Automatic Verification of Key Management Architecture for Hierarchical Group Protocols

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

Automatic Verification of Key Management Architecture for Hierarchical Group Protocols

Mohamed Salah Bouassida, Najah Chridi, Isabelle Chrisment, Olivier Festor et 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 around hierarchical architecture protocols, like military or public emergency applications. However, conceiving such secure hierarchical protocols is not straightforward. Thus, their verification become a primordial issue in order to avoid the possible security attacks and vulnerabilities. Several attempts have been done to deal with formal verification of group protocols, but, in our knowledge, none of them has handled hierarchical ones.

This paper investigates both specific challenges and security issues of hierarchical security group communications, and an overview of works done for their verification. We have chosen the Back-end Cl-AtSe of AVISPA tool, to verify an example of such protocols, as it enables to deal with the exponentiation of Diffie-Hellman often used in group key management.

Mots-clés: Hierarchical Group Protocols, Security, Automatic Verification, AVISPA Tool

1 Introduction

The last decade saw an emerging increasing of the need of applications requiring an unbounded number of participants. Varying from military applications to public emergency ones, conceiving secure group protocols [MS98, WGL98] has gained increasing importance. In order to be close to the new requirements, a large number of security group protocols has been oriented to hierarchical architectures [TRP05, BCF05, 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 become increasingly of primordial importance. As such, the architecture security 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. Security requirements are then more complicated. Some works [STW98, Mea00, TJ03, SBM04] has been done to tackle the problem of verification of such protocols. However, dealing with hierarchical group protocols arise other problems. For instance, the definition of security properties become related to the notion of level or class: a piece of information should be known only by a sub-group and thus must be secret for all the rest of the group. In our knowledge, no works of specification and verification of this kind of group protocols was undertaken.

In this paper, we focus on this problematic, in the aim to peel the security issues of the 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 Cl-AtSe of 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.

This paper is structured as follows. In section 2, we present the security issues in the hierarchical group protocols. An overview of some works done for group protocols verification is provided in section 3. We also introduce the example of hierarchical group protocol that 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 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 respresents 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 ensured, due to the fact that only members authorized to join the groupe are able to hold the traffic encryption key, and consequently to reach group communications. The group key management protocol provides also others security services within the group, such as data integrity, authentication of the source of the flow, ...

A key management protocol should be adapted to the architecture of the secured group, in order to provide the required security services for a defined type of communications between the participants of the group. We distinguich two group architecures:

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. Within a distributed group key management protocol, the control of the group is the responsibility of all the group members, which cooperate and collaborate to ensure secure communications between them.

2.2 Hierarchical Architecture Groups

They generally require a decentralized group key management protocol, taking into account the hierarchisation of the group members and their respective roles within the group. A hierarchical architecture can be obtained thanks to a clustering mechanism sub-dividing the group into identical clusters with the same level, or classes with different levels.

2.2.1 Group clustering into clusters

Each created cluster is managed by a local controller sharing with its local members a local secret key. 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 dynamicity of a cluster don't influence the others group clusters. Thus, a renewal of a local key within a cluster affects only members of this cluster, and not all the group members. Moreover, the global controller can delegate the access control to the local controllers, which will be able to authorize or not members to join the group.

The diffusion of the secured data within a clusterized group is carried out with different manners, according to the security policies established within th group. Each cluster could 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 levels of nodes. The ground nodes include both regular ground nodes and

MBN nodes. 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 (Mobile Backbone Network) 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 thaters 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 estbalishes a centralized group key management protocol within its cluster (OFT [MS98]) ; while a distributed and cooperatif 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 the 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] belong 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 hierarchical architecture of BALADE is composed of three levels: 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 clustering into different levels classes

This clustering consists on 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. A military group is composed, for example, of the classes : captains, lieutenants, sergeants... Within a class, is elected a controller responsible for the management and the control of its class members. Each class c holds a traffic encryption key TEK_c, generated by the class controller or by the global controller of the group (which is the controller of the first group class).
According to the security policies of the group, members of class i should be able to reach 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, is a member of 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, in Hi-KD, 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 accede to their secure communications.

3 Verification of Group Protocols

Research in formal verification of the security of two and three party protocols has been ongoing for a number of years. It has given so successful interesting results in the last years that this field could be considered as saturated. 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 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. They vary 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, he 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 obtain a generic result: this is impossible to design an authenticated group key agreement protocol built on A-GDH for a number of participants greater than or equal to four [PQ04].

Nam, Kim et Won [NKW04] show that the Bresson-Chevassut-Essiari-Pointcheval's group key agreement scheme [ECEP04] does not meet the main security properties. The analysis of the protocol GKE.Setup (one of the 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 modelled 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 been 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 a legal execution of the protocol requires an unlimited number of steps. Meadows [MS01] has then 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 the Cl-Atse tool, one of four back-Ends used in AVISPA [ABB+05]. A tool that has 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. These networks allow users groups, equipped with wireless devices, to secure their voice, data or multimedia communications.

The group key management protocol is composed of several sub-protocols, ensuring the security of the communications between users within the PMR network. These sub-protocols are defined within the RNRT SAFECAST project†, in partnership with ENST-PARIS, UTC, LAAS, and EADS, in order to manage and to follow the behavior of a group in the course of time.

A group is sub-divided into several hierarchical classes, with different levels, as is illustrated in figure 1.

Members of class i use, to ensure secure communications between them, a shared key called TEK_i. Among these members, a privileged agent is distinguished: the chief of the class . The chief the group is thus 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 could not accede to the communications of

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

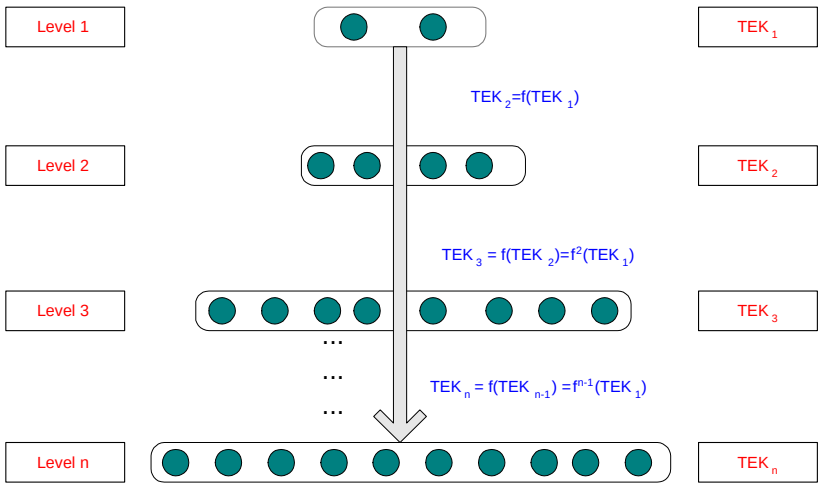

FIG. 1 – *Keys management within PMR networks*

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_0, and distribute it to members of lower classes, encrypted with TEK^{init} keys. In what follows, we present the specification of this protocol. In section 5, we verify and validate it via AVISPA.
2. **Periodic renewal of the TEK**: is triggered each 24 hours, and is the responsibility of the chiefs of the group.
3. **Members Join**: a group on mission can require reinforcement from the base of operations, which sends one or more members who must join the group initially formed.
4. **Members re-integration**: this sub-protocol is used when a member losses its connectivity with its group, due to resources or reachability problems, and wants to re join it and accede to the communications of its original class.
5. **Members exclusion**: the chief of a group can decide to exclude a member when it puts the security of the group in danger. This member will be added to the blacklist, and can not re 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 the two 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:

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

Nodes of the classe C_1 execute DH to generate TEK_1 of level 1. Then, the others TEKs of lower classes will be computed via TEK_1, thanks to the one-way function f. Therefore, a member from a level *c* is able to accede data exchanged between members of its class, and from 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 Modelling and Verification

We aim to formally verify the different protocols of the key management in PMR networks previously described in section 4. We have opted for the AVISPA tool [ABB+05] (*Automated Validation of Internet Security Protocols and Applications*), and more precisely for the Back-end Cl-Atse. This tool has been chosen since it can express two of the most treated security properties: secrecy and authentication. Besides, it permits also to deal with other requirements used in the protocols to be verified such as the exponentiation...

In order to verify a protocol using the AVISPA tool, the protocol has to be first specified in a protocol specification language: 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 Cl-AtSe. Cl-AtSe provides a translation from a specification written as transition relation in the IF, into a set of constraints which can be effectively to find attacks to protocols. Both translation and checking are fully automatic. Cl-AtSe can deal also with algebraic operators such as XOR or also the exponentiation...

5.1 Difficulties of the Verification

Starting from specifications in natural language, the first step of the work consists in raising the ambiguities. Modelling seems then a necessary passage but rather a delicate one since the result of the formal verification is relative to the confidence of this modelisation.

We focus in this section on the protocol of key generation and diffusion presented in section 4. As previously described, this protocol dosn't express concretly neither the entities involved nor the messages exchanged nor the initial knowledge of each participant. A second version of the protocol is as follows:

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

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}.

Athor information is necessary for the verification: 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.

Starting from this description, the protocol is specified in HLPSL. However, there are still constraints that must be solved. For instance, despite the capability of Cl-AtSe to dealing with the exponentiation used here in the agreement of Diffie-Hellman, it can not handle the diffusion for a large number of participants as the case of the 4-*th* message of the protocol. As such, we have focused only on some instances of the protocol. For classes, we have opted for the case of three classes (of level 1, 2 and 3). As for participants, we have considered three entities in the first level and a representative for each other class (2 and 3). It is noticed that thanks to the notion of roles in HLPSL, we were able to 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, it 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 come to an end. In our case, we have been limited 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 for example certificates in protocoles such as the protocol of join…

5.2 Specification in HLPSL

We specify in this section the protocol of key generation and distribution. This protocol describes a messages exchange between members of the first class of the group. Then, a key (relative to a class) is diffused to lower classes. We consider here a group G_1. We suppose we have to deal with three classes. The first class is composed of two elements. As 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: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> R1' := new()
           /\ Snd(exp(Alpha,R1'))
           /\ State':=1
           /\ witness(M111,M211,M211_M111_R22,exp(Alpha,R1'))

     step2. State=1 /\ Rcv(X1')
       =|> TEK1':= exp(X1',R1)
           /\ State':=2
           /\ secret(TEK1',id1,{M111,M211})
           /\ request(M111,M211,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: 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,M111_M211_R2,exp(Alpha,R2'))
           /\ request(M211,M111,M211_M111_R22,X2')

end role
```

In this specification, the first two roles member11 and member12 execute the Diffie-hellman algorithm in order to compute the key of the first class TEK1. It is now for the last member of the first class (the role member12) to diffuse the keys of second and third classes, respectively F(TEK1) and F(F(TEK1)).
The third role: member2 represnts 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). We have to note that it is in this role that the number of participants belonging to lower classes should be varied.

```
%%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 M111_M211_R2
   authentication_on M211_M111_R22
end goal
```

In this protocol, the keys' secret have to be checked according to their belonging to classes. 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 here: the authentication between the two members: M111 and M211. M111 must be sure that the contribution of M211 (exp(Alpha,R2')) comes really from M211 and not from any body else, and the same for M211.

5.3 Results and Limits of the Verification

The whole of protocols of both the architecture proposed in section 4, and the group key management architecture BALADE [BCF05], has been specified in HLPSL and then verified by Cl-AtSe. The result we found for almost all the protocols was positif. Indeed, we were be able to detect an attack on the members promotion protocol. This protocol, as it was defined above in section 6, focus on the case of two parties: the first one m_{1jk} is the leader of the class C_j. The second one m_{lik} belongs to a lower class C_i, and longs to become member of the class C_j.

m_{lik} sends his identity certificate to m_{1jk}, who uses the public key given in the certificate to encrypt the class key TEK_{jk}, and the new generated belonging certificate:

$$\begin{array}{|l|}\hline (1).\ m_{lik} \longrightarrow m_{1jk} : CI(m_{lik}) \\ (2).\ m_{1jk} \longrightarrow m_{lik} : \{TEK_{jk}, CAp(m_{lik})\}_{pk_{m_{lik}}} \\ \hline \end{array}$$

As the certificate is modelled by a first part transmitted in clear, and a second part encrypted, the intruder can intercept the first message, get the public key of m_{lik}, and use it to form a similar message to the one waited by m_{lik}, in order to fool him.

The trace of the attack is given in Figure 2.

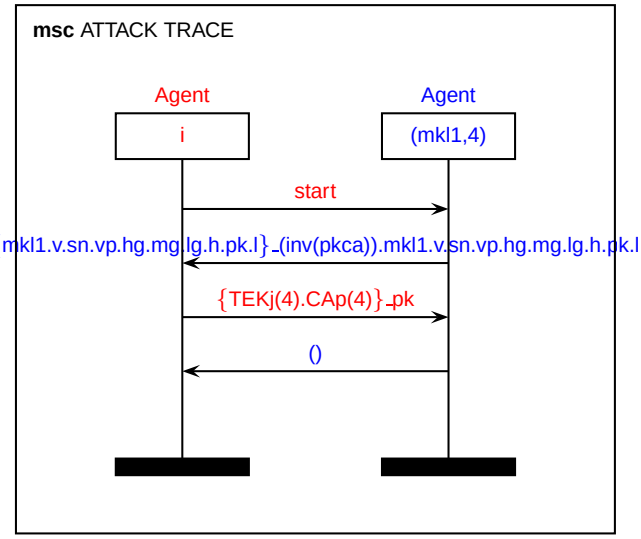

FIG. 2 – *Figure à ajouter*

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

$(1).\ m_{lik} \longrightarrow m_{1jk} : \{SeqNum_{m_{lik}}\}_{pk^{-1}(m_{lik})}, \{Hash1\}_{pk^{-1}(m_{lik})}, CI(m_{lik})$
*Hash*1 *est le condensé du message* $\{SeqNum_{m_{lik}}\}_{pk^{-1}(m_{lik})}$

$(2).\ m_{1jk} \longrightarrow m_{lik} : \{SeqNum_{m_{1jk}}, TEK_{jk}, CAp(m_{lik})\}_{pk_{m_{lik}}}, \{Hash2\}_{pk^{-1}(m_{1jk})}, CI(m_{1jk})$
*Hash*2 *est le condensé du message* $\{SeqNum_{m_{1jk}}, TEK_{jk}, CAp(m_{lik})\}_{pk_{m_{lik}}}$,

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

The results we found were 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] of 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 pinpointed works concerning the verification of such kind of protocols. We have introduced afterwards an hierarchical group key management protocol, within PMR networks, and the different sub-protocols constituting it. This architecture has been verified using the Back-end Cl-AtSe of the

AVISPA tool. We were able to deal with the exponentiation of Diffie-Hellman. An attack was found in the members promotion protocol and a new version is proposed.

Verifying hierarchical group protocols with tools destined for attacks' search is a first step to validate them. Indeed, having positif results (no attack found) does not mean that the protocol is correct. We must go through a second step: verifying this protocol with tools destined to proofs.

As the need of hierarchical group applications is increasing, automatic verification tools should be extended in order to deal with the new requirements in 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 or properties related to time.

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*, Lecture Notes in Computer Science, Edinburgh, Scotland, 2005. Springer.

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

[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 *CFIP 2005*, 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, INRIA, Sep 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)*. 2004.

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

[ECEP04] 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.

[HBBC05] H. Hassan, A. Bouabdallah, H. Bettahar, and Y. Challal. Hi-kd: Hash-based hierarchical key distribution for group communication - ieee infocom poster, 2005.

[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*, number 3097 in 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*, pages 831–, 2001.

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