Some Attacks upon Authenticated Group Key Agreement Protocols

Olivier Pereira, Jean-Jacques Quisquater
UCL Crypto Group - Place du Levant, 3
B-1348 Louvain-la-Neuve, Belgium
{pereira, quisquater}@dice.ucl.ac.be

November 3, 2002

Abstract

During the last few years, a number of authenticated group key agreement protocols have been proposed in the literature. We observed that the efforts in this domain were mostly dedicated to the improvement of their performance in term of bandwidth or computational requirements, but that there were very few systematic studies on their security properties. In this paper, we tried to develop a systematic way to analyse protocol suites extending the Diffie-Hellman key-exchange scheme to a group setting and presented in the context of the Cliques project. This led us to propose a very simple machinery that allowed us to manually pinpoint several unpublished attacks against the main security properties claimed in the definition of these protocols (implicit key agreement, perfect forward secrecy, resistance to known-key attacks).

1 Introduction

An Authenticated Group Key Agreement Protocol (AGKAP for short) is a protocol enabling a group of users to generate a shared secret key on a network possibly controlled by an active attacker. Very often, AGKAP's are defined together with a number of "subprotocols" enabling dynamic changes in group constitution ([2], [6], [11]). Although the experience has shown how complex it is to define security protocols that can be used in the presence of active attackers (see [8], [12] for instance), AGKAP's have rarely been systematically studied until now: only sketch proofs or informal arguments were given to convince of their correctness in their presentation ([2], [11]).

Two types of methods, coming from two little related communities, have been developed for the study of security protocols. "Cryptographic" approaches have been proposed to analyse these protocols: we can notably mention the work of Bellare and Rogaway [3] that has been extended by

Blake-Wilson, Johnson and Menezes [4] in order to enable the handling of the authenticated Diffie-Hellman key exchange. More recently, Bresson et al. [5, 6] further extended these methods for the analysis of AGKAP's. In these approaches, cryptographic operations are considered as functions on bit strings and security properties are expressed in terms of probability and computational complexity of successful attacks. Unfortunately, proofs using such methods are often laborious and do not render pointless the development of analysis methods for reasoning at a higher level of abstraction. In these methods, security properties are formally (logically) modelled and cryptographic operations are viewed as functions on a space of symbolic expressions. These analyses allowed researchers to capture a lot of useful intuitions about security protocols and discover many new flaws by considering only idealized cryptographic primitives and without precisely taking into account the computational capabilities of the intruder (first attempts to bridge the gap between these two views of cryptography can be found in [1] and [10] for example). The methods we are using in this paper can be placed in the second category although they capture arithmetic properties at a level of abstraction lower than the one usually considered in this type of methods. This will be discussed more extensively in section 4.

This paper is organised as follows: we begin in section 2 by examining the main security properties of interest for these protocols. Then, in section 3, we will describe the A-GDH.2 protocol suite [2] whose security is based on the decisional Diffie-Hellman problem. In order to analyse these protocols, we will develop in section 4 a very simple machinery that will allow us to reduce the problem of the verification of security properties to the resolution of a linear equation system. In sections 5 and 6, we analyse two suites of protocols proposed in the context of the Cliques project [2]: the A-GDH.2 and SA-GDH.2 suites. Our analysis will allow us to pinpoint several new attacks against all protocols we analysed.

2 Intended Security Properties

The execution of an AGKAP session allows a set of users M to generate a shared group key. This key must remain secret even in the presence of an active attacker that is able to eavesdrop or intercept messages, as well as to send messages he generated using the information he possesses.

The main desirable security properties for these protocols are the following:

1. *Implicit Key Authentication*

When he completed his role in a protocol session, each $M_i \in M$ is assured that no party $M_q \notin M$ can learn the key $S_n(M_i)$ (i.e. M_i's view of the key) unless helped by a dishonest $M_j \in M$.

This property does not mean that the group members have any knowledge of the group key at the end of the protocol, nor that they agree on its value. Also, it does not imply for a group member that any other member executed a session of the protocol (there is no liveness property intended in the sense of G. Lowe in [13]).

2. *Perfect Forward Secrecy*

This property is defined in [18] as follows: the compromise of long-term keys does not compromise past session keys.

This property, traditionally used in a two parties setting, is exploited in a context where no message was manipulated in the past. This is natural since the two users cannot exchange messages if they do not agree on the value of the key. However, things are slightly different in a multiparty setting: the manipulation of a message can alter the view of the key of one particular user, while all the other group members are computing the same key; and this manipulation does not prevent these last users from communicating.

So, we propose to define two flavours of perfect forward secrecy: *complete forward secrecy* and *individual forward secrecy*. According to the former, no message have been manipulated in the past for any member of the group; but according the latter, one (or a few) members of the group may have been subject to attacks which left unaffected the other members of the group (and thus the protocol was individually correct for each of them). This scenario is plausible since, after executing the protocol, the attacked individuals may just be passive recipients of the messages exchanged by the others.

3. *Resistance to Known-Key Attacks*

This property is defined in [18] as follows: a protocol is said to be vulnerable to a known-key attack if compromise of past session keys allows either a passive adversary to compromise future session keys, or impersonation by an active adversary in the future.

One of the principal motivations for the definition of this property is the protection of future sessions against the compromise of session secrets usually weakly protected than long-term ones.

However, in the protocols we will analyse, the session keys are not the only session secrets: each user generates a contribution to the session key that is also kept secret. So, we think it judicious to consider whether security problems can result in further sessions of the compromise of any secret local to a past session (rather than only of the compromise of session keys).

We will now describe the first AGKAP we will analyse.

3 The A-GDH.2 Protocol Suite

The suite of protocols we will be studying here is the A-GDH.2 suite that has been proposed within the scope of the Cliques project (see [2] for instance). The main A-GDH.2 protocol (that will be referenced as the A-

GDH.2 protocol in the rest of this paper) allows a group of users to agree on a contributively generated key. The other protocols of the suite permit the addition of new members in the group (A-GDH.2-MA), the removal of a member, the fusion of two groups, etc.

All proposed GDH protocols are based on the difficulty of a single problem: the Diffie-Hellman decision (DDH) problem (i.e. given a large integer p and knowing $\alpha^x \bmod p$ and $\alpha^y \bmod p$, it is hard to distinguish $\alpha^{xy} \bmod p$ from a random number). All arithmetic throughout this paper will be performed in a cyclic group G of prime order q which is typically (but not necessarily) a subgroup of $\mathbb{Z}_p^*$ for a prime p such that $p = kq + 1$ for small $k \in \mathbb{N}$ (e.g. $k = 2$).

We assume that p, q and α are public, and that every user M_i shares (or is able to share) with each M_j a distinct secret K_{ij}. For example, we can set $K_{ij} = F(\alpha^{x_i x_j} \bmod p)$ where x_i is a secret long-term exponent selected by every M_i and $\alpha^{x_i} \bmod p$ is the corresponding long-term public key. We will now describe the first two protocols studied in this paper: the Key Generation and the Member Adding protocols (the other protocols of the suite are not described in detail in the literature).

3.1 The A-GDH.2 Protocol

Let $M = \{M_1, \ldots, M_n\}$ be a set of users wishing to share a key S_n. The A-GDH.2 protocol executes in n rounds. In the first stage ($n-1$ rounds), contributions are collected from individual group members and then, in the second stage (n-th round), the group keying material is broadcast. The actual protocol is as follows:

Initialization:
Let p be a prime integer and q a prime divisor of $p-1$. Let G be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and let α be a generator of G.

Round i $(0 < i < n)$:

1. M_i selects $r_i \in \mathbb{Z}_q^*$
2. $M_i \rightarrow M_{i+1} : \{\alpha^{\frac{r_1 \ldots r_i}{r_j}} | j \in [1, i]\}$, $\alpha^{r_1 \ldots r_i}$

Round n:

1. M_n (that will be called the *group controller*) selects $r_n \in \mathbb{Z}_q^*$
2. $M_n \rightarrow$ All M_i: $\{\alpha^{\frac{r_1 \ldots r_n}{r_i}.K_{in}} | i \in [1, n[\}$

Upon receipt of the above, every M_i computes the group key as:

$$S_n = \alpha^{(\frac{r_1 \ldots r_n}{r_i}.K_{in}).K_{in}^{-1}.r_i} = \alpha^{r_1 \ldots r_n}$$

This protocol is intended to provide the security properties described in section 2 excepted the Resistance to Known-Key Attacks where implausible attack schemes are suggested. In order to better understand its structure, we represented a protocol run with four participants in Fig. 1.

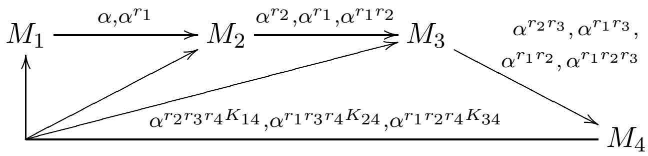


Figure 1: A-GDH.2 Protocol Run with 4 Participants

From this figure, we can observe that the behaviour of the participants is quite repetitive. If we except the first one, each group member M_i receives several elements of G, exponentiate them with a random contribution, and forwards the result to the next group member. The group controller (i.e. the last group member, M_4 in our example) performs two operations more: he keeps the last exponentiated element to compute his view of the key and exponentiates the others with the long-term keys he shares with the $n-1$ first group members. When receiving the final broadcast, these users compute their view of the group key by exponentiating anew an element of this message with a product of keys and random numbers.

3.2 The A-GDH.2-MA Protocol

Let $M = M_1, \ldots, M_n$ be a set of users sharing a key S_n and assume that M_{n+1} is wishing to join the group. The A-GDH.2-MA protocol executes in 2 rounds: in the first one, M_n sends to M_{n+1} a message computed from the one he broadcast in the last round of the A-GDH.2 protocol and from the old key while in the second round, M_{n+1} broadcasts the new keying material to the group. The actual protocol is as follows:

Round 1:

1. M_n selects $\hat{r}_n \in \mathbb{Z}_q^*$
2. $M_n \rightarrow M_{n+1}$: $\{\alpha^{\hat{r}_n \frac{r_1 \ldots r_n}{r_i} K_{in}} | i \in [1, n]\}$, $\alpha^{\hat{r}_n r_1 \ldots r_n}$

Round 2:

1. M_{n+1} selects $r_{n+1} \in \mathbb{Z}_q^*$
2. $M_{n+1} \rightarrow$ All M_i: $\{\alpha^{\hat{r}_n \frac{r_1 \cdots r_{n+1}}{r_i} . K_{in} . K_{in+1}} | i \in [1, n+1]\}$

Upon receipt of the above, every M_i computes the new group key as:

$$\begin{aligned} S_{n+1} &= \alpha^{(\hat{r}_n \frac{r_1 \cdots r_{n+1}}{r_i} . K_{in} . K_{in+1}) . K_{in}^{-1} . K_{in+1}^{-1} . r_i} \\ &= \alpha^{\hat{r}_n r_1 \ldots r_{n+1}} \end{aligned}$$

The security properties of the A-GDH.2 protocol have to be preserved after execution of this protocol. Fig. 2 represents a protocol run where a fifth member is added to the previously represented group.

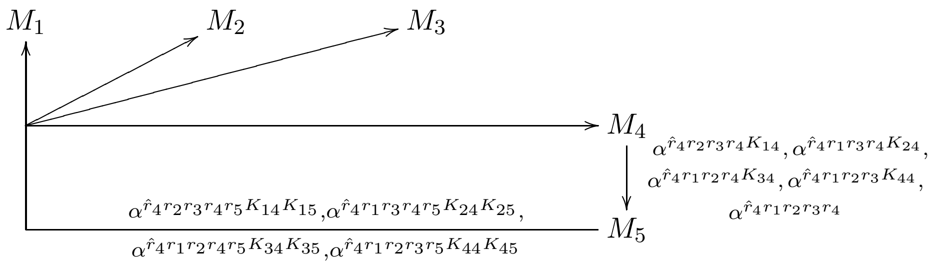


Figure 2: A-GDH.2-MA Protocol Run

We can observe on this figure that the first three elements of the first round of this protocol are those of the broadcast of the setup protocol exponentiated with $\hat{r}_4$.

4 A Model for the Analysis of the Cliques Protocols

4.1 Introduction

A number of methods were developed during the last few years for the formal (or logical) analysis of security protocols. Many of them are based on state-space exploration: they usually proceed by defining an arbitrarily bounded system and exploring it hoping (or after having proved) that if there is an error in the protocol, it can be described by a behaviour included in the considered state-space ([9], [14], [15], ...). However several tools allow proofs to be obtained for unbounded systems at the cost of the interactive proof of several lemmas [16] or of the risk of receiving no answer for some protocols [21]. Other approaches are based on the use of logics ([20], [22], ...). They allow proofs to be obtained for arbitrary size configurations, but they often require particularly error-prone formalization steps and do not provide the same support in pinpointing problems as the direct generation of counter examples. Recently, "manual" approaches were presented, allowing fine-grained proofs to be obtained for systems containing an unbounded number of occurrences of each defined roles (see [23] for example). In order to make such proofs feasible, several simplifying assumptions are typically stated: a very limited set of cryptographic primitives is considered (typically public-key and symmetric-key encryption), and these primitives are usually idealized in such a way that they act as black-boxes (ignoring low-level properties such as the multiplicative structure of RSA or the characteristics of the chaining method used in symmetric-key encryption for example). The use of state-space exploration techniques in the study of group protocols seems very difficult due to their very essence: the number of participants in a honest session of the protocol is basically unbounded. This will intuitively result in dramatic state-space explosion problems. As far as we know, the only successful analyses of group protocols have been performed by theorem proving approaches ([7], [19]), which allow inductive reasoning. Besides this, C. Meadows has performed (independently of us) the analysis of the A-GDH.2 protocol, adapting her NRL Protocol analyzer by extending the power and scope of its theorem-proving capabilities [17]. Beyond the problem of the unbounded number of participants in the protocols, the modelling of the A-GDH.2 protocols suite requires to capture several low-level arithmetic properties: exponentiation, commutativity, associativity, that are out of the scope of most of the works encountered in the literature. Furthermore, the A-GDH.2 key generation protocol is not intended to be used alone: there are several other protocols in the suite (member addition, ...) that use values computed during the key generation protocol and can interfere with its security properties.

In the next paragraphs, we first introduce the way we model the exchanged messages, then we describe the intruder capabilities and, finally, we show how the intended security properties can be verified.

4.2 Messages and Intruder's Knowledge

The messages sent during the execution of the protocols we are analysing are constituted by the concatenation of elements of a group G of prime order q. A particular element, that we will denote α, is a generator of G and is shared by all users of the network (as well as the knowledge of the characteristics of the group G). All exchanged elements of G are expressed as powers of $\alpha \bmod p$. It can then be checked that the participants have to manipulate three types of elements:

- Random Numbers (r_i)
- Long-term Keys (K_{ij})
- Elements of G expressed as α raised to the power of a product of random numbers and long-term keys.

The behaviour of the honest participants is quite simple: they receive elements of G, exponentiate them with random numbers and/or long-term keys (possibly inverted), and send the result to other participants. The group-key is obtained in the same manner, except that the result of the computations is not sent but kept secret. It can be noticed that when a participant receives an element of G, he has to accept it without being able to check anything concerning its constitution or origin. The goal of an intruder will therefore be to possess a pair (α^x, α^y) of elements of G related between them in such a way that α^y is equal to the result of the key-computation operation of a honest M_i applied to α^x. If we take this point of view, there are n secret pairs corresponding to an execution of the protocol between n parties.

Example: If we refer to Fig. 1, the goal of the intruder who wants to fool M_1 is to obtain a pair $(\alpha^x, \alpha^{x r_1 K_{14}^{-1}})$ for some x: if he obtains this pair, he can replace the term $\alpha^{r_2 r_3 r_4 K_{14}}$ with α^x and M_1 will compute $\alpha^{x r_1 K_{14}^{-1}}$ as the group key.

Since all properties of the protocols we are analysing can be described in terms of relation between elements of G, our model will not deal with elements of G but with the possible relations between pairs of them: the pair (α^x, α^y) will be modeled as the ratio y/x.

More precisely, our model considers the manipulation of two sets of elements:

- The set E containing the random numbers r_i and the long-term keys K_{ij}
- The set $R = \{\prod r_i^{e_i} \prod K_{jl}^{e_{jl}} | e_i, e_{jl} \in \mathbb{Z}\}$ that we will use to represent the relation between elements of G: $r \in R$ will represent the pairs of elements of G that can be written as (α^x, α^{rx}) for any value of x.

Example: With our notations, the secret pair corresponding to the secret of M_1 in Fig. 1 will be represented by the element $\frac{x r_1 K_{14}^{-1}}{x} = r_1 K_{14}^{-1} \in R$.

The use of such a construction will be quite convenient and can be intuitively justified if we state a hypothesis that is quite similar to the widely used "perfect encryption assumption". We will refer to this hypothesis as the "perfect Diffie-Hellman assumption" and it can be stated as follows:

> "An element of G can be computed in one and only way: by exponentiating the generator α with the correct random numbers and keys (excepted the permutations in the order of the exponentiation of α and the possibility of exponentiation by an element of E and by its inverse successively)."

This assumption implies in particular that a secret element of G cannot be computed by combining other elements of G (but only elements of E with elements of G). It seems quite plausible in practice given that we work within a large group (lucky guesses or collisions are very unlikely) and that the DDH problem is hard.

It can also be noticed that the use of the R-set implies another restriction due to its very structure: it does not allow capturing relations between more than two elements of G. However, in the presence of our perfect Diffie-Hellman assumption, the relevant security properties always come down to the impossibility of finding two elements of G presenting between them a particular relation, so that the consideration of more complex relations cannot be of any help to prove the correctness of Cliques protocols. It could be useful to use such extensions to discover more dangerous attacks that violate more than one security property, but we are more interested in checking the correctness of protocols than finding "optimal" attack sketches.

We will now be looking at the ways that the intruder can use to manipulate our two sets of elements.

4.3 Intruder Capabilities

Considering our "perfect Diffie-Hellman assumption", the only useful computation for the intruder will be the exponentiation of an element of G by a known element of E. If we note E_I and R_I the subsets of elements of E and R (respectively) that are known by the intruder, we can then transpose this remark as follows:

$$\begin{array}{c}(1) \text{ If } e \in E_I \text{ and } r \in R_I \text{ then} \\ r.e \in R_I \text{ and } r.e^{-1} \in R_I\end{array}$$

There is another way for the intruder to obtain new elements of R: the use of the computations executed by the honest users. As we said above, the behaviour of these users is quite simple: they receive elements of G and exponentiate them with some values of E. We will call such operations *services*. More precisely, a service is a function $s : G \to G : \alpha^x \to \alpha^{p \cdot x}$, and we call S the set of available services. For simplicity purposes, we will refer to the service $s(\alpha^x) \to \alpha^{xp}$ by the power it raises its input: $p \in S$. Let us see how a service can be described in term of growth of R_I. If $r \in R_I$, then the intruder possesses two elements of G that can be written α^x and α^{rx}. If the intruder sends α^x to a honest user performing the service p, then he will learn the element $p^{-1}.r \in R_I$. Conversely, if the intruder sends α^{rx} to the user performing the same service, he will learn the element $p.r \in R_I$. We can then write our second rule for the increasing of the R_I-set:

$$\begin{array}{c}(2) \text{ If } p \in S, \text{ and } r \in R_I \text{ then} \\ r.p \in R_I \text{ and } r.p^{-1} \in R_I\end{array}$$

Nevertheless we have to be careful in the use of this rule and impose some restrictions in its application due to the fact that honest users provide several services in parallel but only once. This will be examined more in the detail in the next section where we will propose a method to determine if a ratio is secret or can be obtained by the intruder.

4.4 Proving Security Properties

In the context of the Cliques protocols, the most general message transformation provided by a user during a single round can be written as follows:

$$\alpha^{x_1} \dots \alpha^{x_n} \to \alpha^{y_1} \dots \alpha^{y_m} \alpha^{x_1 y_{11}} \alpha^{x_1 y_{12}} \dots \alpha^{x_n y_{n1}} \dots \alpha^{x_n y_{np}}$$

This view can be used to express the rules limiting the composition of services in the derivation of the set R_I:

- The (2)-rule can be used at most once for each service.
- If two services p_1 and p_2 are offered during the same round and take distinct inputs (i.e. are applied to distinct α^{x_i}), then they can be used on a single element $r \in R_I$ to produce the following elements: $r.p_1$, $r.p_2$, $r.p_1^{-1}$, $r.p_2^{-1}$, $r.p_1^{-1}.p_2$, $r.p_1.p_2^{-1}$ (but not $r.p_1.p_2$ nor $r.p_1^{-1}.p_2^{-1}$)
- If two services p_1 and p_2 are offered during the same round and take the same inputs (i.e. are applied to the same α^{x_i}), then they can be used to produce the following elements: p_1, p_2, $p_1^{-1}.p_2$ and $p_1.p_2^{-1}$. It can be noticed that these elements are independent from any previously known element of R_I.
- If two services p_1 and p_2 are offered independently of any input (as α^{y_i} in our example), then they can be used to obtain p_1, p_2, $p_1^{-1}.p_2$ and $p_1.p_2^{-1}$. This restriction has to be observed even if p_1 and p_2 are provided during different rounds of the protocol.

From these considerations, we can suggest a systematic scheme to obtain the proof of the secrecy of a particular $r \in R$, i.e. to prove that some element of R cannot be obtained from the initial knowledge of the intruder by using the two rules we defined in the previous section.

1. *Definition of the system*

At first, we define the system we will analyse: we fix the sessions of the protocol we consider and the intended roles for each user. This allows us to define the available services (S-set), the atomic elements initially known by the intruder (E_I-set), and the secret ratios (let R_S be this set).

Example: Let us consider the system containing the session represented in Fig. 1 and assume that r_3 is compromised (because of the compromising of the pseudo-random generator of M_3 for example).

During the first three rounds, the user M_i provides the service $r_i \in S$ several times in parallel. During the fourth round, M_4 provides 3 services: $r_4 K_{14}$, $r_4 K_{24}$ and $r_4 K_{34}$. The secret of M_i ($1 \leq i < 4$) is $r_i K_{i4}^{-1}$ while the secret of M_4 is r_4. So, to summarize, the sets we will consider are the following:

$$
\begin{aligned}
S &= \{r_1, r_2, r_3, r_4K_{14}, r_4K_{24}, r_4K_{34}\} \\
E_I &= \{r_3\} \\
R_S &= \{r_1K_{14}^{-1}, r_2K_{24}^{-1}, r_3K_{34}^{-1}, r_4\}
\end{aligned}
$$

The service r_2 can be seen in two ways: we will assume in the rest of this paper that α^{r_2} is computed by exponentiating the term α that M_2 receives from M_1 (without checking the correctness of this value) rather than by generating a new term. Assuming this last way of executing the protocol should only change minor details in the further developments (i.e. some attacks that can be performed against $n-1$ group members will only be applicable against $n-2$ group members).

2. *Use of the* (1)*-rule*

All elements corresponding to those of E_I can be deleted from the expression of S and R_S. This operation simplifies the problem and does not change its solutions since:

- If $e \in E_I$, every operation that uses the service $e^i y \in S$ $(i \in \mathbb{Z})$ can be performed by using a service y and by suitably applying the (1)-rule.
- If $e \in E_I$, and $r.e^a \in R_S$ then the knowledge of r implies the one of $r.e^a$ (anew by applying (1)-rule).

We will call the resulting sets S^1 and R^1_S.

Example: Applying this step to the sets obtained in the previous example provides:

$$
\begin{aligned}
S^1 &= \{r_1, r_2, r_4K_{14}, r_4K_{24}, r_4K_{34}\} \\
R^1_S &= \{r_1K_{14}^{-1}, r_2K_{24}^{-1}, K_{34}^{-1}, r_4\}
\end{aligned}
$$

We can observe that the service r_3 completely disappeared. The reason is that that service is not useful anymore if r_3 is known by the intruder.

3. *Use of the* (2)*-rule*

The problem is now to determine whether some element of R^1_S can be obtained from S^1 by suitably exploiting the (2)-rule.

The secrecy of a particular element of R_S can be determined by writing a linear system expressing the "balance" of the variables in the construction of the secret from the services. This system contains one variable per element of S^1 and one equation per element in E. On the left side of the equation, the value of each element of the system is the power of the element of E corresponding to the line in the element of S^1 corresponding to the column. The second term of each equation is the power of the element of E corresponding to the line in the studied secret.

This system expresses that the only way to compute the secret is to successively apply some services starting form the only initially known ratio: 1. If this system is inconsistent, then the intended confidentiality property is verified (in our model, and for the considered system). If this is not the case, we have to check the restrictions on the use of services described above and verify whether the initially known ratios are not combined in an inconsistent way. If these conditions are satisfied, then an attack scheme can be constructed by exploiting the known elements of E and the offered services as indicated by the solution(s) of the linear system. This construction will be exemplified in section 5.1.2.

Example: If we apply this last step on the previous example, the linear system corresponding to the verification of the secrecy of $r_1 K_{14}^{-1}$ is:

$$\begin{array}{c} \\ r_1 \\ r_2 \\ r_4 \\ K_{14} \\ K_{24} \\ K_{34} \end{array}
\begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_4K_{14} & r_4K_{24} & r_4K_{34} \end{array} \\
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}
\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

It can be observed that this linear system has no solution since the last four equations are inconsistent. So, we are not able to obtain the secret $r_1 K_{14}^{-1}$ from the studied system.

We will now see how this scheme can be applied for the analysis of the A-GDH.2 protocols suite.

5 Analysis of the A-GDH.2 Protocols

In the following paragraphs, we will analyse the three security properties of the A-GDH.2 key generation protocol described in section 2. Then, we will extend our analysis to the consideration of the concurrent use of the A-GDH.2-MA protocol.

5.1 Implicit Key Authentication

We will divide our analysis in two parts: in the first one we will assume that the intruder is not a member of any group, while in the other we will consider that the intruder is a legitimate member of some groups.

5.1.1 IKA when the intruder is excluded from all groups

As described in the previous section, the first step in our analysis will be the description of the protocol.

Initially, we will consider only one session of the protocol with n participants. The intruder knowing no long-term keys nor any short-term secrets, the E_I-set is to be empty. From the first to the $(n-1)$-th round, the user M_i provides the service $r_i \in S$ several times in parallel. During the n-th round, M_n provides the $n-1$ services: $r_n K_{1n}, \ldots, r_n K_{n-1n}$. The secrets are the following: $r_i K_{in}^{-1}$ for M_i $(1 \leq i < n)$ and r_n for M_n. So, to summarize, the sets we consider are the following:

$$\begin{aligned} S &= \{r_1, r_2, \ldots, r_{n-1}, r_n K_{1n}, \ldots, r_n K_{n-1n}\} \\ E_I &= \emptyset \\ R_S &= \{r_1 K_{1n}^{-1}, \ldots, r_{n-1} K_{n-1n}^{-1}, r_n\} \end{aligned}$$

If we follow the analysis scheme proposed above, we now have to express the linear system describing the "balance" of the variables of E to check the secrecy of the elements of R_S. We will first look at the secrecy of $r_1 K_{1n}^{-1}$ (so we need the balance on r_1 and K_{1n} being equal to 1 and -1 respectively while the balance on the other variables must be null). If we use the "s"-letter to denote the variable indicating how many times the service s has to be used to construct the secret, the linear system can be written as follows:

$$\begin{array}{ll} r_1 = 1 & \text{Balance on } r_1 \\ r_i = 0 & \text{Balance on } r_i \ (2 \leq i < n) \\ \sum_{i=1}^{n-1} r_n K_{in} = 0 & \text{Balance on } r_n \\ r_n K_{1n} = -1 & \text{Balance on } K_{1n} \\ r_n K_{in} = 0 & \text{Balance on } K_{in} \ (2 \leq i < n) \end{array}$$

It can be observed that summing the $n-1$ equations corresponding to the balance on the keys provides an inconsistency with the equation expressing the balance on r_n. Hence we can say that $r_1 K_{1n}^{-1}$ cannot be obtained by using the two enrichment rules we defined and $S_n(M_1)$ is kept secret in our model as claimed in the protocol definition. If we write this system in the case of multiple sessions of the protocol (from which the intruder is excluded), it can easily be checked that this inconsistency is preserved. The transposition of this result for the $r_i K_{in}^{-1}$ -secrets is straightforward and if we transform the second members of these equations in order to prove the secrecy of r_n, we can easily obtain an inconsistency between the same equations. We can then say that the Implicit Key Authentication property is correct with respect to our model provided that the intruder is not a member of any group.

5.1.2 IKA when the intruder is a legitimate member of some groups

We will now check if this property is preserved when the intruder is a member of some groups. As a simple scenario, we will assume a first session of the protocol in which M_1, M_2, M_I and M_3 are the intended participants (M_3

being the group controller), and a second session with the same participants excepted M_I. We will note r'_i the random contribution generated by M_i during this last session. So, the sets of interest become:

$$\begin{aligned} S &= \{r_1, r_2, r_I, r_3K_{13}, r_3K_{23}, r_3K_{I3}, \\ &\quad r'_1, r'_2, r'_3K_{13}, r'_3K_{23}\} \\ E_I &= \{r_I, K_{I3}\} \\ R_S &= \{r'_1K_{13}^{-1}, r'_2K_{23}^{-1}, r'_3\} \end{aligned}$$

The application of the second step of our proof scheme provides the following sets:

$$\begin{aligned} S^1 &= \{r_1, r_2, r_3K_{13}, r_3K_{23}, r_3, \\ &\quad r'_1, r'_2, r'_3K_{13}, r'_3K_{23}\} \\ R_S^1 &= \{r'_1K_{13}^{-1}, r'_2K_{23}^{-1}, r'_3\} \end{aligned}$$

If we solve the three corresponding linear systems (i.e. one system per element of R_S), a number of solutions can be found.

We will give the example of the verification of the secrecy of $r'_2K_{23}^{-1}$. The linear system corresponding to this secret is the following one:

$$\begin{array}{rr} r_1 = r'_1 = r_2 = 0 & r'_2 = \ \ 1 \\ r_3K_{13} + r_3K_{23} + r_3 = 0 & r'_3K_{13} + r'_3K_{23} = \ \ 0 \\ r_3K_{13} + r'_3K_{13} = 0 & r_3K_{23} + r'_3K_{23} = -1 \end{array}$$

A solution to this linear system is:

$$r'_2 = 1 \qquad r_3K_{23} = -1 \qquad r_3 = 1$$

while all other variables are null.

It can be checked that all restrictions defined in section 4.4 are respected, so we can build an attack from this solution.

To build this solution, we firstly have to exploit two services of the first session: r_3K_{23} and r_3K_{I3} (this last service is the one from which the r_3-term of the solution comes). These services are provided by M_3 when he forms the final broadcast, and we have to exploit them in opposite directions (since the value of the variables indicating how they have to be used is 1 and -1 in the solution).

So, the intruder will choose an element of G, say α^x, and place it in the message that M_3 receives in such a way that M_3 will exponentiate α^x with r_3K_{23} and r_3K_{I3}, providing the values $\alpha^{xr_3K_{23}}$ and $\alpha^{xr_3K_{I3}}$. The last term can be exponentiated by the intruder in order to obtain the pair $(\alpha^{xr_3K_{23}}, \alpha^{xr_3})$ that corresponds to the ratio $K_{23}^{-1} \in R_I$.

During the second session, the intruder has to use the r'_2 service in the positive direction. This service is provided by M_2 during the second round, so the intruder will replace an element that M_2 receives from the first round with α^{xr_3} and M_2 will send $\alpha^{xr_3r'_2}$.

Finally, the intruder possesses a pair $(\alpha^{xr_3K_{23}}, \alpha^{xr_3r'_2})$ that corresponds to the secret of M_2 during the second session. So, if he replaces the term of the broadcast intended to M_2 with $\alpha^{xr_3K_{23}}$, M_2 will compute $\alpha^{xr_3r'_2}$ as the (secret) group key; and this term is known by the intruder.

An instantiation of this scenario where M_I has chosen α^x to be equal to $\alpha^{r_1r_2}$ is represented in Fig. 3.

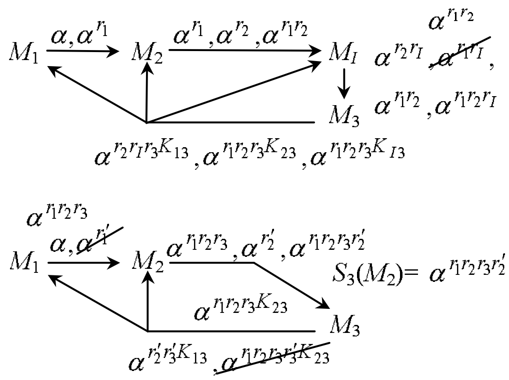


Figure 3: Attack against Implicit Key Authentication

The key computed by the group controller M_3 can be obtained by using the following solution of the linear system:

$$r'_3K_{23} = 1 \qquad r_3K_{23} = -1 \qquad r_3 = 1$$

We can observe that giving the second place in the first session to M_I (instead of the third) does not change anything to the equations we solved. In fact, this would only change the number of times services are provided in parallel, what does not affect the secrecy of particular element. Moreover, the solutions we described remain valid when considering larger groups.

It can also be observed that a similar scenario can be applied in parallel against all members of the first group, and thus that the intruder is able to share a (different) key simultaneously with all members of the second group (from which he is normally excluded).

To conclude, the implicit key authentication property seems to be very problematic in this protocol as soon as the intruder is a member of a group and intended to be excluded from another non disjoint group.

We will now analyse the other security properties, considering that the intruder is not a member of any group.

5.2 Perfect Forward Secrecy

We will divide our analysis in two parts, considering successively the two flavours of perfect forward secrecy that we described in section 2.

5.2.1 Complete Forward Secrecy

In the study of this property, we will assume that a session of the protocol has been executed in the past, and that E_I contains all long-term keys K_{in}. To simplify our writings, we are considering anew a system with one session of four legitimate participants, but the results can be easily extended to a group of size n.

Since the session has been executed in the past without any manipulation, the intruder only receives a set of values corresponding to all transmitted elements of G. Furthermore, the group key computed by the different users is not the result of the exponentiation of an unknown element with some secret value: since the intruder cannot influence this session, each user is computing the same key: $\alpha^{r_1 r_2 r_3 r_4}$. The sets of interest are therefore:

$$\begin{aligned} S &= \{r_1, r_2, r_1 r_2, r_2 r_3, r_1 r_3, r_1 r_2 r_3, \\ &\quad\ r_2 r_3 r_4 K_{14}, r_1 r_3 r_4 K_{24}, r_1 r_2 r_4 K_{34}\} \\ E_I &= \{K_{14}, K_{24}, K_{34}\} \\ R_S &= \{r_1 r_2 r_3 r_4\} \end{aligned}$$

The second step of our proof scheme will allow us to delete all keys from these sets. However, our fourth restriction on the use of the services (section 4.4) imposes that the only element of R_I that can be obtained from S are those corresponding to elements of S or to one element of S multiplied by the inverse of another one. Nevertheless, this is not sufficient to obtain $r_1 r_2 r_3 r_4$.

It can be easily verified that considering several sessions with any number of participants does not help to obtain a solution to the problem. So, if the intruder is not allowed to participate to any session, the complete forward secrecy property is verified from our model's point of view.

5.2.2 Individual Forward Secrecy

In the previous section, we considered that the sessions executed before the compromise of the long-term keys had not been manipulated by the intruder. Now, we will consider that the intruder is able to manipulate the previous sessions for one (or a few) group participants leaving these sessions correct for the other ones.

For simplicity purposes, we will anew assume a system containing only one session with four participants: M_1, M_2, M_3 and M_4. We will assume that the services can be exploited and that the long-term keys are compromised. So, the sets of interest are:

$$\begin{aligned} S &= \{r_1, r_2, r_3, r_4 K_{14}, r_4 K_{24}, r_4 K_{34}\} \\ E_I &= \{K_{14}, K_{24}, K_{34}\} \\ R_S &= \{r_1 K_{14}^{-1}, r_2 K_{24}^{-1}, r_3 K_{34}^{-1}, r_4\} \end{aligned}$$

If we delete the keys from these sets (as described in the second step of our proof-scheme), we can find that for each secret r_i, the resulting linear system has a trivial solution: $r_i = 1$. These solutions meet all restrictions described above, and we can then assume that the individual forward secrecy is somehow suspicious.

We first consider the solution $r_4 = 1$ that corresponds to the secret of M_4. This service is in fact offered three times, when M_4 forms the broadcast by exponentiating the terms he receives with r_4K_{14}, r_4K_{24} and r_4K_{34}. These three services can be used indifferently, and we will choose to use the first one. So, the intruder replaces the term that M_4 will use to compute its view of the key by any element of G (say α^x) and replaces the term that M_4 will exponentiate with r_4K_{14} by the same element. A representation of this scenario where the intruder has chosen α^x to be $\alpha^{r_1r_2r_3}$ can be found in Fig. 4.

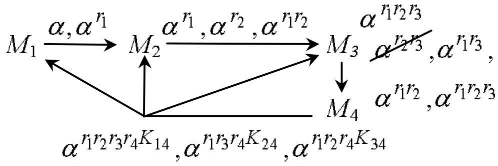


Figure 4: Attack against Individual Forward Secrecy

As we required, this manipulation only affects the key computed by M_1 ($\alpha^{r_1^2r_2r_3r_4}$) while the other group members are still computing the same key: $\alpha^{r_1r_2r_3r_4}$. Finally, if K_{14} is compromised, the intruder is able to compute this key from the term $\alpha^{r_1r_2r_3r_4K_{14}}$ sent by M_4. The individual forward secrecy property is therefore not verified.

The fact that this attack provides the key computed by group-members other than M_4 corresponds to the exploitation of solutions of the type $r_i = 1$, $r_4(K_{i4}) = -1$, $r_4(K_{14}) = 1$ that are less trivial solutions of the system corresponding to the secret of M_i.

We can now check the solutions of the linear equations corresponding to the secrets of M_1, M_2 and M_3. Besides the solution discovered when exploiting the solution $r_4 = 1$, we can also use the solutions $r_i = 1 (1 \leq i < 4)$ (and all other services unused). So, if the intruder wants to attack M_i, he simply has to replace the term that M_i will exponentiate with $r_iK_{i4}^{-1}$ by any term that M_i previously exponentiated with r_i. An example of this scenario where $i = 2$ is represented in Fig. 5.

It can be observed however that this time M_i is computing a key that he does not share with anyone. So, even if this scenario can be performed in parallel against several group members, it appears to be less awkward than the previous one.

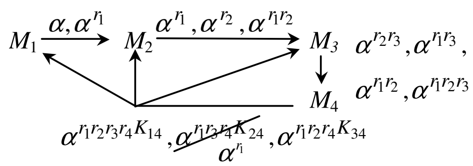


Figure 5: Attack against Individual Forward Secrecy

5.3 Resistance to known-keys attacks

This property expresses that the compromising of session keys does not allow a passive adversary to compromise keys of other sessions nor an active adversary to impersonate one of the protocol parties. The part of this property concerning the passive adversary is studied in [2] and we will focus on the second part. However the authors of [2] claim that the resistance to an active adversary is more dubious and suggest an attack that does not seem very useful in practice.

We will apply our method to the verification of this property as follows: we are assuming two sessions of the protocol with the same four participants. The random numbers generated by M_i during the first and second sessions of the protocol will be denoted r_i and r'_i respectively. In order to model the compromise of the session keys, we will assume that the users are providing services corresponding to the secrets of the first session: $r_i K_{i4}^{-1}$ and r_4. Hence we can write that

$$\begin{aligned} S &= \{r_1, r_2, r_3, r_4K_{14}, r_4K_{24}, r_4K_{34}, \\ &\quad r'_1, r'_2, r'_3, r'_4K_{14}, r'_4K_{24}, r'_4K_{34}, \\ &\quad r_1K_{14}^{-1}, r_2K_{24}^{-1}, r_3K_{34}^{-1}, r_4\} \\ E_I &= \emptyset \\ R_S &= \{r'_1K_{14}^{-1}, r'_2K_{24}^{-1}, r'_3K_{34}^{-1}, r'_4\} \end{aligned}$$

The linear system corresponding to $r'_iK_{in}^{-1} (1 \le i < 4)$ is

$$\begin{aligned} r_j + r_jK_{j4}^{-1} = 0 \qquad r'_j &= \delta(i,j) \\ r_4K_{14} + r_4K_{24} + r_4K_{34} + r_4 &= 0 \\ r'_4K_{14} + r'_4K_{24} + r'_4K_{34} &= 0 \\ r_4K_{j4} + r'_4K_{j4} - r_jK_{j4}^{-1} &= -\delta(i,j) \end{aligned}$$

where $1 \le j < 4$ and $\delta(i,j) = 1$ if $i = j$ and 0 otherwise. It can be checked that:

$$r_i = -1, r_iK_{i4}^{-1} = 1, r'_i = 1$$

and all other services being unused, is a solution.

If $i = 1$, it is however impossible to find an attack scheme: this solution would need the service $r_1K_{14}^{-1}$ (i.e. the service corresponding to the key computed by M_1 during the first session) to be applied on the value $\alpha^{r'_1}$ that is provided during the second session. Nevertheless, for all other values of i, the following attack is possible:

1. Let α^x be one of the input terms of the i-th round of the first protocol run. M_i will therefore send α^{xr_i}.
2. The intruder replaces then the term $\alpha^{\frac{r_1 \ldots r_4}{r_i} K_{in}}$ with α^x. Hence $S_n(M_i)$ will be equal to $\alpha^{xr_i K_{in}^{-1}}$. Since we study known-key attacks, we will assume that this value is compromised.
3. In the second run of the protocol, the intruder replaces one of the i-th round inputs with $\alpha^{xr_i K_{in}^{-1}}$. M_i will therefore send $\alpha^{xr_i K_{in}^{-1} r'_i}$.
4. In the broadcast of the second run of the protocol, the intruder finally replaces the term intended to M_i with α^{xr_i} (obtained in the first step of our scenario). Hence $S'_n(M_i)$ will be computed as $\alpha^{xr_i K_{in}^{-1} r'_i}$ that has been obtained during the third step of our scenario.

At the end of this scenario, the intruder will possess a key that M_i believes to be secret. However this key is unknown to the rest of the group and the compromised key used is a malformed key which reduces the scope of these attacks. However, if all malformed keys are available, the intruder can perform this attack simultaneously against almost all members of the group.

We can now turn to the secrecy of r'_n. If we look at the linear system corresponding to this secret (i.e. the system we just described where the right part of each equation has been updated), we can find two types of solutions. The first one is:

$$r_i = -1, r_i K_{in}^{-1} = 1, r'_n K_{in} = 1$$

From these solutions, we can obtain the scenarios corresponding to the attack proposed in [2]. The scope of these attacks is the same as the one we just described.

However another type of solution can be found:

$$r_n = 1, r_n K_{in} = -1, r'_n K_{in} = 1$$

For $1 \leq i < 4$, it is possible to apply the following scenario:

1. In the last round inputs of the protocol's first session, the intruder replaces $\alpha^{r_1 \ldots r_3 / r_i}$ with $\alpha^{r_1 r_2 r_3}$. Hence all elements of the broadcast will be preserved except the one intended to M_i that will be equal to $\alpha^{r_1 r_2 r_3 r_4 K_{i4}}$. $S_4(M_4)$ will therefore be equal to $\alpha^{r_1 r_2 r_3 r_4}$ and shared by all members of the group except M_i. In a context of known-key attacks, we will assume that this key is compromised.
2. In the last round inputs of the protocol's second session, the intruder will substitute $\alpha^{r'_1 r'_2 r'_3 / r'_i}$ with $\alpha^{r_1 r_2 r_3 r_4}$ and $\alpha^{r'_1 r'_2 r'_3}$ with $\alpha^{r_1 r_2 r_3 r_4 K_{i4}}$. Hence M_4 will broadcast $\alpha^{r_1 r_2 r_3 r_4 K_{i4} r'_4}$ and compute his view of the key $S_4(M_4) = \alpha^{r_1 r_2 r_3 r_4 K_{i4} r'_4}$.

A representation of this attack for $i = 1$ can be found in Fig. 6. This scenario is more dangerous since we assume that a key, shared (and used normally) by all members of the group except one, is compromised. To conclude, the resistance of this protocol to known-key attacks is more problematic than expected in [2].

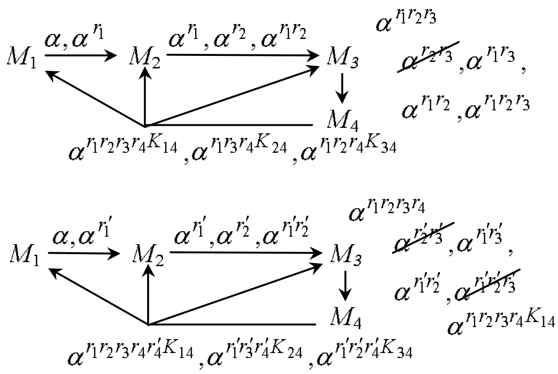


Figure 6: Attack against Resistance to Known Keys

5.4 Consideration of the Use of the A-GDH.2-MA Protocol

The key generation protocol (A-GDH.2) is not intended to be used alone: it is often useful to enable the addition or deletion of group members after the initial group creation. New protocols will be used in order to provide each of these services. As we said above, the aim of the A-GDH.2-MA protocol is the addition of a new member in the group. In this paragraph, we will extend our analysis of the A-GDH.2 protocol by taking into account the presence of the Member Adding protocol. As a first step, we will study the Implicit Key Authentication property and consider two sessions of the protocols: in the first session, the A-GDH.2 protocol is executed by M_1, M_2 and M_3; while in the second session a member is added to this group. Following the same approach as above, we will first write the sets S, E_I and R_S that will be the union of those corresponding to the sessions of each protocol:

$$\begin{aligned}
S &= \{r_1, r_2, r_3K_{13}, r_3K_{23}, \\
&\quad r_3\hat{r}_3, r_3\hat{r}_3K_{13}, r_3\hat{r}_3K_{23}, r_3\hat{r}_3K_{33}, \\
&\quad r_4K_{14}, r_4K_{24}, r_4K_{34}, K_{43}K_{44}\} \\
E_I &= \emptyset \\
R_S &= \{r_1K_{13}^{-1}, r_2K_{23}^{-1}, r_3, r_1K_{13}^{-1}K_{14}^{-1}, r_2K_{23}^{-1}K_{24}^{-1}, \\
&\quad r_3K_{33}^{-1}K_{34}^{-1}, r_4K_{43}^{-1}K_{44}^{-1}\}
\end{aligned}$$

The first part of the expression of S corresponds to the execution of the key-generation protocol, the second to the first round of the A-GDH.2-MA protocol, and the last to the second round of the A-GDH.2-MA protocol. The first three elements of R_S are corresponding to the key generation protocol while the others are corresponding to the member adding protocol.

E_I being empty, we can immediately study the linear system corresponding to the secrets. This system is a little larger than the previous one but remains quite regular. We give its expression for the verification of the secrecy of $r_1 K_{13}^{-1}$.

$$\begin{array}{rcl}
r_1 = 1 \qquad r_2 & = & 0 \\
r_3K_{13} + r_3K_{23} + r_3\hat{r}_3 + r_3\hat{r}_3K_{13} + r_3\hat{r}_3K_{23} + r_3\hat{r}_3K_{33} & = & 0 \\
r_4K_{14} + r_4K_{24} + r_4K_{34} & = & 0 \\
r_3\hat{r}_3 + r_3\hat{r}_3K_{13} + r_3\hat{r}_3K_{23}r_3\hat{r}_3K_{33} & = & 0 \\
r_3K_{13} + r_3\hat{r}_3K_{13} = -1 \qquad r_3K_{23} + r_3\hat{r}_3K_{23} & = & 0 \\
r_3\hat{r}_3K_{33} & = & 0 \\
r_4K_{14} = -1 \qquad r_4K_{24} = 0 \qquad r_4K_{34} = 0 \qquad K_{43}K_{44} & = & 0
\end{array}$$

If we solve this system, we find that $r_1K_{13}^{-1}$ and $r_2K_{23}^{-1}$ can be compromised: they can be obtained by combining the services r_i, $r_3\hat{r}_3$ and $r_3\hat{r}_3K_{i3}$ (for $i = 1, 2$). The other secrets cannot be compromised in this scheme: the solutions corresponding to r_3 do not respect the restrictions on the use of the services; and the system has no solution for the other secrets. The scenario corresponding to $r_iK_{i3}^{-1}$ is as follows:

1. M_1, M_2 and M_3 execute the key-generation protocol, but the intruder intercepts the broadcast of the last round.
2. The intruder obtains that M_n starts the A-GDH.2-MA protocol with some other user of the network, and intercepts the first message.
3. The intruder sends the parts of this message corresponding to the users M_1 and M_2 faking the broadcast of the A-GDH.2 protocol.

When it is done, M_1 and M_2 are sharing with the intruder the key $\alpha^{r_1r_2r_3\hat{r}_3}$ that has been sent by M_3 as the last part of the first message of the A-GDH.2-MA protocol. A scheme corresponding to this attack is represented in Fig. 7.

To conclude, the Implicit Key Authentication property seems to become even more problematic when we consider the possibility of the use of the A-GDH.2-MA protocol in parallel with the A-GDH.2 protocol: it is not anymore necessary for the intruder to be intended to take part to some session if he wants to defeat the IKA property. The other security properties could be similarly analysed.

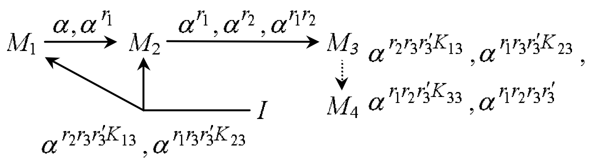


Figure 7: Attack considering the A-GDH.2-MA Protocol

6 Analysis of the SA-GDH.2 protocol

In the previous sections, we used our machinery to analyse the A-GDH.2 protocols and obtained attacks against all claimed security properties. We think that these attack schemes are very systematic and can be exploited in a large range of protocols that have been (or could be) defined using the A-GDH scheme.

We will illustrate this intuition with an attack against the SA-GDH.2 protocol [2]. This protocol is an extension of the A-GDH.2 one providing *complete group authentication*, i.e. guaranteeing that two group members M_i and M_j compute the same key $S_{i,j}$ only if $S_{i,j}$ has been contributed to by every $M_p \in M$ (assuming that M_i and M_j have the same view of the group membership). The main change in the requirements for this protocol is in the definition of the long-term keys: it is assumed that each user M_i shares two keys (one for each direction of communication) with every other user M_j: for every ordered pair $< i, j >$ $(1 \leq i \neq j \leq n)$, we use $< K_{ij}, K_{ij}^{-1} >$ to denote the unidirectional key shared by M_i and M_j and its inverse, respectively.

As for A-GDH.2, this protocol executes in n rounds; the $n-1$ first ones collecting the key contributions, while the last one is used to broadcast the keying material. It is as follows:

Initialization:
Let p be a prime integer and q a prime divisor of $p-1$. Let G be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and let α be a generator of G.

Round i $(1 \leq i \leq n)$:

1. M_i selects $r_i \in \mathbb{Z}_q^*$
2. M_i receives a set of n intermediate values: $\{V_k | 1 \leq k \leq n\}$ (M_1 can be thought of as receiving an empty set in the first round and, in the initial round, M_1 sets $V_1 = \alpha$):

$$V_k = \begin{cases} \alpha^{\frac{r_1 \ldots r_{i-1}}{r_k}.(K_{1k} \ldots K_{(i-1)k})} & \text{if } k \leq (i-1) \\ \alpha^{r_1 \ldots r_{i-1}.(K_{1k} \ldots K_{(i-1)k})} & \text{if } k > (i-1) \end{cases}$$

3. M_i updates each V_k as follows:

$$V_k = \begin{cases} (V_k)^{K_{ik} r_i} = \alpha^{\frac{r_1 \ldots r_i}{r_k}.(K_{1k} \ldots K_{ik})} & \text{if } k < i \\ (V_k)^{K_{ik} r_i} = \alpha^{r_1 \ldots r_i.(K_{1k} \ldots K_{ik})} & \text{if } k > i \\ V_k & \text{if } k = i \end{cases}$$

4. M_i sends the updated V_k to M_{i+1} if $i < n$ or broadcast $V_1 \ldots V_{n-1}$ if $i = n$.

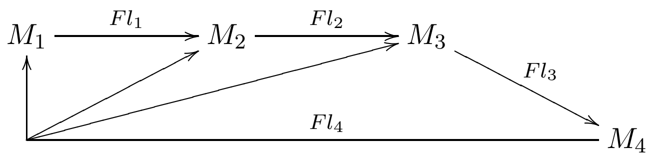
Upon receipt of the above, every M_i selects the appropriate V_i where :

$$V_i = \alpha^{\frac{r_1 \ldots r_n}{r_i}.(K_{1i} \ldots K_{ni})}$$

and computes:

$$S_n = V_i^{(K_{1i}^{-1} \ldots K_{ni}^{-1}).r_i} = \alpha^{r_1 \ldots r_n}$$

In order to better understand the structure of this protocol, we represented a protocol run with four participants in Fig. 8.



$$\begin{aligned} Fl_1 &= \alpha, \alpha^{r_1 K_{12}}, \alpha^{r_1 K_{13}}, \alpha^{r_1 K_{14}} \\ Fl_2 &= \alpha^{r_2 K_{21}}, \alpha^{r_1 K_{12}}, \alpha^{r_1 r_2 K_{13} K_{23}}, \alpha^{r_1 r_2 K_{14} K_{24}} \\ Fl_3 &= \alpha^{r_2 r_3 K_{21} K_{31}}, \alpha^{r_1 r_3 K_{12} K_{32}}, \alpha^{r_1 r_2 K_{13} K_{23}}, \alpha^{r_1 r_2 r_3 K_{14} K_{24} K_{34}} \\ Fl_4 &= \alpha^{r_2 r_3 r_4 K_{21} K_{31} K_{41}}, \alpha^{r_1 r_3 r_4 K_{12} K_{32} K_{42}}, \alpha^{r_1 r_2 r_4 K_{13} K_{23} K_{43}} \end{aligned}$$

Figure 8: SA-GDH.2 Protocol Run with 4 Participants

We can now examine whether the implicit key authentication property is verified by using our method in the same way as we used it with the A-GDH.2 protocol: we will assume a first session with group constitution M_1, M_I, M_2, M_3 and a second one with group constitution M_1, M_2, M_3. If we use the letters r_i and r'_i to denote the random contributions generated by the user M_i during the first and the second sessions respectively, we can describe this configuration as:

$$\begin{aligned} S &= \{r_1 K_{1I}, r_1 K_{12}, r_1 K_{13}, r_I K_{I1}, r_I K_{I2}, r_I K_{I3}, \\ &\quad r_2 K_{21}, r_2 K_{2I}, r_2 K_{23}, r_3 K_{31}, r_3 K_{3I}, r_3 K_{32}, \\ &\quad r'_1 K_{12}, r'_1 K_{13}, r'_2 K_{21}, r'_2 K_{23}, r'_3 K_{31}, r'_3 K_{32}\} \\ E_I &= \{r_I, K_{1I}, K_{2I}, K_{3I}\} \\ R_S &= \{r'_1 K_{21}^{-1} K_{31}^{-1}, r'_2 K_{12}^{-1} K_{32}^{-1}, r'_3 K_{13}^{-1} K_{23}^{-1}\} \end{aligned}$$

If we look at the equation system for the secret of M_2 for instance, we can obtain that a solution allowing to obtain the secret $r_2' K_{12}^{-1} K_{32}^{-1}$ is:

$$r_2' K_{21} = r_2(K_{2I}) = r_1(K_{1I}) = r_3(K_{3I}) = 1$$
$$r_2 K_{21} = r_1 K_{12} = r_3 K_{32} = -1$$

This solution meets all constraints defined in section 4.4, and can be used to obtain the attack scenario described in Fig. 9.

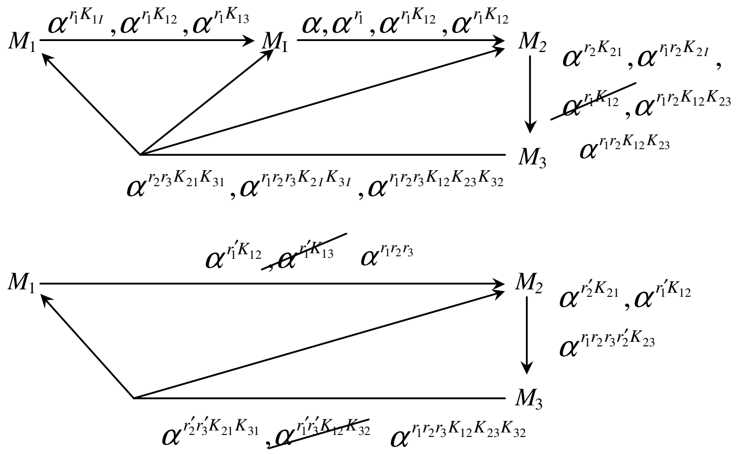


Figure 9: Attack against the SA-GDH.2 Protocol

In this figure, we have shown the values exchanged by the different group members ordered by the indexes of V. During his round of the first session, the intruder (that is the second member of the group) replaces V_3 by $\alpha^{r_1K_{12}}$ and V_2 by α^{r_1}, so M_2 provides $\alpha^{r_1r_2K_{2I}}$ and $\alpha^{r_1r_2K_{12}K_{23}}$. M_I will put this last value in place of V_2 in the input message of M_3, so M_3 will send $\alpha^{r_1r_2r_3K_{2I}K_{3I}}$ and $\alpha^{r_1r_2r_3K_{12}K_{23}K_{32}}$. These last two values will be used in the second session: in the first message of this session, M_I will replace V_2 (i.e. the element intended to M_3) with $\alpha^{r_1r_2r_3}$ that he computed from the previous one. M_2 will therefore update (and send) V_2 as $\alpha^{r_1r_2r_3r_2'K_{23}}$. Finally, in the last broadcast, M_I will replace V_2 with $\alpha^{r_1r_2r_3K_{12}K_{23}K_{32}}$ and M_2 will compute his view of the group key as $\alpha^{r_1r_2r_3r_2'K_{23}}$, value that he sent during the previous round.

We can see that this key was obtained in a way similar to the one we pointed out in the case of the A-GDH.2 protocol: the random contribution to the key was obtained from the message M_2 emits during the second round, while the ratios corresponding to the long-term keys included in M_2's secret were obtained during the first session of which the intruder is a regular member. According to our experience, this scheme can be reproduced on many other variants of the A-GDH protocols.

7 Concluding Remarks

In this paper, we presented a number of attacks against authenticated group key agreement protocols. Most of them are coming from the fact that the properties of group protocols are not trivial extensions of the ones of two-parties protocols: for instance, the fact that a group member computes a bad key can remain unnoticed, particularly if the group is large, while it prevents any exchange of messages when only two users are considered. The discovery of all these vulnerabilities emphasizes the need of using systematic methods for reasoning about security protocols.

Due to their particular structure, the analysis of the GDH protocols (A-GDH.2 and SA-GDH.2) required the development of a new machinery. We discovered that within some reasonable extensions of the idealizations usually stated in the context of formal approaches for protocol analysis, and by adequately modeling the operations performed by the honest users during the protocol sessions, it was particularly easy to verify the security properties of these protocols since it came down to solving a linear equation system.

Trying to draw lessons from all the weaknesses we encountered, we are currently working on the definition of a protocol that would be able to provide the security properties we examined throughout this paper.

Acknowledgements

The authors would like to thank the anonymous referees for their useful comments (among other suggestions, they proposed the definition of the two flavours of perfect forward secrecy) and S. Baudine for her valuable help to finish off this paper. O. Pereira would like to thank the Belgian Funds for Scientific Research (FNRS) for its financial support.

References

[1] M. Abadi and P. Rogaway. Reconciling two views of cryptography. In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *Proceedings of the IFIP International Conference on Theoretical Computer Science 2000*, pages 3–22, Sendai, Japan, 2000. LNCS Vol. 1872.

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

[3] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of Advances in Cryptology: Crypto'93*, pages 232–249, Santa Barbara, USA, 1994. LNCS Vol. 773.

[4] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Cryptography and Coding*, pages 30–45, Cirencester, UK, 1997. LNCS Vol. 1355.

[5] E. Bresson, O. Chevassut, and D. Pointcheval. Provably authenticated group Diffie-Hellman key exchange - the dynamic case. In C. Boyd, editor, *Advances in Cryptology - Proceedings of AsiaCrypt 2001*, pages 290–309, Gold Coast, Australia, 2001. LNCS Vol. 2248.

[6] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group diffie-hellman key exchange under standard assumptions. In L. Knudsen, editor, *Advances in Cryptology - Proceedings of Eurocrypt 2002*, pages 321–336, Amsterdam, the Netherlands, 2002. LNCS Vol. 2332.

[7] J. Bryans and S. Schneider. CSP, PVS, and a recursive authentication protocol. In *Proceedings of the DIMACS Workshop on Formal Verification of Security Protocols*, Rutgers, USA, 1997.

[8] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[9] A. Durante, R. Focardi, and R. Gorrieri. CVS: A tool for the analysis of cryptographic protocols. In *Proceedings of the 12-th IEEE Computer Security Foundations Workshop*, pages 203–212, Mordano, Italy, 1999. IEEE Computer Society Press.

[10] J. Guttman, F. J. Thayer Fábrega, and L. Zuck. The faithfulness of abstract protocol analysis: Message authentication. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 186 – 195, Philadelphia, USA, 2001. ACM Press.

[11] Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Proceedings of IFIP-SEC 2001*, pages 229–244, Paris, France, 2001. Kluwer Publishers.

[12] G. Lowe. Some new attacks upon security protocols. In *Proceedings of 9th IEEE Computer Security Foundations Workshop*, pages 162–169, Kenmare, Ireland, 1996. IEEE Computer Society Press.

[13] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of 10th IEEE Computer Security Foundations Workshop*, pages 31–44, Rockport, USA, 1997. IEEE Computer Society Press.

[14] G. Lowe. Casper: A compiler for the analysis of security protocols. *Journal of Computer Security*, 6:53–84, 1998.

[15] W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In *Proceedings of the DIMACS Workshop on Formal Verification of Security Protocols*, Rutgers, USA, 1997.

[16] C. Meadows. The NRL protocol analyzer : an overview. *Journal of Logic Programming*, 26(2):113–131, 1996.

[17] C. Meadows. Extending formal cryptographic protocol analysis techniques for group protocols and low-level cryptographic primitives. In *Proceedings of the Workshop on Issues in the Theory of Security*, pages 1–4, Geneva, Switzerland, 2000.

[18] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, chapter 12, pages 489–541. CRC Press, 1999.

[19] L. C. Paulson. Mechanised proofs for a recursive authentication protocol. In *Proceedings of the 10-th IEEE Computer Security Foundations Workshop*, pages 84–95, Rockport, USA, 1997. IEEE Computer Society Press.

[20] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[21] D. Song, S. Berezin, and A. Perrig. Athena, a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1,2):47–74, 2001.

[22] P. Syverson and P. van Oorschot. On unifying some cryptographic protocols logics. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, USA.

[23] F. J. Thayer, J. H. Herzog, and J. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.