocol specifications during the design process is provided.

Pereira and Quisquater [20] 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 recent work Pereira and Quisquater [21] provided a systematic way to derive an attack against any A-GDH-type 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 the general problem of deciding whether a term is derivable in an attacker algebra with the equational theory of multiplication, or whether a particular symbolic attack trace has a feasible instantiation [18].

In a similar work, Sun and Lin [26] 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. The above work is related to ours since they try to provide a method to verify complex group protocols. However, the solutions provided in all cases focus only on specific aspects of one protocol rather than focusing on general requirements.

In another related work, Steel *et al.* [24] 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. More recently, Truderung [27] 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. the 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 formal analysis for GDH style protocols. Some used symbolic approaches such as the work of Mazaré [16] who proposed a symbolic model to analyze cryptographic protocols using bilinear pairing. Boreale and Buscemi [3] used another symbolic approach to verify protocols checking consistency of symbolic traces, however they require an *a-priori* upper bound on the number of participants. Millen and Shmatikov [18] considered a symbolic approach to reason about GDH protocol style operators, such as exponentiation, with a bounded number of role instances. In another effort, Bresson *et al.* [4, 5, 6] discussed the GDH problem thoroughly and suggested a model for this class of protocols in the presence of malicious participants. Kats *et al.* [12, 11] addressed the case of attacks by malicious insiders for authenticated key exchange protocols. Finally, Abadi [1] discussed decidability issues for knowledge formalizations of both participants and attackers. These recent amount of work provide mathematical models for GDH like protocols, in addition, they focus on rigorous analysis of security of these protocols, however, these approaches lack the ability to mechanize the proof, in particular with theorem proving techniques.

Rank Functions and Theorem Proving. Dutertre and Schneider [10] used an embedding of CSP in PVS in order to verify the authentication property of Needham Shroeder public key protocol. They proposed the idea of rank functions in order enable CSP verification of Needham Shroeder protocol. Later, Schneider [23, 22] used the idea of rank functions for the verification of CSP (Communication Sequential Process). 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.* [10, 23, 22, 7] 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.

In a more recent work, Layouni *et al.* [14] 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 [9]. 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.

Archer [2] provided a mechanized correctness proof of the basic TESLA protocol based on establishing a sequence of invariants for the protocol using the tool 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 all the above efforts in this area, we noticed that there is a need for a mechanized approach, since most of the approaches in the literature focused on mathematical proofs, neglecting machine assisted verification techniques, such as theorem proving. We try to fill this gap in the work we introduce here.

3 Preliminaries

In this section we present our formal model and the notations we will use throughout this paper.
$\mathbb{M}$: 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.

We define traces since it will be used to prove the soundness of the inference system. We use t to represent 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.

A *rank function* is a map between the set of facts about the protocol and the set of natural integers. The set of facts include protocol events, protocol execution traces, keys, and messages. This map assigns a value or *rank* to each fact, such that facts that can be generated by the protocol have positive rank, and facts that 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. This map function will be useful in partitioning the message space and enabling mechanized proof of security protocols properties. The set of events and traces are concretely defined by the protocol, which allows defining them at different levels of abstraction in the final step of our approach.

The definition of the rank function is formally given as follows:

Definition 3.1. *Rank Functions [22]. A rank function ρ is a map function $\rho : \mathbb{M} \rightarrow \mathbb{Z}$ which maps the set of all messages into integers.*

4 Rank Functions based Inference System

4.1 Rank Functions Soundness

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 4.1. *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 4.2. *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* $m_1, m_2 \in \mathbb{M}$*.*

Definition 4.3. *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 is able to apply the inverse.*

Definition 4.4. *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 4.1. *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.

Proof. We prove this theorem using absurdum, by assuming that the rank function evaluates to zero then we show that for all possible execution events, there exists no message in the intruder's initial set of knowledge that can generate this zero rank.

Assume there exists $m \in \mathbb{M}$ such that $\rho(m) = 0$, the message m is either in the intruders initial set of knowledge (case (a) below) or generated after applying on single event on a message in the intruder's initial set of knowledge (case (b) below), therefore, we can write:

$$\rho(m) = 0 \ \Rightarrow \exists\ m' \in \mathbb{K}_0 \text{ such that } \begin{cases} (a) : \ m = m' \quad \text{or} \\ (b) : \ m = e(m') \end{cases}$$

Now we consider both cases and show the contradiction of the assumption:

For case (a): $m = m'$, since the rank function ρ is *initially sound* by definition, then there is a clear contradiction which can be stated as follows:

Only messages in $\mathbb{S}$ have the rank zero: $\rho(m') = 0 \ \Rightarrow m' \in \mathbb{S}$. However, messages in $\mathbb{S}$ are not in $\mathbb{K}_0$: $m' \in \mathbb{S} \Rightarrow m' \notin \mathbb{K}_0$. Which contradicts the assumption stated above: $m' \in \mathbb{K}_0$. Therefore $\rho(m') > 0$ is valid.

For case (b): $m = e(m')$, and $\rho(m) = 0$: the message m is generated after the application of one single event e on a message m' from $\mathbb{K}_0$.

Since ρ is bounded, then we can say that the rank of the message m is bounded by the rank of the message m', we can write this as follows:

$m = e(m') \Rightarrow \rho(m') - 1 \leq \rho(m) \leq \rho(m') + 1$

By assumption, we have $\rho(m) = 0$. This means that either $\rho(m') = 0$ or $\rho(m') = 1$ (from above inequality). Now we consider both cases: The case where $\rho(m') = 0$ is similar to case (a) above, and will lead to the same contradiction.

We consider the second possibility: $\rho(m') = 1$, $m = e(m')$ and $\rho(m') = \rho(m) + 1$.

ρ is *invertible*, therefore, there is an event e' that we can apply on the message m to generate the message m', we can write:

$m = e(m')$, $m' = e'(m)$ and therefore $m' = e'(e(m'))$ (which is a typical invertible relation in encryption).

We have $\rho(m') = 1$ and $m' = e'(m)$ therefore $\rho(m) = \rho(m') - 1$ which means that the message m' is generated after one single application of an event e' in the set of events on the message m: $m' = e'(m)$.

If the intruder can apply one single invertible event, then he/she can apply the other one. Since the events e and e' are invertible, and the intruder can apply e on m' to generate m, therefore he/she can also apply the event e' on m to generate m'.

This means that $e, e' \in \mathbb{E}$, $m' \in \mathbb{K}_0$, and $m \in \mathbb{S}$, so $\rho(m) = 0 \Rightarrow m \notin \mathbb{K}_0$ or $e \notin \mathbb{E}$ which contradicts the assumption stated above.

The fact that the intruder cannot generate secret knowledge from its initial knowledge (without executing the protocol), i.e., the intruder cannot decrypt a message encrypted with a secret key. $\square$

4.2 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 tuple $\langle m, c\rangle$ 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 $\perp$. Figure 1 shows the set of rules in the inference system. 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*.

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

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:* $\frac{\mathbb{K}\cup\{\langle m,0\rangle\}}{\perp}$

Figure 1: Inference System

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.

4.3 Soundness and Completeness

In this subsection 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 4.2. *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* $\mathbb{P}$.

Proof. We prove this theorem by deduction, where we assume the left hand side and deduce the right hand side of the theorem.

Assume there exists $m \in \mathbb{K}$ such that $\rho(m) = 0$

Given that $m \in \mathbb{K}$, then we have $m = m_0 \in \mathbb{K}_0$, $m = e(m_0)$, or $m = T_n(\mathbb{E}, \mathbb{M})$. However, $\rho(m) = 0 \Rightarrow m \notin \mathbb{K}_0$, the rank function is sound since $\rho \in \mathbb{R}$.

$\rho(m) = 0 \Rightarrow \nexists\ (e \in \mathbb{E},\ m_0 \in \mathbb{K}_0) : m = e(m_0)$, since the rank function is sound.

It is clear now that there exists $m \in \mathbb{K}$ such that $\rho(m) = 0 \Rightarrow \exists T_n \in \mathbb{T}$ such that $m = T_n(\mathbb{E}, \mathbb{M})$, which means that there exists a trace the intruder can execute to compute m. Also $\rho(m) = 0 \Rightarrow m \in \mathbb{S}$, since only messages in $\mathbb{S}$ have the rank zero. Hence, we find that $m \in \mathbb{S}$ and $m \in \mathbb{K}$. Therefore, the rule $attack$ can be applied. Then, we can write:

$attack \Rightarrow m \in \mathbb{S}$ and $m \in \mathbb{K}$. This means that there exists an attack in the system.

□

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. We can view it as the complimentary case of the above theorem.

Corollary 4.1. *Absence of Attack*

Assuming the same conditions as Theorem 4.2, 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 4.3. *Completeness*

Assuming the same conditions as Theorem 4.2, 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* $\mathbb{P}$.

Proof. We prove this theorem by absurdum. We assume the right hand side is false and deduce a contradiction to the left hand side of the theorem.

Assume there exists an $attack$ in the system, then we can write:

$\exists attack \Rightarrow \exists m$ such that $m \in \mathbb{K}$ and $m \in \mathbb{S}$.

A message m in the intruder's set of knowledge $\mathbb{K}$ means that either the message is in his initial set of knowledge $\mathbb{K}_0$, is generated by applying one single event in $\mathbb{E}$, or is generated after applying a trace in $\mathbb{T}$.

$m \in \mathbb{K} \Rightarrow m = m_0 \in \mathbb{K}_0$, $m \in \mathbb{K}_1 = \mathbb{E}(\mathbb{K}_0)$, or $m = T_n(\mathbb{E}, \mathbb{M})$.
However, since $\rho \in \mathbb{R}$ is sound and $m \in \mathbb{S}$ then $m \notin \mathbb{K}_0$ and $m \notin \mathbb{K}_1$.
Therefore, $m = T_n(\mathbb{E}, \mathbb{M})$
Since $m \in \mathbb{S}$, then the rank of this message $\rho(m) = 0$.
So $\exists attack \Rightarrow \rho(m) = 0$.
Therefore, we conclude that $\rho(m) > 0 \Rightarrow \nexists attack$

□

Corollary 4.2. *Detecting Attacks*

Assuming the same conditions as Theorem 4.2, 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 4.2 and 4.3 and Corollaries 4.1 and 4.2 provide the formal link between the protocol model and the implementation model in PVS. Therefore, the soundness of the above theorems and corollaries represents the sounds of the verification technique.

In summary, the inference system we defined can prove that an attack exists in the protocol using Soundness Theorem 4.2. However, the limitation of the Soundness Theorem comes in the type of implementation that will be used. In case of theorem proving, there is no guarantee that the attack can be generated. This is a general problem and is applicable on any approach for tool supported verification this type of protocols.

In case we want to prove that there is no attack in the protocol, the inference system can diverge in case we apply the Completeness Theorem 4.3. The inference system may terminate with a result, depending on the nature of the protocol and the strategies used while conducting theorem proving. However, there is no guarantee for termination because of two reasons: first, the type of problem we are trying to solve has unbounded number of participants, and unbounded message space. Second, the lower level implementation method, theorem proving, does not guarantee termination, which means the inference system may run infinitely without reaching a result.

However, we still can reason about absence of attacks in protocols using Completeness Theorem 4.3. An *indirect* proof can be generated using the completeness theorem 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.

4.4 PVS Embedding of the Inference System

The most important and challenging part is how to define the inference system, and how to instantiate it by the dishonest user. For this purpose, we represent each inference rule as a PVS deduction statement which allows the user who is executing these rules to compute new messages and add it to his/her own set of knowledge. In our case, the intruder is such user. These rules are defined based on the events of the GDH protocol. We abstract the mathematical power operation used in the protocol, since power operator is not supported in PVS. Therefore α^N is represented in PVS by the $alphaN$, where N represents the power of alpha, so when applying a rule that generates α^{N1N2} from α^{N1} and $N2$ we just multiply the *nounce* in $alpha$ by $N2$, and the rank function is defined of a type that maps $MESSAGE$ to int.

```
MESSAGE : TYPE
ALPHA:  TYPE
FROM MESSAGE
nounc: int
alpha: int
alphaN : ALPHA
m: MESSAGE
rankf: [MESSAGE -> int]
```

Next we define the appropriate inference rules for this protocol. These rules are used by the intruder in order to build his/her set of knowledge starting from his initial knowledge and applying one rule at a time. The first rule is *compose*, where two message records are used to generate one new message with one rank. The second one we show here is the decompose message, which is used to separate an already composed message. Similarly, we define *encrypt* and *decrypt* rules, where the rank is updated then the rules are applied. We also show the rule *expo* which is used to generate the α^N messages. Finally we show the rule attack which is executed when there is a message of rank zero in the intruders set of knowledge.

```
rule_compose(msg1: MESSAGE, msg2 : MESSAGE) : MESSAGE
   = (comp(msg1,msg2), min(rankf(msg1), rankf(msg2)))

rule_decomp(msg1: MESSAGE) : [MESSAGE, int, MESSAGE, int]
   = (left_part(msg1),rankf(msg1), (right_part(msg1), rankf(msg1))

rule_encr(msg1: MESSAGE, key: MESSAGE) : [MESSAGE, int]
   = (append(msg1,key), rankf(msg1)-1)

rule_decr(msg1: MESSAGE, key: MESSAGE) : [MESSAGE, int]
   = (extract(msg1,key), rankf(msg1)+1)

rule_expo(a:ALPHA,N:int) : [ALPHA, int]
   = (alphaN, rankf(a)-1)

rule_attack(msg: MESSAGE): bool =
   rankf(msg) = 0
```

5 Application: GDH Protocol

In order to illustrate the proposed verification methodology, we consider the Group Diffie-Hellman (GDH) protocol [25, 8], which is a basic group key management protocol widely studied in the literature. In the first part, we show 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 for two, three, and n-users case.

Although the protocol is not a challenging case study, since it has been studied quite enough in the literature, we use it as illustrative case to show the feasibility of our approach, not to prove something that has been already proved before. In addition, we provide, to the best of our knowledge, a new attempt to use theorem proving in the context of group key protocols verification. In the first part of this section we show how the inference system can be applied directly on the GDH protocol of 4 participants. We demonstrate how the attack is generated in the step by step application of the inference system. In the second part, we discuss the embedding and verification of the protocol using theorem proving in PVS.

The protocol is used to generate and distribute a safe key between a group of members over a non-secure network. It consists of two stages: upflow and downflow. The first stage is used to collect contributions from all group members that will be used in calculating the group key. Given n members in the group: $P_1, P_2, ..., P_n$, who are willing to generate a secret key that will be used among them, the protocol works as follows: in the upflow stage, every intermediate member P_i receives a collection of intermediate values from member P_{i-1}, computes another value, by adding his/her own share of the key, appends it to the values he/she received, and forwards this information to the next group member P_{i+1}. In the downflow stage, the last member appends his own share of the key to every value he received and sends them back to previous members. This way, every member receives partial information to compute the key.

For example, in a group of 4 members, the first member uses a generator α and a random number N_1, computes $\{\alpha^{N_1}\}$ and forwards it to member P_2. P_2 chooses a random number N_2 and computes $\alpha^{N_1 N_2}$, then forwards $\{\alpha^{N_1}, \alpha^{N_1 N_2}\}$ to P_3. P_3 computes $\alpha^{N_1 N_2 N_3}$ and forwards $\{\alpha^{N_1 N_3}, \alpha^{N_1 N_2}, \alpha^{N_1 N_2 N_3}\}$ to P_4. Now P_4 uses the last value and a random number he/she generates, N_4, to compute the group key $\alpha^{N_1 N_2 N_3 N_4}$. For the downflow stage, the member raises all other values to N_4 and sends back to P_3 $\{\alpha^{N_4}, \alpha^{N_1 N_2 N_4}, \alpha^{N_1 N_3 N_4}\}$. P_3 uses the latest value $\alpha^{N_1 N_2 N_4}$ and his/her own random number N_3 to compute the key, raises the first value to N_3 and sends $\{\alpha^{N_3 N_4}, \alpha^{N_1 N_3 N_4}\}$ back to P_2, who uses the last one to compute the key, then computes and sends $\{\alpha^{N_2 N_3 N_4}$ to P_1, who can compute the same key.

We choose a group of three members, $P_1, P_2,$ and P_3, and apply our verification approach on the protocol, by defining the intruder I, and executing the inference system by this intruder. The first step is to define the rank function ρ for the set of messages in the message space M as follows:

$$\rho(m) = \begin{cases} 0, & if\ m \in \{N_1, N_2, N_3, \alpha^{N_1 N_2 N_3}, \alpha^{N_i N_2 N_3}, \alpha^{N_1 N_i N_3}, \alpha^{N_1 N_2 N_i}, \\ & \qquad\qquad \alpha^{N_1 N_2 N_3 N_i}\} \\ 1, & if\ m \in \{N_i, \alpha, \alpha^{N_1}, \alpha^{N_2}, \alpha^{N_3}, \alpha^{N_1 N_2}, \alpha^{N_1 N_3}, \alpha^{N_2 N_3}\} \end{cases}$$

Here, $\alpha^{N_1 N_2 N_3}$ represents the group key members intend to generate, and N_i represent the intruders nounce. For the protocol to be correct, there should not be a way for an intruder to share a key with the rest of the members that can be considered as the group key, even if its different from the group key the members intend to generate, simply because the members have no clue about the final key, until each has enough shares from other members to compute it. Therefore, we consider $\alpha^{N_i N_2 N_3}, \alpha^{N_1 N_i N_3}, \alpha^{N_1 N_2 N_i}, \alpha^{N_1 N_2 N_3 N_i}$ as assumed group keys that the intruder should not be able to share with the members making them believe it is the group key they intended to share when they started the protocol.

Then, we define the events that can be executes by members. This includes $send(m)$, $recv(m)$, $expo(m, n)$, $comp(m_1, m_2)$, $decomp(m_1, m_2)$, and $block(m)$. The latest is an event that can be executed only by the intruder. The rank function

ρ can be defined for these events as follows:
$\rho(send(m)) = \rho(m)$, $\rho(recv(m)) = \rho(m)$, $\rho(expo(m, n)) = \rho(n) + 1$, $\rho(comp(m_1, m_2)) = \rho(min(m_1, m_2))$, $\rho(block(m)) = \rho(m)$. For the event $decomp(m_1.m_2)$, $\rho(m_1) = \rho(m_1.m_2)$ and $\rho(m_2) = \rho(m_1.m_2)$.

The inference system is composed of an inference rule for each of these events in addition to the inference rule *Attack* defined above. The event $expo(m, n)$ models the exponent function used in the protocol. The intruder starts with an initial set of knowledge $\mathbb{K}_0 = \{N_i, \alpha\}$, and we assume he has the ability to fully monitor the network, send, receive, or block messages on his/her will. We also assume that signing messages is not used between members.

The upflow stage of the protocol is started by member P_1, who uses α, N_1, and $expo(\alpha, N_1)$ to compute α^{N_1}, where $\rho(\alpha^{N_1}) = \rho(N_1) + 1 = 1$. P_1 sends this message to next user in the group, P_2. The intruder, apply the inference rules corresponding to the event $recv(\alpha^{N_1})$, therefore, $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_1}\}$. We use $\mathbb{K}'$ to represent the intruder's set of knowledge before he executes the inference rule. The intruder similarly applies the inference rule $block(\alpha^{N_1})$, then, computes α^{N_i} and sends it to the member P_2, who computes $\alpha^{N_i N_2}$, uses the $compose$ rule to generate $\alpha^{N_i}.\alpha^{N_i N_2}$ and forwards it to the last member P_3. The intruder can receive this message and block it, then, composes and sends to P_3 the message $\alpha^{N_1}.\alpha^{N_i N_2}$. P_3 receives it, decomposes it and uses the last term $\alpha^{N_i N_2}$ to compute the key $\alpha^{N_i N_2 N_3}$ believing it is the intended group key. At this point the intruder's set of knowledge is updated to the following $\mathbb{K} = \{N_i,\ \alpha,\ \alpha^{N_1},\ \alpha^{N_1 N_i},\ \alpha^{N_i N_2}\}$

Member P_3 then starts the downflow stage, and computes, composes and sends $\alpha^{N_3}.\alpha^{N_1 N_3}$ back to user P_2. The intruder will receive and block this message, and therefore, updates his knowledge such that $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_3},\ \alpha^{N_1 N_3}\}$. The intruder composes and sends to P_2 the message $\alpha^{N_3}.\alpha^{N_i N_3}$. P_2 receives the message and he/she uses the last term $\alpha^{N_i N_3}$ to compute his/her key $\alpha^{N_i N_2 N_3}$. Then he sends back to user P_1 the message $\alpha^{N_2 N_3}$. The intruder receives this message, updates his set of knowledge, blocks the message, and instead, he sends $\alpha^{N_i N_3}$ to P_1 who, in turn will use it to compute his key $\alpha^{N_1 N_3 N_i}$. The intruder updates his/her set of knowledge at this point, and it will be
$\mathbb{K} = \{N_i,\ \alpha,\ \alpha^{N_1},\ \alpha^{N_1 N_i},\ \alpha^{N_i N_2},\ \alpha^{N_1 N_3},\ \alpha^{N_i N_2 N_3},\ \alpha^{N_i N_1 N_3}\}$. Figure 2 shows a step by step execution of the inference system by the intruder.

Now, given a fair system, the intruder can apply the inference rule *Attack*, since there is a message in his/her set of knowledge that has the rank zero.

$I\ :\mathbb{K}_0 = \{N_i, \alpha\}$

P_1: $\mathit{expo}(\alpha, N_i) \mapsto \alpha^{N_i}$; $\mathbb{K} = \{N_i, \alpha, \alpha^{N_i}\}$

P_1: $\mathit{expo}(\alpha, N_1) \mapsto \alpha^{N_1}$, $\rho(\alpha^{N_1}) = \rho(N_1) + 1 = 1$

P_1: $\mathit{send}(\alpha^{N_1})$; $P_1 \to P_2$

$I\ :\mathit{recv}(\alpha^{N_1}), \mathit{block}(\alpha^{N_1})$; $\mathbb{K} = \{N_i, \alpha, \alpha^{N_1}\}$

$I\ :\mathit{send}(\alpha^{N_i})$; $I \to P_2$

P_2: $\mathit{recv}(\alpha^{N_i})$;

P_2: $\mathit{expo}(\alpha^{N_i}, N_2) \mapsto \alpha^{N_iN_2}$

P_2: $\mathit{compose}(\alpha^{N_i}, \alpha^{N_iN_2}) \mapsto \alpha^{N_i}.\alpha^{N_iN_2}$

P_2: $\mathit{send}(\alpha^{N_i}.\alpha^{N_iN_2})$; $P_2 \to P_3$

$I\ :\mathit{recv}(\alpha^{N_i}.\alpha^{N_iN_2}), \mathit{block}(...)$; $\mathbb{K} = \{N_i, \alpha, \alpha^{N_1}, \alpha^{N_i}.\alpha^{N_iN_2}\}$

$I\ :\mathit{decompose}(\alpha^{N_i}.\alpha^{N_iN_2}) \mapsto \alpha^{N_i}, \alpha^{N_iN_2}$; $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_iN_2}\}$

$I\ :\mathit{compose}(\alpha^{N_1}, \alpha^{N_iN_2}) \mapsto \alpha^{N_1}.\alpha^{N_iN_2}$

$I\ :\mathit{send}(\alpha^{N_1}.\alpha^{N_iN_2})$; $I \to P_3$

P_3: $\mathit{recv}(\alpha^{N_1}.\alpha^{N_iN_2})$

P_3: $\mathit{decompose}(\alpha^{N_1}.\alpha^{N_iN_2}) \mapsto \alpha^{N_1}, \alpha^{N_iN_2}$

P_3: $\mathit{expo}(\alpha^{N_iN_2}, N_3) \mapsto \alpha^{N_iN_2N3}$; P_3 generates a bad group key $\alpha^{N_iN_2N3}$

P_3: $\mathit{expo}(\alpha^{N_1}, N_3) \mapsto \alpha^{N_1N3}$;

P_3: $\mathit{expo}(\alpha, N_3) \mapsto \alpha^{N3}$;

P_3: $\mathit{compose}(\alpha^{N_3}, \alpha^{N_1N_3}) \mapsto \alpha^{N_3}.\alpha^{N_1N_3}$

P_3: $\mathit{send}(\alpha^{N_3}.\alpha^{N_1N_3})$; $P_3 \to P_2$

$I\ :\mathit{recv}(\alpha^{N_3}.\alpha^{N_1N_3}), \mathit{block}(...)$; $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_3}.\alpha^{N_1N_3}\}$

$I\ :\mathit{decompose}(\alpha^{N_3}.\alpha^{N_1N_3}) \mapsto \alpha^{N_3}, \alpha^{N_1N_3}$; $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_3}, \alpha^{N_1N_3}\}$

$I\ :\mathit{expo}(\alpha^{N_3}, N_i) \mapsto \alpha^{N_iN_3}$;

$I\ :\mathit{compose}(\alpha^{N_i}, \alpha^{N_iN_3}) \mapsto \alpha^{N_i}.\alpha^{N_iN_3}$

$I\ :\mathit{send}(\alpha^{N_i}.\alpha^{N_iN_3})$; $I \to P_2$

P_2: $\mathit{recv}(\alpha^{N_i}.\alpha^{N_iN_3})$

P_2: $\mathit{decompose}(\alpha^{N_i}.\alpha^{N_iN_3}) \mapsto \alpha^{N_i}, \alpha^{N_iN_3}$

P_2: $\mathit{expo}(\alpha^{N_iN_3}, N_2) \mapsto \alpha^{N_iN_2N3}$; P_2 generates a bad group key $\alpha^{N_iN_2N3}$

P_2: $\mathit{expo}(\alpha^{N_i}, N_2) \mapsto \alpha^{N_iN2}$

P_2: $\mathit{send}(\alpha^{N_iN_2})$; $P_2 \to P_1$

$I\ :\mathit{recv}(\alpha^{N_iN_2}), \mathit{block}(...)$

$I\ :\mathit{expo}(\alpha^{N_3}, N_i) \mapsto \alpha^{N_iN3}$;

$I\ :\mathit{send}(\alpha^{N_iN_3})$; $I \to P_1$

P_1: $\mathit{recv}(\alpha^{N_iN_3})$

P_1: $\mathit{expo}(\alpha^{N_iN_3}, N_1) \mapsto \alpha^{N_iN_1N3}$; P_1 generates a bad group key $\alpha^{N_iN_1N3}$

$I\ :\mathit{expo}(\alpha^{N_1N_3}, N_i) \mapsto \alpha^{N_iN_1N3}$; $\mathbb{K} = \mathbb{K}' \cup \{\alpha^{N_iN_1N_3}\}$

$I\ :\mathit{Attack}(\langle \alpha^{N_iN_1N_3}, 0\rangle) \mapsto \perp$

Figure 2: Applying the Inference System on GDH Protocol

PVS Implenetation. This manual step by step application of the inference system can be mechanized using available tools. In our case, we used the PVS theorem prover in order to show that the inference system will lead to the same attack. In our PVS model, we first define the sets of messages we used, including the set of all messages, secret messages, events, traces, intruders initial knowledge, intruders updated knowledge. We also define the dischonest user *I* and a set of *n* users who all together will participate in the protocol. We also define the intruder's initial set of knowledge to be α, N_i as stated above.

```
   M: VAR set[MESSAGE]
   Key: VAR set[KEY]
   K: VAR set[MESSAGE]
   A: VAR USER
   I: VAR USER
   S: VAR set[MESSAGE]

rule_initial(alpha: ALPHA)  =
    K_0 := [alpha,Ni]
```

In order to conduct the verification for the secrecy property in PVS, we first considered the simple case of GDH protocol where two users are establishing the secret key. For this purpose, we define two users and a dishonest user and the set of messages used in the protocol. In addition, we show how the intruder updates his set of knowledge when a message is sent between two users and blocked by the intruder. For illustration purposes, we show parts of the protocol implementation including $send$ and $receive$ operations that take place between users. The *GDHP_update* function applies the inference rules on the set of messages and then updates the intruders set of knowledge. The functions *send* and *receive*, if the message is not blocked, then it is received at the destination user, the intruder add the message to his set of knowledge, and finally he/she updates this set by executing the function *update*.

The secrecy property we verify for this protocol is defined as a lemma stating that the protocol satisfies this secrecy property if it does not execute the inference rule attack. The property shows that when the protocol is initiated by a user and the intruder can execute the the events of the protocol (rules in the inference system), then the intruder will be able to share a secret key between him/her and the user in the group. The property is defined as follows:

In the next step we did the verification for the same property by executing the

```
GDHP_update?(K, I): bool =
    (FORALL m: K  do K := union(K, inf_rule(I,m)))

GDHP_send?(UserA, UserB, m) =
        if(!block(UserA,UserB,m) then GDHP_recv(UserA,UserB,m)
        addMsg(K,m)
        GDH_update(K,I)

GDHP_recv?(UserA, UserB, m) =
        addMsg(UserB.knldgSet,m)
```

```
secrecy_prop_x : THEORY
  BEGIN
      secrecy_attack: LEMMA
         Reachable(rule_inital)AND knows(I,K_0)
                     IMPLIES Reachable(rule_attack)
  END forward_secrecy
```

inference system on a protocol between three users instead of two. The verification complexity and effort were more in this case, however there was no technical changes in the verification techniques and strategies used. In following we show how the property definition for the GDH protocol with three users in PVS:

```
secrecy_prop_3 : THEORY
  BEGIN
      A, B, C: VAR USER
      secrecy_attack: LEMMA
         Reachable(rule_inital)AND knows(I,K_0) AND GDHP(A,B,C)
                     IMPLIES Reachable(rule_attack)
  END secrecy_prop_3
```

The challenging part was to verify the N users case of GDH protocol. This was achieved by applying the same proof strategies used for the 3 users case for an array of n-users in order to show that the same attack can be generated in this case. In the following, we show the secrecy property definition for the GDH protocol for an array for n users:

Implementing and verifying this part in PVS required hundreds lines of code, including several proof strategies. We believe using the framework to verify similar protocols can be achieved in shorter time given the provided implementation of the inference system and the experience gained. As opposed to previous works, our approach give a simple, natural and elegant proof strategy. The computer

```
secrecy_prop_n : THEORY
  BEGIN
      n: VAR int
      users: array[n] of USER
      secrecy_attack: LEMMA
         Reachable(rule_inital)AND knows(I,K_0) AND GDHP(users)
                     IMPLIES Reachable(rule_attack)
  END secrecy_prop_n
```

experiment shows that our technique is very promising.

6 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. While most approaches in the literature target cryptographic properties for two parties protocols, the verification problem for group key distribution protocols is more challenging because properties for these protocols are not trivial extensions of the two-parties models. For example, the fact that a group member computes a bad key can remain undiscovered by the group, specially for a large group.

We provided a new approach for the verification of group key management protocols by using 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. We believe to have contributed in defining a set of rank function and providing the proof of soundness of these functions, and a complete and sound inference system for verification of group protocols. Discovering if the protocol is vulnerable for attacks from an intruder is done by executing the inference system by a model of the intruder with specific assumptions about the protocol and the intruder. We applied this system on a group key protocol, the Group Diffie-Hellman protocol. Although the protocol is not a challenging case study, since it has been studied quite enough in the literature, we use it as illustrative case to show the feasibility of the proposed approach. Therefore we provide a mechanized approach using theorem proving in the context of group key protocols verification. We found that, under certain assumptions, the intruder can force members using the protocol to generate bad keys, which is a well known weakness point in the protocol. The results we achieved are very promising and we believe that our framework can be applied efficiently on protocols of similar complexity level.

As future work, an open issue is applying abstraction techniques on the rank function to be able to model them in first-order logic, and therefore, make model checking feasible using supporting tools. This will reduce the complexity of the verification process and make it more automatic, but it will limit the applicability of the method on large scale and complex protocols. In addition, the application of the inference system on similar protocols in the existence of a passive adversary should be investigated further. 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. Other group protocols [15] will be considered for future study under this approach.

References

[1] M. Abadi and V. Cortier. Deciding Knowledge in Security Protocols under Equational Theories. *Theoretical Computer Science*, 367(1):2–32, 2006.

[2] 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.

[3] M. Boreale and M. Buscemi. Symbolic Analysis of Crypto-Protocols Based on Modular Exponentiation. In *Mathematical Foundations of Computer Science*, volume 2747 of *Lecture Notes in Computer Science*, pages 269–278. Springer-Verlag, 2003.

[4] E. Bresson, O. Chevassut, and D. Pointcheval. Provably-Secure Authenticated Group Diffie-Hellman Key Exchange. *ACM Trans. on Information and System Security*, 10(3), August 2007.

[5] E. Bresson and M. Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *Proceedings of the 4th Autonomic and Trusted Computing Conference*, volume 4610 of *Lecture Notes in Computer Science*, pages 395–409. Springer-Verlag, July 2007.

[6] E. Bresson, M. Manulis, and J. Schwenk. On Security Models and Compilers for Group Key Exchange Protocols. In *the second International Workshop on Security*, volume 4752 of *Lecture Notes in Computer Science*, pages 292–307. Springer-Verlag, October 2007.

[7] 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.

[8] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

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

[10] 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.

[11] J. Katz and J. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *the 12th ACM Conference on Computer and Communications Security*, pages 180–189. ACM Press, 2005.

[12] 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.

[13] R. Küsters. On the Decidability of Cryptographic Protocols with Open-ended Data Structures. *International Journal of Information Security*, 4(1–2):49–70, 2005.

[14] 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.

[15] M. Manulis. Security-Focused Survey on Group Key Exchange Protocols. Technical Report 2006/03, Rohr-University of Bochum, November 2006.

[16] L. Mazaré. Computationally Sound Analysis of Protocols using Bilinear Pairings. In *Preliminary Proceedings of International Workshop on Issues in the Theory of Security*, pages 6–21, Braga, Portugal, March 2007.

[17] 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.

[18] J. Millen and V. Shmatikov. Symbolic Protocol Analysis with an Abelian Group Operator or Diffie-Hellman Exponentiation. *J. Comput. Secur.*, 13(3):515–564, 2005.

[19] 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.

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

[21] 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.

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

[23] S. Schneider. Verifying Authentication Protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, September 1998.

[24] 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.

[25] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *Proc. of the 3rd ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

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

[27] T. Truderung. Selecting Theories and Recursive Protocols. In *Concurrency Theory*, volume 3653 of *Lecture Notes in Computer Science*, pages 217–232. Springer-Verlag, 2005.