Automatic Verification of a Key Management Architecture for Hierarchical Group Protocols

Mohamed Salah Bouassida, Najah Chridi, Isabelle Chrisment, Olivier Festor and Laurent Vigneron

LORIA, Campus scientifique, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex - France
tel : +33-3-83-59-30-49 - fax : +33-3-83-41-30-79

Emerging applications require secure group communications involving hierarchical architecture protocols. Designing such secure hierarchical protocols is not straightforward, and their verification becomes a primordial issue in order to avoid any possible security attack and vulnerability. Several attempts have been made to deal with formal verification of group protocols, but, to our knowledge, none of them did addressed hierarchical ones.
This paper presents the specific challenges and security issues of hierarchical secure group communications, and an overview of the work done for their verification. We have chosen the back-end AtSe of the AVISPA tool, to verify one of these protocols, as it enables to deal with the exponentiation of Diffie-Hellman often used in group key management.

Keywords: Hierarchical Group Protocols, Security, Automatic Verification, AVISPA Tool

1 Introduction

The last decade saw a linear exponential increase in applications requiring an unbounded number of participants. Varying from dedicated applications to large public ones, the importance of the design of secure group protocols [MS98, WGL98] has grown. To secure their communications, group members need to use a secret key named group key which has to be updated following the dynamics of the group (join or leave operations,...). The establishment of group key management protocols becomes therefore of primordial importance. A large number of applications have to handle groups composed of different classes or levels related to the roles of members. As such, group key management protocols have been oriented to these hierarchical architectures like [HBBC05]. Conceiving such complex protocols is not straightforward. In fact, such protocols highlight new requirements and consider some complicated intended security properties. Thus, the need to secure them is an important issue. As such, the security architecture established within a group application, should be formally verified in order to detect possible vulnerabilities and consequently to correct them.

Problems for analysing group protocols automatically arise from the fact that most of the verification approaches can only tackle specific models of protocols, and most of the time require the size of the group to be set in advance. The main consequence is the restriction of the chances to discover attacks. Moreover, group membership is very dynamic; participants can join or leave the group at any time. This dynamics makes security requirements more complicated to satisfy. Some work [STW98, Mea00, TJ03, SBM04] have been done to tackle the problem of verification of such dynamic protocols. However, dealing with hierarchical group protocols arises other problems. For instance, as the definition of security properties is related to the notion of level or class, a piece of information should be known only by a sub-group and must be secret for all the rest of the group. To our knowledge, no work of specification and verification of this kind of group protocols has been published so far.

In this paper, we focus on the automatic verification of key management architectures for hierarchical group protocols. We present a study carried out arround an example of hierarchical group protocol, in order to formally verify its key management architecture, and detect the possible security attacks on it. We have chosen to use the back-end AtSe of the AVISPA tool, which provides the possibility of dealing with some requirements, usually used in group protocols such as algebraic operators like XOR or the exponentiation of Diffie-Hellman [DH76].

To presend our results, the paper is structured as follows. In Section 2, we present the security issues in the hierarchical group protocols. An overview of some work done for group protocols verification is provided in Section 3. We also introduce an example of hierarchical group protocol we have chosen to specify and verify in Section 4. The steps for its verification and the limits of this verification are given in Section 5. Finally, we summarise our study and give some directions for future works.

2 Security Issues in Hierarchical Group Protocols

A group security protocol implies the communication between a set of entities, which can be randomly large. Group members have to share a secret key to secure communications between them. This key is called Traffic Encryption Key (TEK). Thus, the management of this group key represents a crucial functionality of such type of protocols.

The TEK is used to ensure confidentiality of data, encrypted by the source and decrypted by the receivers. Moreover, the access control to the group communications is provided, because only members authorized to join the group are able to hold the traffic encryption key, and consequently to reach group communications. The group key management protocol provides also other security services within the group, such as data integrity, authentication of the data flow, ...

A key management protocol has to be adapted to the architecture of the secure group, in order to offer the required security services between the participants of the group communication. We distinguish two main group architectures, described in the next two sections.

2.1 Flat Architecture Groups

This kind of groups often requires a centralized or a distributed group key management protocol. For a centralized protocol such as OFT [MS98] or LKH [WGL98], only one entity called global controller, is responsible for the generation and the distribution of the group key to its members. At any join or leave event in the group, the group key has to be renewed for all members of the group. This phenomenon called ”1 affects n”, presents a weakness in terms of scalability and communication cost.

Within a distributed group key management protocol such as [ITW82, KPT00], the control of the group is under the responsibility of all the group members, who cooperate and collaborate to ensure secure communications between them. The group key is computed as a combination of the contributions of all the group members. Therefore, the key generation process suffers from a lack of scalability, and causes an overhead in terms of communication and computations cost.

Secure key management protocols established within flat architectural group are generally lacking scalability, and suffer from expensive overheads in both communication and computation. To attenuate these problems, a clustering process is needed in order to delegate key management responsibilities to special nodes in the group, as it is usually carried out in decentralized architecture groups.

2.2 Decentralized Architecture Groups

They generally require a decentralized group key management protocol, taking into account the members roles within the group. A decentralized architecture can be obtained by sub-dividing the group into clusters, or into hierarchical classes.

2.2.1 Group Sub-dividing into Clusters

Each created cluster is managed by a local controller sharing with its local members a local secret key. There is no hierarchical relationship between the clusters. The local controllers are managed by the global

controller of the group, which is responsible for their election and the delegation of the management of their respective clusters to them. The clustering mechanism ensures that the dynamics of a cluster does not influence the others group clusters. Thus, a renewal of a local key within a cluster affects only the members of this cluster, and not all members of the group. Moreover, the global controller can delegate the access control to the local controllers, who will be able to authorize or not members to join the group.

The transmission of the secure data within a clusterized group is carried out of different manners, according to the security policies established within the group. Each cluster can hold its local traffic encryption key, requiring two operations of data decryption and re-encryption while passing from a cluster to another.

TSUDIK AND AL. [TRP05] propose a group key management protocol, within UAV-MBN networks. In these networks, there are three types of nodes. The ground nodes include both regular ground nodes and MBN nodes (Mobile Backbone Network). Regular ground nodes are typically soldiers or agents equipped with computation and communication limited devices. They communicate through bandwidth-constraint short-range broadcast wireless channels.
MBN nodes have more capacities than ground nodes. They form with the regular ground nodes, a hierarchical clusterized ad hoc network, the clusterheads being the MBN nodes.
The UAV Nodes (Unmanned Aerial Vehicles) lead the area theaters containing the ground nodes, while ensuring the connectivity between the different MBNs. The group key management protocol architecture in [TRP05] is therefore composed of two levels, the first one is the group cells containing the ground nodes, the second one is the control group represented by the MBN nodes. Each MBN node establishes a centralized group key management protocol within its cluster (OFT [MS98]), while a distributed and cooperative group key management protocol is carried out between the MBN nodes (TGDH [KPT00]).

Others protocols such as [BCF05, DMS99] adopt only one traffic encryption key for all group members. The local clusters keys ensure the role of key encryption keys, and consequently the secure distribution of the TEK to the group members. The group key management protocol BALADE [BCF05, BLCF04] belongs to this approach. BALADE is dedicated to operate in ad hoc networks. The multicast flow is encrypted by the source using the traffic encryption key TEK, and forwarded to the group members through the established multicast tree. The basic idea of BALADE is to clusterize the multicast group, dynamically, into sub-groups. Each one is managed and controlled by a local controller which shares with its local members a local cluster key. The local controllers form a multicast group, and share a key encryption key, in order to ensure a secure distribution of the group TEK.
The architecture of BALADE is composed of three specific nodes: the global controller which is represented by the source responsible for the generation and the distribution of the TEK, the local controllers group responsible for the secure TEK forwarding, and the group members.

2.2.2 Group Sub-dividing into Classes

This sub-dividing consists in assembling group members into classes, with different levels. A class level represents the function or the degree of its members. The classification is thus closely related to the type of the application to be secured. Within a class, a controller responsible for the management and the control of its class members is elected. Each class c holds a traffic encryption key TEK_c, generated by the class controller or by the global controller of the group (who is the controller of the first group class).
According to the security policies of the group, members of class i should be able to decrypt only communications of their class, or of the lower classes. The key management within the group must take into account this assumption, to provide an efficient keys distribution process, in terms of bandwidth and computation power.

HI-KD [HBBC05], a hash-based hierarchical key distribution protocol for group communication, illustrates this approach. Within this protocol, group members are assembled into several classes. Members in class i share a secret key k_i. Each member can communicate with members of its class (intra class communications), or with members belonging to the others classes (inter classes communications).
The hierarchical confidentiality model is ensured via a list of keys established as follows. Based on a randomly generated key, called k_1, corresponding to the first class key, a list of keys is generated through a one-way hash function, such that: k_{c+1}=H(k_c), k_c being the key of the class c. Members in class i can therefore generate the keys of the lower classes j ($j >= i$), and access to their secure communications.

3 Verification of Group Protocols

Research in formal verification of security of two and three party protocols has been ongoing for a number of years [CJ97]. It has given very successful results in the last years. Nowadays, there is a renewed interest for modelling and verifying other kinds of protocols that are more complex in the sense that they highlight new requirements (dealing with unbounded number of participants...) and consider some complicated intended security properties. As such, the problem of analysing group protocols [STW98, Mea00, TJ03, SBM04] has gained increasing importance. Significant attacks on such protocols have been found. Several analysis methods have been used in this task, varying from manual techniques to automatic ones.

One of the most interesting techniques done by hand is suggested by Pereira and Quisquater in [PQ03]. They have introduced a method converting the problem of ownership of some information by the intruder to a problem of resolution of a system of linear equations. With this method, several attacks have been discovered in the protocols suite CLIQUES [AST00]. In these attacks, the intruder can have an imaginative and free behaviour. For example, it can take part to a protocol session, and eavesdrop some information to use it in another protocol session in order to obtain some secret. This method has also permitted to get a generic result: it is impossible to design an authentication group key agreement protocol built on A-GDH for a number of participants greater than or equal to four [PQ04].

Nam, Kim and Won [NKW04] show that the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme [BCEP04] does not meet the main security properties. The analysis of the protocol GKE.Setup (one of the scheme's protocols) leads to three attacks. The first attack is a violation of the property of implicit authentication. The second one is about forward secrecy and the last one is a known keys attack. An improved version of the protocol has then been proposed.

Some automatic tools have been extended with the purpose of dealing with group protocols and their specific requirements. Several attacks have been found. For example in [TJ03], Taghdiri and Jackson have modeled a multicast group key management protocol proposed by Tanaka and Sato [TS01]. They have been able to discover counterexamples to supposed properties. They have then proposed an improved protocol. However, in their model, no active attacker was included. Their improved protocol has been analysed in [SB04] by CORAL and two serious attacks have been found. CORAL has also been used to discover other attacks concerning two protocols: Asokan-Ginzboorg [AG00] and Iolus [Mit97].

Other works have tackled the complexity due to the exploitation of an infinite search space. This problem, often met in group protocols, arises from the fact that even one legal execution of the protocol requires an unlimited number of steps. Meadows [MS01] has extended the NRL protocol analyser in order to tackle the GDOI's protocols. Although the Diffie-Hellman exponentiation has been introduced in this tool, Meadows' attempt to rediscover Pereira-Quisquater attacks on the CLIQUES suite has failed [Mea00].

Other automatic tools, intended rather to search for attacks, have been extended to tackle the new requirements used in group protocols: for example algebraic primitives (XOR ...) or the exponentiation often met in extensions of key agreement based on Diffie-Hellman. Among these tools, we find Atse, one of four back-ends used in AVISPA [ABB+05], a tool that has already treated a large number of Internet security protocols.

4 Example of Hierarchical Group Protocols

In this section, we describe the hierarchical group key management protocol that we chose to verify and validate. This protocol is dedicated to operate within PMR (Private Mobile Radio) networks, which allow users groups, equipped with wireless devices, to secure their voice, data or multimedia communications. This group key management protocol, which is an instanciation of the Hi-KD protocol [HBBC05], was designed in the context of the RNRT SAFECAST project†, in collaboration with all partners. A group is sub-divided into several hierarchical classes, with different levels, as illustrated in Figure 1. Members of class i use, to ensure secure communications between themselves, a shared key called TEK_i. Among these members, a privileged agent is distinguished: the chief of the class. The chief of the group is thus

† http://www.telecom.gouv.fr/rnrt/rnrt/projects/safecast.htm

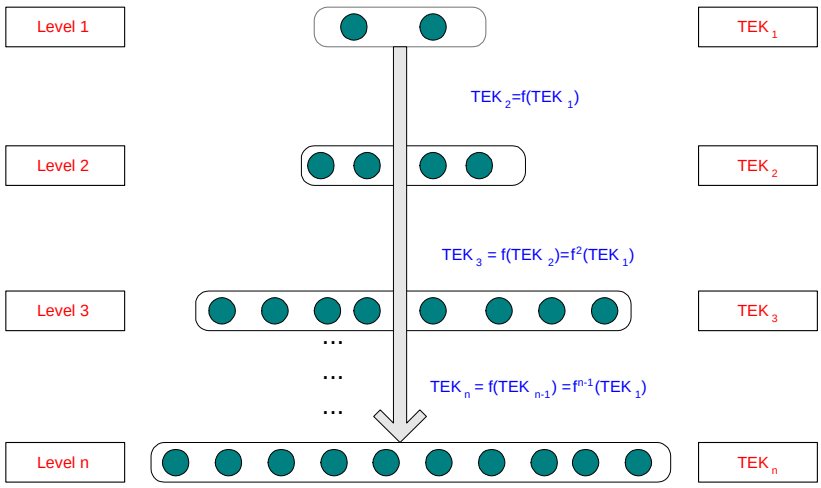

FIG. 1 – *Keys management within PMR networks*

represented by the chief of the first class. Members of class i have access to the secure communications of the lower classes (of level j such that $j <= i$). However, they cannot access to the communications of the upper classes. To do so, the different keys TEK_i must be linked via a one-way hash-function f (cf. Figure 1), as follows:

$$f(TEK_i) = TEK_{i+1}$$

$$f^{n-1}(TEK_1) = TEK_n$$

The keys confidentiality is ensured via the establishment of 8 sub-protocols:

1. **TEK generation and distribution**: Initially, the chiefs of the first class generate the traffic encryption key TEK_1, and distribute it to members of lower classes, encrypted with TEK^{init} keys. In what follows, we present the detailled specification of this protocol and verify and validate it with AVISPA.
2. **Periodic renewal of the TEK**: this is triggered every fixed period of time, and is under the responsibility of the group chiefs. The forward and backward secrecies are not needed in this protocol.
3. **Members Join**: an operational group can require reinforcement from the base of operations, that sends one or more members who must join the group initially formed.
4. **Members re-integration**: this sub-protocol is used when a member looses its connectivity with its group, due to resources or reachability problems, and wants to join it again and access to the communications of its original class.
5. **Members exclusion**: the chief of a group can decide to exclude a member for security purpose. This member cannot join the group any more.
6. **Members promotion**: a member moves from a class i to an upper class j.
7. **Members degradation**: a member moves from a class i to a lower class j, following a request of this member, or an order from a group chief.
8. **Merge of groups**: two groups k1 and k2 can merge into a group k, managed by a chief of both initial groups, after negociation and election.

TEK generation and distribution protocol

Each member of class C_c holds a pre-deployed key TEK_c^{init} (obtained for example at its authentication). c represents the hierarchical level (C_1 is the highest level class and C_n the lowest level class). Nodes in class C_c know the keys TEK_l^{init} $(l >= c)$. The TEK generation and distribution process is carried out as follows:

1. Nodes of C_1 execute Diffie-Hellman (DH) to ensure cooperation and collobaration between them.
2. The last node M in DH computes the TEKs via the relation $TEK_{i+1} = f(TEK_i)$.

3. *M* distributes the TEK_MESSAGE:

$$(\{TEK_1\}_{TEK_1^{init}}, \{TEK_2\}_{TEK_2^{init}}, \ldots, \{TEK_n\}_{TEK_n^{init}})$$

Nodes of the classe C_1 execute DH to generate TEK_1 of level 1. Then, the other TEKs of lower classes will be computed from TEK_1, thanks to the one-way function f. Therefore, a member from a level *c* is able to access data exchanged between members of its class, and of lower classes, and is not able to reach upper levels communications.

The algorithm executed by a node belonging to the class C_c is thus the following:

while receiving TEK_MESSAGE(m_1,m_2,...,m_n) with $m_i = \{TEK_i\}_{TEK_i^{init}}$, i=1,2,...,n:
BEGIN
Decrypt m_c with TEK_c^{init}
END

5 Modeling and Verification

To formally verify the different protocols of the key management in PMR networks previously described in Section 4, we opted for the AVISPA tool [ABB+05] (*Automated Validation of Internet Security Protocols and Applications*), and more precisely for the back-end AtSe. This tool has been chosen since it can express two of the most treated security properties: secrecy and authentication. Besides, it allows to check other requirements such as the exponentiation...

Before using the AVISPA tool, the protocol has first to be specified in HLPSL [CCC+04] (*High Level Protocol Specification Language*). This language is based on roles: basic roles for representing each participant role, and composition roles for representing scenarios of basic roles. Each role is independent from the others, getting some initial information by parameters, communicating with other roles by channels. The HLPSL specification is translated to a low level language named IF (*Intermediate Format*) via a HLPSL2IF translator. The output of this translator is an input to different verification tools: back-ends. We limit ourselves to the back-end AtSe. It provides a translation from a specification written as transition rules in IF, into a set of constraints that is used to find attacks in protocols. Both translation and checking are fully automatic.

5.1 Difficulties of the Verification

The first step of the work consists in raising the ambiguities which exists in specifications written in a natural language. A modeling phase is mandatory before beginning the formal verification, but is a rather delicate step since the result of the formal verification is relative to the confidence of this modeling.

We focus in this section on the protocol of key generation and diffusion presented in Section 4. As previously described, this protocol does not express concretely neither the entities involved nor the messages exchanged nor the initial knowledge of each participant. A second version of the protocol is described by:

$$\begin{array}{l}
(1).\ m_{11k} \longrightarrow m_{21k} : \alpha, \alpha^{r_{11k}} \\
(2).\ m_{21k} \longrightarrow m_{31k} : \alpha^{r_{21k}}, \alpha^{r_{11k}}, \alpha^{r_{11k} r_{21k}} \\
(3).\ m_{31k} \longrightarrow m_{11k} : \alpha^{r_{21k} r_{31k}} \\
(3).\ m_{31k} \longrightarrow m_{21k} : \alpha^{r_{11k} r_{31k}} \\
(4).\ \forall C_{jk} \in G_k \quad \text{such that} \quad j > 1, \forall m_{ijk} \in C_{jk}, \\
m_{31k} \longrightarrow m_{ijk} : \{f^{j-1}(\alpha^{r_{11k} r_{21k} r_{31k}})\}_{TEK_{jk}^{init}}
\end{array}$$

where:

- G_k denotes the *k-th* group;
- C_{jk} denotes the *j-th* class of the *k-th* group;
- m_{ijk} denotes the *i-th* element of the class C_{jk};
- m_{1jk} denotes the Leader of the class C_{jk};

– r_{ijk} denotes a random number generated by m_{ijk}.

Another information is necessary for the verification: the initial knowledge of each entity:

m_{11k}	$\alpha, r_{11k}, TEK_{jk}^{init}$ *and* f.
m_{21k}	$\alpha, r_{21k}, TEK_{jk}^{init}$ *and* f.
m_{31k}	$\alpha, r_{31k}, TEK_{jk}^{init}$ *and* f.
m_{ijk} $j > 1$	TEK_{jk}^{init} *and* f.

¿From this description, we can easily specify the protocol in HLPSL. However, there are still constraints that must be solved. Despite the capability of AtSe to deal with the exponentiation used here in the agreement of Diffie-Hellman, AtSe cannot take into account the diffusion of a message concerning a large number of participants as required by the fourth message of our protocol. Consequently, we have focused only on some instances of the protocol. We opted for three classes (of level 1, 2 and 3). For participants, we considered three entities in the first level and a representative for each other class (2 and 3). Note that thanks to the notion of roles in HLPSL, we are able to easily vary the number of participants in both classes 2 and 3. In fact, for the studied protocols, members of the same class have almost the same behavior concerning messages' reception and emission. As such, this is sufficient to define the representative's role of one class and then to mention the number of roles to be executed in the session. However, by increasing the number of roles, the procedure of attack's search may not complete. In our case, we limited our modeling to two entities per class. The specification of the key generation protocol is given in Section 5.2. The whole of protocols have been checked in a similar way, thus making it possible to manage certificates in some protocols (join,...).

5.2 Specification in HLPSL

In this specification, we specify in HLPSL the key generation and distribution protocol. Firstly, messages between members of the first class are exchanged in order to compute a first class key. Then, a key (relative to a class) is diffused to lower classes. We consider here a group G_1. We assume that we deal with three classes. The first class is composed of two elements. For the two other classes, each one contains one element. We have then four basic roles[‡]: member11, member12, member2, and member3. The two basic roles member11 and member12 are specified as follows:

```
%% TEK Generation with Four Participants

%% First Role of the first Class

role member11 (M111,M211,M121,M131: agent,
           Alpha: nat,
           Snd, Rcv: channel(dy))
     played_by M111 def=

   local State: nat,
         R1: text,
         X1,TEK1: message
   const id1,a_M111_M211_R2,a_M211_M111_R22: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> R1' := new()
           /\ Snd(exp(Alpha,R1'))
           /\ State':=1
           /\ witness(M111,M211,a_M211_M111_R22,exp(Alpha,R1'))

     step2. State=1 /\ Rcv(X1')
       =|> TEK1':= exp(X1',R1)
           /\ State':=2
```

[‡] For more details on syntactic and semantic description of HLPSL specification see [CCC+04]

```
           /\ secret(TEK1',id1,{M111,M211})
           /\ request(M111,M211,a_M111_M211_R2,X1')
end role

%% Second Role of The First Class

role member12 (M111,M211,M121,M131: agent,
               Alpha: nat,
               TEK2init, TEK3init: symmetric_key,
               F: hash_func,
               Snd, Rcv: channel(dy))
     played_by M211 def=

   local State: nat,
         R2: text,
         X2,TEK1: message
   const id1,id2,id3,a_M111_M211_R2,a_M211_M111_R22: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(X2')
       =|> R2':=new()
           /\ TEK1':= exp(X2',R2')
           /\ Snd(exp(Alpha,R2'))
           /\ Snd({F(TEK1')}_TEK2init,{F(F(TEK1'))}_TEK3init)
           /\ State':=1
           /\ secret(TEK1',id1,{M111,M211})
           /\ secret(F(TEK1'),id2,{M111,M211,M121})
           /\ secret(F(F(TEK1')),id3,{M111,M211,M131})
           /\ witness(M211,M111,a_M111_M211_R2,exp(Alpha,R2'))
           /\ request(M211,M111,a_M211_M111_R22,X2')

end role
```

In this specification, the two roles member11 and member12 execute the Diffie-Hellman algorithm and compute the key of the first class TEK1. The last member of the first class (the role member12) diffuses the keys related to the second and third classes, respectively F(TEK1) and F(F(TEK1)).
The third role member2 represents a member of the second class.

```
%% Third Role : A member from The second class

role member2 (M111,M211,M121,M131: agent,
              TEK2init: symmetric_key,
              Snd, Rcv: channel(dy))
     played_by M121 def=

   local State: nat,
         TEK2: symmetric_key
   const id2: protocol_id
   init State:=0
   transition
     step1. State=0 /\ Rcv({TEK2'}_TEK2init)
       =|> State':=1
           /\ secret(TEK2',id2,{M111,M211,M121})

end role
```

The fourth role member3 is specified in a similar way as member2. For studying the protocol we have to define what is a session: it corresponds to a parallel execution of roles member11, member12, member2 and member3.

```
%%The role session between the four participants

role keyGeneration(SC, RC: channel(dy),
                   M111, M211, M121, M131: agent,
```

```
                  F: hash_func,
                  Alpha: nat,
                  TEK2init, TEK3init: symmetric_key
                 ) def=

   composition
     member11(M111,M211,M121,M131,Alpha,SC,RC)
     /\ member12(M111,M211,M121,M131,Alpha,TEK2init,TEK3init,F,SC,RC)

     /\ member2(M111,M211,M121,M131,TEK2init,SC,RC)
     /\ member3(M111,M211,M121,M131,TEK3init,SC,RC)

end role
```

The HLPSL specification is completed by the environment role which is composed of the instances of sessions to be analysed. In the example below, a normal execution is presented (without the participation of the intruder).

```
%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const m111,m211,m121,m131: agent,
         f: hash_func,
         alpha: nat,
         tekin2, tekin3: symmetric_key

   intruder_knowledge = {m111,m211,m121,m131}

   composition
      keyGeneration(Snd,Rcv,m111,m211,m121,m131,f,alpha,tekin2,tekin3)

end role
```

Finally, the security properties to be checked are listed in the goal section:

```
goal
   secrecy_of id1, id2, id3
   authentication_on a_M111_M211_R2
   authentication_on a_M211_M111_R22
end goal
```

In this protocol, the keys' secret is checked according to their class membership. For instance, the key of the first class has to be known only by members of this class (M111 and M211). The key of the second class is known by members of the second class and of the higher classes (here only the first class). Another security property was checked: the authentication between both M111 and M211. M111 must be sure that the contribution of M211 (exp(Alpha,R2')) comes really from M211 and not from anybody else, and the same issue for M211.

5.3 Results and Limits of the Verification

The whole set of protocols described in Section 4, has been specified in HLPSL and then verified by AtSe. We have been able to detect an attack on the members promotion protocol. This protocol, as it was defined above in Section 4, focuses on the case of two parties: the first one m_{1j1} is the leader of the class C_{j1}; the second one m_{kl1} belongs to a lower class C_{l1}, and wants to become member of the class C_{j1}.

m_{kl1} sends his Identity Certificate (*CI*) to m_{1j1}. This certificate is composed of two parties. The first one contains some informations related to the certificate (serial number *sn*, validity period *vp*, ...) and its owner (identity m_{kl1}, public key *pk*, ...). The second part of the certificate is an encryption of the first one, with the private key of the certification authority (*pkca*).

Then, m_{1j1} uses the public key given in the certificate to encrypt the class key TEK_j, and the group membership certificate(*CAp*):

$$(1).\ m_{kl1} \longrightarrow m_{1j1} : CI(m_{kl1})$$
$$(2).\ m_{1j1} \longrightarrow m_{kl1} : \{TEK_j, CAp(m_{kl1})\}_{pk_{m_{kl1}}}$$

As the certificate is modeled by a first part transmitted encrypted, and a second part in clear, the intruder (i) can intercept the first message, get the public key of m_{kl1}, and use it to form a similar message to the one waited by m_{kl1}. The trace of the attack is given in Figure 2.

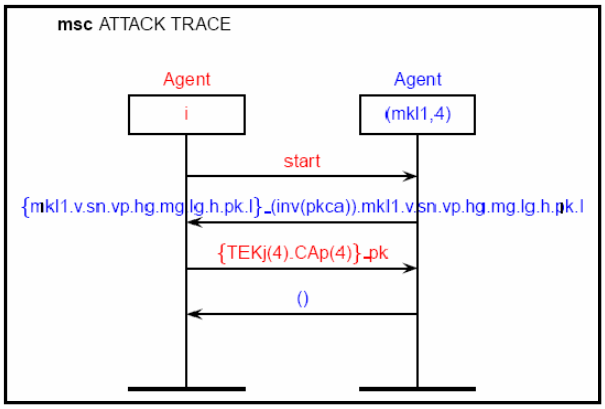

FIG. 2 – *Trace of the attack*

To avoid this attack, a new version of the protocol has been proposed:

$$(1).\ m_{kl1} \longrightarrow m_{1j1} : \{SeqNum_{m_{kl1}}\}_{pk^{-1}(m_{kl1})}, \{Hash1\}_{pk^{-1}(m_{kl1})}, CI(m_{kl1})$$
$$Hash1\ is\ the\ digest\ of\ the\ message\ \{SeqNum_{m_{kl1}}\}_{pk^{-1}(m_{kl1})}$$

$$(2).\ m_{1j1} \longrightarrow m_{kl1} : \{SeqNum_{m_{1j1}}, TEK_j, CAp(m_{kl1})\}_{pk_{m_{kl1}}}, \{Hash2\}_{pk^{-1}(m_{1j1})}, CI(m_{1j1})$$
$$Hash2\ is\ the\ digest\ of\ the\ message\ \{SeqNum_{m_{1j1}}, TEK_j, CAp(m_{kl1})\}_{pk_{m_{kl1}}}$$

The solution consists in adding a signature of the message (2), using the private key of m_{1j1}. Thus, m_{kl1} can authenticate the source of the message, and extract the valid class key TEK_j. A sequence number is introduced in each message, to avoid replay attacks.

The results that we have found have been influenced by several constraints. In fact, only particular instances of protocols have been checked (number of classes in a group, number of participants per class...). Then, each protocol has been verified independently from other protocols. However, eventual attacks may come from an interleaving between messages issued from different sub-protocols managing the same set of participants such as the attack found by CORAL [SB04] on the protocol Tanaka-Sato [TS01]. Thus, it would be more interesting to be able to treat automatically interleavings between the different sub-protocols of a protocol or the different protocols of the same architecture.

6 Summary and Future Works

In this paper, we have presented the specific security issues and vulnerabilities of hierarchical group protocols. We have also discussed related work concerning the verification of such kinds of protocols. We introduced a hierarchical group key management protocol within PMR networks, and its associated sub-protocols. This architecture has been verified using the back-end AtSe of the AVISPA tool, able to deal with the exponentiation of Diffie-Hellman. An attack has been found in the members promotion protocol and a new version has been proposed. We have also checked some scenarios of the BALADE protocol, presented in Section 2.2.1, and verified that no attacks could be found.

Verifying hierarchical group protocols with tools dedicated to attacks' search is a first step to validate them. Indeed, having positive results (no attack found) does not mean that the protocol is correct. We must go through a second step: verifying this protocol with tools dedicated to proofs. Besides, since we have not treated the notion of time in our study, other tools like TURTLE [ACLSS04], should be used for this.

As the need of hierarchical group applications is increasing, automatic verification tools should be extended in order to deal with the new requirements of such complex security protocols. Indeed, they should be able to handle an unbounded number of participants. Besides, they have to tackle some new security properties like integrity, backward or forward secrecies, ...

Références

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

[ACLSS04] L. Apvrille, J.P. Courtiat, C. Lohr, and P. Saqui-Sannes. Turtle: A real time uml profile supported by a formal validation toolkit. *IEEE Transactions on Software Engineering*, 30(7):473–487, july 2004.

[AG00] N. Asokan and P. Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[AST00] G. Ateniese, M. Steiner, and G. Tsudik. New multiparty authentication services and key agreement protocols. *IEEE Journal on Selected Areas in Communications*, 18(4):628–639, 2000.

[BCEP04] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. *Journal of Computer Communications*, 27(17):1730–1737, July 2004. Special Issue on Security and Performance in Wireless and Mobile Networks. Elsevier Science.

[BCF05] M.S. Bouassida, I. Chrisment, and O. Festor. BALADE : Diffusion multicast sécurisée d'un flux multimédia multi-sources séquentielles dans un environnement ad hoc. In R. Castanet, editor, *CFIP 2005*, Hermès Lavoisier, pages 531–546, Bordeaux France, March 2005.

[BLCF04] M.S. Bouassida, A. Lahmadi, I. Chrisment, and O. Festor. Diffusion multicast sécurisée dans un environnement ad-hoc (1 vers n séquentiel). Rapport de recherche 5310, INRIA, September 2004.

[CCC+04] Y. Chevalier, L. Compagna, J. Cuellar, P-H. Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Workshop on Specification and Automated Processing of Security Requirements (SAPS)*. 2004.

[CJ97] J. A. Clark and J. L. Jacob. A survey of authentication protocol literature. Technical Report 1.0, 1997.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *"IEEE Transactions on Information Theory"*, 22(6):644–654, 1976.

[DMS99] L. Dondeti, S. Mukherjee, and A. Samal. Secure one-to-many group communication using dual encryption. In *Computer Communicatio mun 23*, november 1999.

[HBBC05] H. Hassan, A. Bouabdallah, H. Bettahar, and Y. Challal. HI-KD: Hash-Based Hierarchical Key Distribution for Group Communication - IEEE Infocom Poster, 2005.

[ITW82] I. Ingemarson, D. Tang, and C. Wong. A conference key distribution system. In *IEEE transactions on information theory*, September 1982.

[KPT00] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 235–244, New York, NY, USA, 2000. ACM Press.

[Mea00] C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In P.Degano, editor, *the First Workshop on Issues in the Theory of Security*, pages 87–92, Geneva, Switzerland, July 2000.

[Mit97] S. Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.

[MS98] D. Mcgrew and A. Sherman. Key establishement in large dynamic groups using one-way functions trees, May 1998.

[MS01] C. Meadows and P. Syverson. Formalizing gdoi group key management requirements in npatrl. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 235–244, New York, USA, 2001. ACM Press.

[NKW04] J. Nam, S. Kim, and D. Won. Attacks on bresson-chevassut-essiari-pointcheval's group key agreement scheme for low-power mobile devices. Cryptology ePrint Archive, Report 2004/251, 2004.

[PQ03] O. Pereira and J.-J. Quisquater. Some attacks upon authenticated group key agreement protocols. *Journal of Computer Security*, 11(4):555–580, 2003.

[PQ04] O. Pereira and J.-J. Quisquater. Generic insecurity of cliques-type authenticated group key agreement protocols. In *CSFW*, pages 16–19, 2004.

[SB04] G. Steel and A. Bundy. Attacking group multicast key management protocols using CORAL. In A. Armando and L. Viganó, editors, *Proceedings of the ARSPA Workshop*, volume 125 of *ENTCS*, pages 125–144, 2004.

[SBM04] G. Steel, A. Bundy, and M. Maidl. Attacking a protocol for group key agreement by refuting incorrect inductive conjectures. In D. Basin and M. Rusinowitch, editors, *Proceedings of the International Joint Conference on Automated Reasoning*, volume 3097 of *LNAI*, pages 137–151, Cork, Ireland, July 2004. Springer-Verlag Heidelberg.

[STW98] M. Steiner, G. Tsudik, and M. Waidner. Cliques: A new approach to group key agreement, 1998.

[TJ03] M. Taghdiri and D. Jackson. A lightweight formal analysis of a multicast key management scheme. In *FORTE*, pages 240–256, 2003.

[TRP05] G. Tsudik, K. Rhee, and Y. Park. A Group Key Management Architecture for Mobile Ad Hoc Wireless Networks. *Journal of Information Science and Enfineering*, 21:415–428, 2005.

[TS01] S. Tanaka and F. Sato. A key distribution and rekeying framework with totally ordered multicast protocols. In *ICOIN*, page 831, 2001.

[WGL98] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *ACM SIGCOMM*, pages 68–79, 1998.