Generic Insecurity of Cliques-Type Authenticated Group Key Agreement Protocols

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

Abstract

The A-GDH.2 and SA-GDH.2 authenticated group key agreement protocols showed to be flawed in 2001. Even though the corresponding attacks (or some variants of them) have been rediscovered in several different frameworks, no fixed version of these protocols has been proposed until now.

In this paper, we prove that it is in fact impossible to design a scalable authenticated group key agreement protocol based on the same design assumptions as the A-GDH ones. We proceed by providing a systematic way to derive an attack against any A-GDH-type protocol with at least four participants and exhibit protocols with two and three participants which we cannot break using our technique. As far as we know, this is the first generic insecurity result reported in the literature concerning authentication protocols.

1 Introduction

The A-GDH.2 and SA-GDH.2 authenticated group key agreement protocols [1, 2], which are part of the Cliques protocol suites, have been shown to be flawed in 2001 [17, 18]. Even though the corresponding attacks (or some variants of them) have been rediscovered in several different frameworks (using the Casper tool [5], rank functions [6] or the constraint solving approach [13] for instance), no fixed version of these protocols has been proposed until now.

As we tried to design such fixes, i.e. authenticated group key agreement protocols built from the same ingredients as the A-GDH protocols, we found

*O. Pereira is postdoctoral researcher of the Belgian National Funds for Scientific Research (FNRS)

that the method we proposed in [18] could always be used to find attacks against our candidates.

Actually, we prove in this paper that it is impossible to build a scalable authenticated group key agreement protocol using the technique adopted for the A-GDH protocols, that is, by constructing a group Diffie-Hellman key $\alpha^{r_1 \cdots r_n}$ through the exchange of partial group Diffie-Hellman values of form $\alpha^{\prod r_i}$, possibly exponentiated with long-term symmetric keys shared between the different group members. Our proof proceeds by providing a systematic procedure allowing the building of an attack against the implicit key authentication property for any protocol of the family we consider (provided that the protocol is executed by at least four principals). As far as we know, this is the first such impossibility result reported in the literature concerning authentication protocols.

In the next section, we will define the protocol family we want to analyze. The next step of our analysis, exposed in Section 3, will consist in the definition of several properties all protocols of our family must exhibit, mainly due to the fact that the different group members must be able to compute the same group key. The main result of this section will be the proof that the (secret) computation that each group member performs at the end of a protocol execution in order to obtain the group key can be written as the composition of computations executed by honest users during different protocol sessions, computations whose inputs and outputs can be eavesdropped.

This result does not however guarantee that the routing of the messages as given in the protocol definition will allow an active attacker to compose these computations as he would like to do: we will exhibit a three-party protocol for which we cannot use our Section 3 result to derive an attack. However, in Section 4, we will prove that it is possible to exploit that result in order to undermine the implicit key authentication property for at least one member of any protocol of our family provided that it is executed by at least four users.

2 The GDH-Protocols

2.1 Authenticated Group Key Agreement Protocols

The protocols of the family we consider, which we will call the *GDH-Protocols*, are group key agreement protocols.

Definition 2.1 *A* Group Key Agreement Protocol *is a protocol enabling a group of n users $\mathsf{M} = \{M_1, \ldots, M_n\}$ to contributively generate a key that should be known by all group members at the end of a protocol execution.*

It can be observed that the protocols we consider are different from the key exchange protocols usually analyzed through Dolev-Yao-type methods (such as the Needham-Schroeder symmetric key protocol [14], the Otway-Rees protocol [15], the Woo and Lam protocol [22], for instance), because they are required to be contributive: no group member can choose the key in advance.

In the presence of an active attacker, i.e. an attacker who can intercept, reroute and send messages of his own, authentication properties are often required. The most usual authentication property, which is the one we will consider in this paper, is the implicit key authentication (IKA) [12].

Definition 2.2 *A protocol expected to be executed by a group of users* M *is said to achieve* Implicit Key Authentication *(IKA) if, when he completed his role in a session of the protocol, each* $M_i \in \mathsf{M}$ *is assured that no party* $M_I \notin \mathsf{M}$ *can learn the key* $S(M_i)$ *(i.e.* M_i*'s view of the session key).*

The attacker is assumed to be a regular network user (and possibly to have several identities), and to be a regular member of some groups. His goal is then to obtain the key computed by honest users in groups from which he is expected to be excluded. We may observe that this property does not mean that all group members have any knowledge of a group key at the end of the protocol execution, nor that they agree on its value. These last properties could be achieved through key confirmation extensions of the protocols and will not be considered here.

Besides the IKA property, two other types of security properties are usually desirable: forward secrecy which guarantees that the compromise of long-term keys cannot result in the compromise of past session keys; and resistance to known session-secret attacks which guarantees that the compromise of old session-secrets cannot result in the compromise of future session keys [18]. We do not discuss these properties anymore and will only consider the IKA property in the rest of this paper.

2.2 The A-GDH.2 Protocol

A well-known example of authenticated group key agreement protocol is the A-GDH.2 protocol [1, 2] which we will use in order to provide intuitions about our attack methodology. The A-GDH.2 protocol is executed by a pool of users M who agreed on performing all computations in an algebraic group $\mathcal{G}$ of prime order q, group in which the Decisional Diffie-Hellman problem is believed to be hard (the subgroup of order q of $\mathbb{Z}_p^*$ where p and q are large prime numbers can be chosen to this effect). All users also agree on the use of a specific generator α of $\mathcal{G}$, and these two choices are public.

The authentication mechanism adopted in the A-GDH protocols relies on the assumption that each pair of users (M_i, M_j) share a long-term secret key $K_{ij} \in \mathbb{Z}_q^*$.

These assumptions and notations having been introduced, we now define the way the A-GDH.2 protocol is executed.

When starting a protocol execution, each group member $M_i \in \mathsf{M}$ selects a random key contribution $r_i \in \mathbb{Z}_q^*$. The first group member, M_1, computes α^{r_1} and sends $\langle \alpha, \alpha^{r_1} \rangle$ to M_2. Then, all group members from M_2 to M_{n-1} perform the following sequence of actions: M_i exponentiate the i elements of $\mathcal{G}$ he received with r_i, inserts the last received element in the next to last position and sends the result to the next group member. Finally, for each but the last element of $\mathcal{G}$ that M_n received, he exponentiates the i-th of them with $r_n K_{in}$ and finally broadcasts the result. The messages transcript is then as follows.

Protocol 1: A-GDH.2 Protocol

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

$M_i \rightarrow M_{i+1}$: $\quad \{\alpha^{\frac{r_1 \cdots r_i}{r_j}} | j \in [1, i]\}$, $\alpha^{r_1 \cdots r_i}$

Round n:

$M_n \rightarrow$ All M_i: $\quad \{\alpha^{\frac{r_1 \cdots r_n}{r_i} K_{in}} | i \in [1, n[\}$

Upon receipt of the above, every $M_i \in \mathsf{M}$ computes the group key as:

$$S_n(M_i) \;=\; \alpha^{\frac{r_1 \cdots r_n}{r_i} \cdot r_i \cdot K_{in}^{-1}} = \alpha^{r_1 \cdots r_n},$$

except M_n who computes:

$$S_n(M_n) \;=\; \alpha^{(r_1 \cdots r_{n-1}) \cdot r_n}$$

from the last element of $\mathcal{G}$ he received during the $(n-1)$-th round.

A typical run of this protocol with 3 participants is represented using the strand space notations in Fig. 1. In this figure, $\rightarrow$-arrows denote message transmission and $\Rightarrow$-arrows distinguish successive actions performed by a given protocol participant. Note that both horizontal arrows on the last line of this figure correspond to the broadcast sent by M_3.

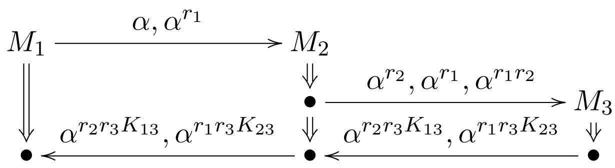


Figure 1: A-GDH.2 Protocol Run with 3 Participants

An important feature of this protocol is that all group members do not check anything about the sequences of elements of $\mathcal{G}$ they receive (except that these sequences have the expected length) and, as a result, that the attacker can always generate messages with the structure expected by the different users. This is not the case for the classical authentication protocols usually analyzed in Dolev-Yao-type models where messages are typically expected to be encrypted through keys the attacker does not know or to contain secret values such as nonces that the attacker should not be able to guess. In this protocol, and in all protocols we will consider in this paper, the key authentication relies on the fact that the values that the different group members are using for computing their view of the group key are not known by the attacker.

2.3 An Attack Against the A-GDH.2 Protocol

In order to provide intuitions regarding the systematic attack construction process we will describe further, we now describe an attack against the A-GDH.2 protocol when it is executed by three participants.

Let us consider an attacker whose identifier is M_I and who wants to undermine the IKA property by fooling M_2 into accepting a key he knows in a session M_2 thinks he is executing with M_1 and M_3. Since M_2 computes his view of the group key by exponentiating the second term of M_3's final broadcast with $r_2K_{23}^{-1}$, the goal of the intruder consists in obtaining a pair of elements of the form $(\alpha^x, \alpha^{xr_2K_{23}^{-1}})$ and in replacing the second term of that broadcast with α^x so that M_2 will compute $\alpha^{xr_2K_{23}^{-1}}$ as group key.

The attacker can obtain such a pair by using the protocol participants as oracles, that is, by exploiting the services they provide. We call *service* a computation carried out by a honest user during a protocol execution; computation of which the input and result can be eavesdropped by the intruder. In most cases, the intruder will furthermore be able to exploit these services in a more efficient way: he will be able to replace services' input with a value of his own choice and then to obtain the result of these services provided for this chosen input. All services provided during an A-GDH.2 protocol execution are (modular) exponentiations. As it does not cause any ambiguity, we will therefore call a service consisting in exponentiating an element α^x with a value s as providing the s-service. If we look at the protocol execution described in Fig. 1, we may observe that M_1 provides the r_1-service, that M_2 provides the r_2-service, and that M_3 provides the r_3K_{13}- and r_3K_{23}-services.

Let us now consider a second protocol session executed by M_I, M_2 and M_3. If we use r'_i to denote M_i's contribution in that session, the provided services are r'_2, r'_3K_{I3} and r'_3K_{23} (we do not consider the actions of M_I since they would only involve values that the intruder knows).

It can now be observed that a pair of form $(\alpha^x, \alpha^{xr_2K_{23}^{-1}})$ can be built by exploiting the services r_2, r'_3K_{I3} and r'_3K_{23}. Actually, if the intruder replaces the input values of these last two services with a random value he knows, say α^y, M_3 will send the values $\alpha^{yr'_3K_{I3}}$ and $\alpha^{yr'_3K_{23}}$. Then, if M_I replaces the input of the r_2-service with $\alpha^{yr'_3K_{I3}}$, M_2 will send the value $\alpha^{yr'_3K_{I3}r_2}$. Finally, if the intruder exponentiates this last value with K_{I3}^{-1}, he will be in possession of the pair $(\alpha^{yr'_3K_{23}}, \alpha^{yr'_3r_2})$ which has the desired form. The final step of this attack consists in replacing the second term of the broadcast sent by M_3 with $\alpha^{yr'_3K_{23}}$, in such a way that M_2 will compute $\alpha^{yr'_3r_2}$ as group key. So, at this point, M_2 expects that only M_1 and M_3 could know $\alpha^{yr'_3r_2}$, but it is known by M_I and this is in opposition with the implicit key authentication property.

We would like to distinguish two phases in this attack. The first one consists in finding which services can be used in order to obtain a pair of the desired form. This comes down to trying to write the value that M_2 will use in order to compute his view of the group key as a product of services and values that the intruder knows: in the attack above, we found that $r_2K_{23}^{-1} = r_2 \cdot r'_3K_{I3} \cdot (r'_3K_{23})^{-1} \cdot (K_{I3})^{-1}$. In Section 3, we will show that, for the family of protocols we consider, such equations can always be found, provided that the protocol is executed by at least 3 users.

The second phase consists in finding a way to exploit the equation found during the first step in order to obtain an attack scenario. In our example above, it consisted in starting with a pair of form (α^y, α^y) and replacing the input of services inverted in the previous equation with the first term of the pair while the input of non-inverted services was replaced by the second term. So, we successively constructed the pairs (α^y, α^y), $(\alpha^{yr'_3K_{23}}, \alpha^{yr'_3K_{I3}})$ and $(\alpha^{yr'_3K_{23}}, \alpha^{yr'_3K_{I3}r_2})$. This was however an easy case: if we had to use the r_1-service for instance, we would not have been able to replace the input of this service with a value of our choice since M_1 always uses α as input value for this service. Our goal in Section 4 will be to prove that at least one of the equations obtained during our first attack phase uses services which can be collected in order to build an attack against the protocols we consider, provided that they are executed by at least four users.

2.4 A Fix for the A-GDH.2 Protocol?

As we found an attack against the A-GDH.2 protocol, we now naturally wonder how this protocol could be fixed, that is, how we could write a protocol based on the same design assumptions as the A-GDH.2 protocol and guaranteeing the expected implicit key authentication property.

A first design assumption we keep is that we only consider protocols executed by exchanging elements of the public group $\mathcal{G}$, those elements being built by exponentiating a public generator α with a product of:

- random values generated during the protocol execution and only known by the user who generated them and
- long-term shared keys of form K_{ij}, where K_{ij} is only known by M_i and M_j.

So, we only consider the exchange of elements of $\mathcal{G}$ of form $\alpha^{\prod r_i \prod K_{jk}}$ (but not elements obtained by multiplying two other elements of $\mathcal{G}$ for instance).

A second design assumption is that we consider protocols aiming at building a shared group key of form $\alpha^{r_1 \cdots r_n}$ where r_i has been generated by the i-th group member M_i. This guarantees that the protocol is contributive, which is required for a key agreement protocol.

A third design assumption is that these protocols are constant under member substitution: substituting member M_i with a user M_j in the group constitution will only change the protocol execution by substituting keys of the form K_{ik} with K_{jk}. This assumption excludes protocols the definition of which would contain rules such as: "User M_i exponentiates the term intended to M_j with K_{ij}^x where x is the last bit of M_j's identifier" for instance.

2.5 Modelling the GDH-Protocols

The protocols based on the design assumptions we just described will be called *GDH-Protocols*. Before giving a precise definition of this protocol family and starting our analysis, we propose a second example of GDH-Protocol, the Ex-GDH protocol, which we will use to illustrate our further discussion.

Example 2.3 We describe the Ex-GDH protocol in a similar form as the one commonly used in the literature and in [2] for instance. This protocol allows a group of three users M_1, M_2 and M_3 to contributively generate a key $\alpha^{r_1 r_2 r_3}$.

Protocol 2: Ex-GDH Protocol
Let $r_i, \hat{r}_i \in \mathbb{Z}_q^*$ be random values generated by M_i at the beginning of each protocol session. The three group members M_1, M_2 and M_3 then generate the group key by exchanging the following messages:

$$\begin{aligned} M_1 \to M_2 &\ :\ \alpha^{\hat{r}_1}, \alpha^{r_1} \\ M_2 \to M_3 &\ :\ \alpha^{\hat{r}_1 r_2 K_{23}}, \alpha^{r_1 K_{23}}, \alpha^{r_1 r_2} \\ M_3 \to M_1, M_2 &\ :\ \alpha^{\hat{r}_1 r_2 r_3 K_{13}}, \alpha^{r_1 r_3 K_{23}^2} \end{aligned}$$

When executing this protocol, M_1 computes $\alpha^{\hat{r}_1}, \alpha^{r_1}$ and sends these values to M_2, then M_2 exponentiates the first term he received with $r_2 K_{23}$, the second with K_{23} and also with r_2, sends the three resulting elements of $\mathcal{G}$ to M_3; and finally, M_3 exponentiates the first term he received with $r_3 K_{13} K_{23}^{-1}$ and the second with $r_3 K_{23}$.

Upon receipt of the above, M_1 computes the group key $\alpha^{r_1 r_2 r_3}$ from $\alpha^{\hat{r}_1 r_2 r_3 K_{13}}$, M_2 from $\alpha^{r_1 r_3 K_{23}^2}$ and M_3 from $\alpha^{r_1 r_2}$.

We now present our modelling of the GDH-Protocol famils, and start by defining the set of messages which can be exchanged.

Definition 2.4 *Let:*

1. R *be the set of symbols representing random values generated during the protocol execution.*

2. K *be the set of symbols representing the long-term symmetric keys shared by pairs of users. We assume that* $\mathsf{R} \cap \mathsf{K} = \emptyset$ *and call elements of* $\mathsf{R} \cup \mathsf{K}$ atoms*. Furthermore, we denote* K_i *the subset of keys of* K *known by* M_i *and* K_{ij} *a key of* K *known by* M_i *and* M_j*.*

3. $(\mathsf{P}, \cdot)$ *be the commutative group freely generated from* $\mathsf{R} \cup \mathsf{K}$*.*

4. G *be defined from* P *through a bijection* $\mathbf{alphaexp} : \mathsf{P} \rightarrow \mathsf{G}$ *which represents the exponentiation of the public group generator* α *with a product of random values and keys. This set will be used to model the finite group* $\mathcal{G}$*.*

As it can be seen, we do not take any arithmetic relation that could exist between elements of R and K into account. It can also be observed that, according to our definitions, the set G is infinite, while $\mathcal{G}$ is a finite group of prime order. It would be interesting to relate this abstraction of $\mathcal{G}$ with the pseudo-freeness computational assumption introduced by S. Hohenberger and R. Rivest [9, 19].

In order to make the use of these sets more convenient, we introduce the following notations:

Definition 2.5 *Suppose* $p \in \mathsf{P}$ *and* $g \in \mathsf{G}$*.*

1. p_a *denotes the projection of* p *on* a*, that is,* a^e *where* $p = a^e a_1^{e_1} \cdots a_n^{e_n}$*,* $a \notin \{a_1, \ldots, a_n\}$*, and* a*,* a_1*, ...,* a_n *are atoms.*

2. p_{R} *and* p_{K} *denote the projection of* p *on* R *and* K *respectively, that is, are elements of* P *such that* $p = p_{\mathsf{R}} \cdot p_{\mathsf{K}}$ *where* p_{R} *is a product of elements of* R *and* p_{K} *a product of elements of* K*.*

3. $\alpha^p \in \mathsf{G}$ *denotes* $\mathbf{alphaexp}(p)$ *(*α^1 *will usually be abbreviated as* α*).*

4. g^p *denotes* $\mathbf{alphaexp}(\mathbf{alphaexp}^{-1}(g) \cdot p)$

We illustrate these definitions through the following example.

Example 2.6 If we consider our Ex-GDH protocol, $\{r_1, \hat{r}_1, r_2, r_3\} \subset \mathsf{R}$ and $\{K_{13}, K_{23}\} \subseteq \mathsf{K}$; $p = r_1 \cdot r_3 \cdot K_{23}^2$ is an element of P, $p_{\mathsf{R}} = r_1 \cdot r_3$, $p_{\mathsf{K}} = K_{23}^2$, $p_{K_{23}} = K_{23}^2$, $p_{r_2} = 1$ and $\alpha^{r_2 K_{23}}$ is an element of G.

The messages of the protocols we consider are all constituted of sequences of elements of $\mathcal{G}$ (modelled as elements of G). In order to simplify our notations, and since an active attacker has complete control over concatenation, we can model (without loss of generality) the sending (resp. the reception) of the concatenation of n elements of $\mathcal{G}$ as n sending (resp. receptions) of elements of G. So, in our model, all messages are single elements of G. We will call these messages *GDH-Terms*.

In order to describe our protocols, we now exploit the strand-space and bundle definitions, which are given in Appendix A. A strand is a sequence of nodes representing some party's view of a protocol run. Associated with each node is a GDH-Term with a sign, + or −, indicating that the GDH-Term is sent or received, respectively, on that node. The function $term(n)$ (resp. $uns\_term(n)$) provides the signed (resp. unsigned) GDH-Term associated with the node n, while $\langle s, i\rangle$ denotes the i-th node of the strand s and $strand(n)$ the strand to which n belongs. A bundle is a directed graph whose edges express the causal dependencies of the nodes: a "→"-edge connects two nodes whose associated GDH-Terms are of form $+t$ and $-t$, while a "⇒"-edge connects two consecutive nodes of a strand. We also use the notation $n \Rightarrow^+ n'$ to express that n and n' are connected through a sequence of "⇒"-edges and "$\rightarrow_\mathcal{C}$" and "$\Rightarrow_\mathcal{C}$" to denote the "→" and "⇒"-edges of the bundle $\mathcal{C}$. In a bundle, "→" and "⇒"-edges allow defining a partial order relation between nodes: the node n precedes the node n', written $n \prec n'$ if there is a path made of "→" and "⇒"-edges from n to n'. The following example shows a bundle representing a session of our Ex-GDH protocol.

Example 2.7 Let s_1, s_2 and s_3 be three strands representing the roles of M_1, M_2 and M_3 in the Ex-GDH protocol. A bundle containing these three strands is represented in Fig. 2 (all four arrows of the last two rows of this figure originate on nodes of the s_3 strand).

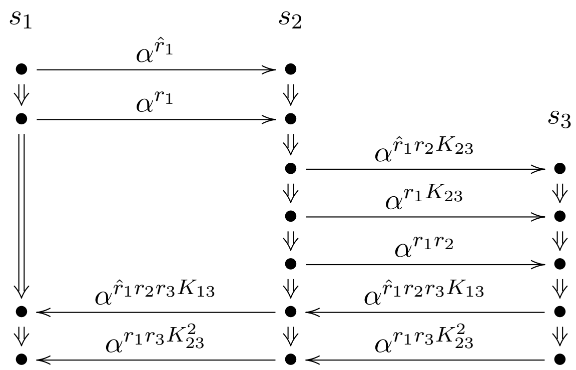


Figure 2: A run of the Ex-GDH protocol

Considering a bundle allows us to understand the way messages are exchanged during a protocol run. However, it does not express how these messages are built, which is an important property for the class of protocols we are analyzing. As explained in the literature concerning the A-GDH protocols [2], the protocols we consider are executed in a very regular way: the group members receive elements of $\mathcal{G}$ and exponentiate these elements with products of known random values and keys in order to construct the messages they send. So, for any element used by a group member to compute his view of the group key, it is possible to write a history describing how this element has been built from the group generator α. This history can be described as a simple path since the combination of two elements of $\mathcal{G}$ into a third one never occurs.

Definition 2.8 *Given a bundle $\mathcal{C}$, a* path π *of length m in $\mathcal{C}$ is a sequence of nodes $\langle n_1, \ldots, n_m \rangle$ of $\mathcal{N}_\mathcal{C}$ such that:*

- $term(n_1) = +t$ *and* $term(n_m) = -t'$
- $(n_{2i+1}, n_{2i+2}) \in \rightarrow_\mathcal{C} \quad (0 \leq i < m/2)$
- $(n_{2i}, n_{2i+1}) \in \Rightarrow^+_\mathcal{C} \quad (0 < j < m/2)$

We furthermore consider that each path has at its extremities two "virtual" nodes n_0 and n_{m+1} which are assumed to belong to the same strands as n_1 and n_m respectively, and the associated GDH-Terms of which are $uns\_term(n_0) = \alpha$ and $uns\_term(n_{m+1}) = \alpha^{r_1 \cdots r_n}$ (for a protocol executed by n parties).

These two virtual nodes (which do not correspond to the transmission of any GDH-Term) are added in order to make the further notations more convenient. They correspond to the fact that the first element of an history is always computed from α and that the last element of an history is used in order to compute a group key $\alpha^{r_1 \cdots r_n}$.

We now introduce a few more definitions about paths:

Definition 2.9 *Consider a path $\pi = \langle n_1, \ldots, n_m \rangle$ in a bundle $\mathcal{C}$*

1. $\pi(i) = n_i$. *As we will often be interested in the end of these paths, we also define* $\pi(\bar{i}) = n_{m+1-i}$
2. $P(\pi(i)) = \frac{\mathbf{alphaexp}^{-1}(uns\_term(\pi(i)))}{\mathbf{alphaexp}^{-1}(uns\_term(\pi(i-1)))}$ *is the element of* P *which must be used to compute* $term(\pi(i))$ *from* $term(\pi(i-1))$
3. $Id(\pi(i)) = M_j$ *where* M_j *is the user executing* $strand(\pi(i))$
4. $length(\pi) = m$

From this definition, $\pi(i)$ is the i-th node of π (starting from the end of π if i is complemented). We may also note that $term(\pi(\bar{1}))$ is the last GDH-Term of π, which will be used for computing the group key, and that $P(\pi(\bar{0}))$, the element of P required for computing $term(\pi(m+1))$ from

$term(\pi(m))$, is the product of random values and long-term keys which will be used to compute the group key from $term(\pi(\bar{1}))$. The length of a path π, denoted $length(\pi)$, is defined as the number of nodes belonging to this path, without including the two virtual nodes.

These notions are exemplified below.

Example 2.10 If we consider the bundle of Example 2.7, a path π describing the history of $\langle s_2, 7\rangle$ is

$$\pi = \langle\langle s_1, 2\rangle, \langle s_2, 2\rangle, \langle s_2, 4\rangle, \langle s_3, 2\rangle, \langle s_3, 5\rangle, \langle s_2, 7\rangle\rangle$$

This path is represented on Fig. 3, on which we added the virtual nodes at the beginning of s_1 and at the end of s_2.

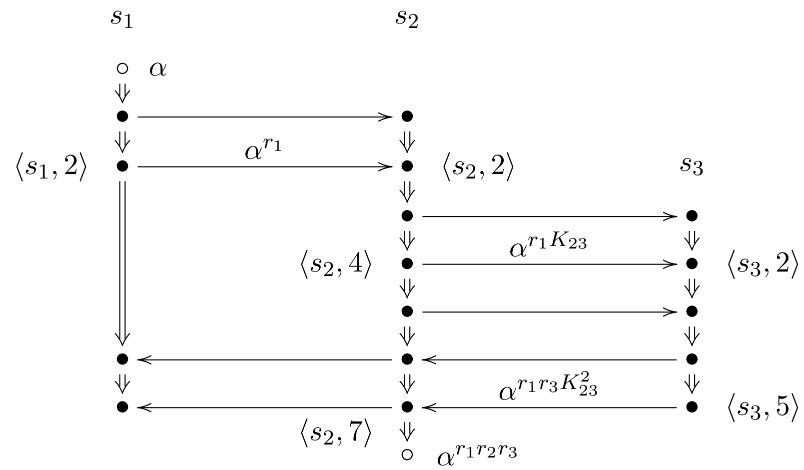


Figure 3: Example of Path

$$term(\pi(1)) = +\alpha^{r_1},\ term(\pi(\bar{4})) = +\alpha^{r_1 K_{23}}$$

$$P(\pi(1)) = r_1,\ P(\pi(2)) = 1,\ P(\pi(\bar{0})) = r_2 K_{23}^{-2}$$

$$strand(\pi(2)) = s_2,\ Id(\pi(6)) = M_2,\ length(\pi) = 6$$

As paths will be used in order to describe the way messages are transformed along strands, they can also be used to define a notion of knowledge expressing that a party must know specific values in order to be able to perform the transformation required at some node. This notion will make use of the subterm relation $\sqsubset$ defined as follows:

Definition 2.11 *Suppose a is an atom, $p \in \mathsf{P}$ and $g \in \mathsf{G}$.*

- $a \sqsubset p$ *if* $p_a \neq 1$

- $a \sqsubset g$ *iff* $a \sqsubset \mathbf{alphaexp}^{-1}(g)$

If $a \sqsubset x$*, we say that* a *is a subterm of* x *or that* x *contains* a*.*

The following example illustrates this definition.

Example 2.12 Let $g = \alpha^{r_1 K_{23}}$ be an element of G. Then $r_1 \sqsubset g$ and $K_{13} \not\sqsubset g$.

We can now define our notions of knowledge.

Definition 2.13 *Consider a set* $\pi = \{\pi_1, \ldots, \pi_n\}$ *of paths in* $\mathcal{C}$*. We say that:*

1. $p \in \mathsf{P}$ *is* known *on* $\pi_i(j)$ *iff for any atom* $a \sqsubset p$*, we have that* $a \sqsubset P(\pi_i(j))$*,*
2. $p \in \mathsf{P}$ *is* known *on the strand* s *iff for any atom* $a \sqsubset p$*, there are values for* i *and* j *such that* $a \sqsubset P(\pi_i(j))$ *and* $strand(\pi_i(j)) = s$*,*
3. $p \in \mathsf{P}$ *is* locally known *on the strand* s *of* $\mathcal{C}$ *if* s *is the only strand of* $\mathcal{C}$ *on which* p *is known.*

Example 2.14 If we consider our Ex-GDH protocol as represented in Example 2.7 and the three paths π_1, π_2 and π_3 defined as

$$\begin{aligned}
\pi_1 &= \langle\langle s_1, 1\rangle, \langle s_2, 1\rangle, \langle s_2, 3\rangle, \langle s_3, 1\rangle, \langle s_3, 4\rangle, \langle s_1, 3\rangle\rangle \\
\pi_2 &= \langle\langle s_1, 2\rangle, \langle s_2, 2\rangle, \langle s_2, 4\rangle, \langle s_3, 2\rangle, \langle s_3, 5\rangle, \langle s_2, 7\rangle\rangle \\
\pi_3 &= \langle\langle s_1, 2\rangle, \langle s_2, 2\rangle, \langle s_2, 5\rangle, \langle s_3, 3\rangle\rangle
\end{aligned}$$

then $r_3 K_{13}$ is known on $\pi_1(5)$. We also have that K_{23} is known on s_3 but not locally known on that strand as it is also known on s_2. Finally, r_2 is locally known on s_2 since it is only known on $\pi_1(3)$, on $\pi_2(7)$ and on $\pi_3(3)$ which all belong to s_2 (note that $\pi_2(7)$ is the virtual node at the end of π_2).

Equipped with these definitions, we can now define the GDH-Protocols. The first part of this definition expresses how GDH-Terms are exchanged, which is described through a bundle containing n strands; while the second part expresses how the exchanged GDH-Terms are computed by the different group members, which is expressed through n paths whose final element is the element of G used by the different group members to compute their view of the group key. The last part of this definition expresses constraints on the use of the different protocol variables (random key contributions and long-term keys). The first constraint expresses that the random values generated by the protocol participants can be used on only one strand: they are never transmitted to other users, and the probability of two users generating the same random value can be considered negligible. The second constraint expresses that each contribution r_i included in the group key $\alpha^{r_1 \cdots r_n}$ must be generated by the user M_i. Finally, the last constraint expresses that the long-term symmetric keys can only be used by the two users who are assumed to know them.

Definition 2.15 *A* GDH-Protocol *on a group of n principals $\mathsf{M} = \{M_1, \ldots, M_n\}$ is a protocol aiming at enabling a key $\alpha^{r_1 \cdots r_n}$ to be shared by the principals in M and the regular execution of which can be described through two elements:*

1. *a bundle $\mathcal{C}_{GDH}$ containing n strands $s_1, \ldots, s_n$, M_i being the active principal for s_i.*

2. *a set $\pi = \{\pi_1, \ldots, \pi_n\}$ of n paths in $\mathcal{C}_{GDH}$. These specific paths are called* histories *and express how the GDH-Terms exchanged in the $\mathcal{C}_{GDH}$ bundle are built: $Id(\pi_i(2j))$ computes $term(\pi_i(2j+1))$ from $term(\pi_i(2j))$ and M_i computes the group key $\alpha^{r_1 \cdots r_n}$ from $term(\pi_i(\bar{1}))$ (so, $strand(\pi_i(\bar{1})) = s_i$).*

   Furthermore, in a GDH-Protocol,

a. *If $a \in \mathsf{R}$ is known on $\pi_i(j)$ then it is locally known*

b. *The random contribution $r_i \in \mathsf{R}$ is locally known on s_i*

c. *If $a \in \mathsf{K}$ is known on $\pi_i(j)$ then $a \in \mathsf{K}_{Id(\pi_i(j))}$*

We may verify that the Ex-GDH protocol, defined by the bundle represented in Fig. 2 and by the three histories given by the paths in Example 2.14, is a GDH-Protocol.

From now on, except when specified otherwise, we always refer to GDH-Protocols executed by a group $\mathsf{M} = \{M_1, \ldots, M_n\}$ and described through a GDH-Bundle $\mathcal{C}_{GDH}$ and through histories $\pi_1, \ldots, \pi_n$.

We introduce one last (and central) definition before turning to properties of GDH-Protocols: the definition of the notion of *contribution*. Given a session of a GDH-Protocol, the product of the elements of P used by M_i in order to exponentiate elements of G belonging to π_j will be called the contribution of M_i to M_j, denoted $C(M_i \to M_j)$. In other words, $C(M_i \to M_j)$ is the product of the services offered by M_i on the path π_j.

Definition 2.16 *Given a GDH-Protocol with paths $\pi_1, \ldots, \pi_n$, the* contribution *of M_i to M_j, denoted $C(M_i \to M_j)$, is defined as*

$$\prod_{0 < k \leq length(\pi_j)} P(\pi_j(k))^{e_k}$$

where $e_k = 1$ if $Id(\pi_j(k)) = M_i$ and $e_k = 0$ otherwise.

Example 2.17 The table below indicates the value of $C(M_i \to M_j)$ for the Ex-GDH protocol in the line M_i of column M_j.

	M_1	M_2	M_3
M_1	$\hat{r}_1$	r_1	r_1
M_2	r_2K_{23}	K_{23}	r_2
M_3	$r_3K_{13}K_{23}^{-1}$	r_3K_{23}	1

3 Properties of GDH-Protocols

3.1 Introduction

Starting from the definitions of the previous section, we now define a few constitutive properties of GDH-Protocols. These properties express characteristics that GDH-Protocols must respect if they conform to their definition. The goal of this section will be to achieve our first attack phase described in Section 2.3, that is, to obtain a relation expressing the value that each user M_i executing a session of a GDH-Protocol uses for computing his view of the group key (i.e. $P(\pi_i(\bar{0}))$) as a product of contributions (which are themselves a product of services) and of long-term keys the attacker knows.

In the following paragraphs, we will never precisely specify to which session of a protocol we refer: we simply state the corresponding group constitution when it is different from $\mathsf{M} = \{M_1, \ldots, M_n\}$. This is because we will always consider a single protocol execution for each specified group constitution.

3.2 Properties of GDH-Protocols

We start this subsection with two simple observations which will be useful further.

Observation 3.1 *Let p_1, p_2 and p_3 be elements of P and a be an atom. If $p_1 = p_2 \cdot p_3$ and $a \sqsubset p_1$, then $a \sqsubset p_2$ or $a \sqsubset p_3$. Similarly, If $p_1 = p_2 \cdot p_3$ and $(p_1)_a = (p_2)_a$ then $a \not\sqsubset p_3$.*

These observations result from the definition of P.

Observation 3.2 *From the definition of the notion of contribution,*

$$term(\pi_j(\bar{1})) = -\alpha^{\prod_{i=1\ldots n} C(M_i \to M_j)}.$$

This observation can be verified in Example 2.17 if we keep in mind that $term(\pi_1(\bar{1})) = -\alpha^{\hat{r}_1 r_2 r_3 K_{13}}$, $term(\pi_2(\bar{1})) = -\alpha^{r_1 r_3 K_{23}^2}$ and $term(\pi_3(\bar{1})) = -\alpha^{r_1 r_2}$.

We can now write a first proposition about the value of $C_{\mathsf{R}}(M_i \to M_j)$ when $i \neq j$, where $C_{\mathsf{R}}(M_i \to M_j)$ denotes the projection of $C(M_i \to M_j)$ on the free abelian group generated from R (as given in Def. 2.5).

Proposition 3.3 *For any GDH-Protocol, if* $1 \leq i, j \leq n$ *and* $i \neq j$*, then* $C_{\mathsf{R}}(M_i \to M_j) = r_i$.

Proof. From Observation 3.2, we can write

$$\prod_{i=1\ldots n} C_{\mathsf{R}}(M_i \to M_j) \cdot P_{\mathsf{R}}(\pi_j(\bar{0})) = r_1 \cdots r_n. \qquad (1)$$

We observe that $r_i \not\sqsubset C_{\mathsf{R}}(M_k \to M_j)$ when $k \neq i$ or else r_i would be known on s_k which is in contradiction with Point (b) of Def. 2.15 of GDH-Protocols. Furthermore, $r_i \not\sqsubset P_{\mathsf{R}}(\pi_j(\bar{0}))$ for the same reason. We can deduce from these remarks and from Observation 3.1 that $r_i \sqsubset C_{\mathsf{R}}(M_i \to M_j)$ and that $C_{r_i}(M_i \to M_j) = (r_1 \cdots r_n)_{r_i} = r_i$.

Let us now imagine that $C_{\mathsf{R}}(M_i \to M_j) = r_i \cdot p$. Then $r_i \not\sqsubset p$. Suppose $r_a \sqsubset p$. From Observation 3.1, $r_a \sqsubset C_{\mathsf{R}}(M_i \to M_j)$, so $r_a \in \mathsf{R}$ is locally known on s_i (from Point (a) of Def. 2.15). Therefore, it is not known on s_k when $k \neq i$, $r_a \notin \{r_1, \ldots, r_n\}$, $r_a \not\sqsubset C_{\mathsf{R}}(M_k \to M_j)$ $(k \neq i)$ and $r_a \not\sqsubset P_{\mathsf{R}}(\pi_j(\bar{0}))$. But this is in contradiction with Observation 3.1 and Equation (1) and we must have that $p = 1$. ∎

Concerning the value of $C_{\mathsf{R}}(M_i \to M_i)$, the following relation must be valid:

Proposition 3.4 *For any GDH-Protocol,* $C_{\mathsf{R}}(M_i \to M_i) = r_i \cdot P_{\mathsf{R}}(\pi_i(\bar{0}))^{-1}$.

Proof. Definition 2.9 gives us that

$$P(\pi_i(\bar{0})) = \prod_{j=1\ldots n} r_j \cdot (\mathbf{alphaexp}^{-1}(uns\_term(\pi_i(\bar{1}))))^{-1}$$

So, by successively exploiting Observation 3.2 and Proposition 3.3, we can write:

$$\begin{aligned}
P_{\mathsf{R}}(\pi_i(\bar{0})) &= \prod_{j=1\ldots n} r_j \cdot \left(\prod_{j=1\ldots n} C_{\mathsf{R}}(M_j \to M_i) \right)^{-1} \\
&= \prod_{j=1\ldots n} r_j \cdot \left(\prod_{j=1\ldots n,\ j\neq i} r_j \right)^{-1} \cdot C_{\mathsf{R}}(M_i \to M_i)^{-1} \\
&= r_i \cdot C_{\mathsf{R}}(M_i \to M_i)^{-1}
\end{aligned}$$

∎

These two propositions can be checked for the Ex-GDH protocol in the tables of Example 2.17.

Having characterized the value of $C_{\mathsf{R}}(M_i \to M_j)$, we will now write two propositions concerning the value of $C_{\mathsf{K}}(M_i \to M_j)$.

Proposition 3.5 *For any GDH-Protocol, if* $C_{K_{jk}}(M_j \to M_i) = K_{jk}^a$ $(K_{jk} \notin \mathsf{K}_i)$ *then* $C_{K_{jk}}(M_k \to M_i) = K_{jk}^{-a}$.

Proof. Observation 3.2 gives us that $\prod_{l=1\ldots n} C_{\mathsf{K}}(M_l \to M_i) \cdot P_{\mathsf{K}}(\pi_i(\bar{0})) = 1$; so the sum of the powers of K_{jk} in the components of the left part of this equation must be null. But $K_{jk} \not\sqsubset P_{\mathsf{K}}(\pi_i(\bar{0}))$ since $K_{jk} \notin \mathsf{K}_i$. Just as $K_{jk} \not\sqsubset C_{\mathsf{K}}(M_l \to M_i)$ $(l \neq j, k)$ since $K_{jk} \notin \mathsf{K}_l$. Therefore, K_{jk} can only be a subterm of $C_{\mathsf{K}}(M_j \to M_i)$ and of $C_{\mathsf{K}}(M_k \to M_i)$, and the powers of K_{jk} in these two contributions must be of the form a and $-a$ since their sum is null. ∎

Until now, we considered the relations between values inside one session of a protocol. Now, we would like to write a proposition concerning the use of long-term keys in sessions executed by different groups of users. To this purpose, we introduce a substitution operator: if $p \in \mathsf{P}$ is such that $p_{\mathsf{R}} = 1$ and is a function of the keys of a bundle corresponding to a session of a GDH-Protocol executed by the group M, then $[M_i \backslash M_I : p]$ (where $M_i \in \mathsf{M}$ and $M_I \notin \mathsf{M}$) refers to the value that p would have in a session with the same participants except that M_i is substituted with M_I. More precisely:

Definition 3.6 *If* $p = \prod_j K_{ij}^{e_{ij}} \cdot K_x$ *where* $K_{ij} \not\sqsubset K_x$ $(\forall j)$ *then* $[M_i \backslash M_I : p] = \prod_j K_{Ij}^{e_{ij}} \cdot K_x$. *More generally, if* $\mathsf{S} = \{M_{i_1}, \ldots M_{i_s}\}$, $[\mathsf{S} \backslash M_I : p] = [M_{i_1} \backslash M_I : [(\mathsf{S} - \{M_{i_1}\}) \backslash M_I : p]]$.

Example 3.7 In the Ex-GDH protocol, $[M_1 \backslash M_I : C_{\mathsf{K}}(M_3 \to M_1)] = K_{I3} K_{23}^{-1}$ and $[\{M_1, M_2\} \backslash M_I : C_{\mathsf{K}}(M_3 \to M_1)] = K_{I3} K_{I3}^{-1} = 1$

As above, M_I denotes a user that is not a member of the group M and plays the role of the intruder. This user is however considered as a legitimate member of some other groups, $K_{Ij} \in (\mathsf{K}_I \cap \mathsf{K}_j)$ denoting a long-term key shared by M_I and M_j.

We can now write a proposition relating the key part of the contribution of a honest member M_j to M_i, i.e. $C_{\mathsf{K}}(M_j \to M_i)$, with his contribution $[\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)]$ in a session where a set of honest members $\mathsf{S} \subset \mathsf{M}$ has been replaced with the intruder. These two values are in fact equal, excepted that all occurrences of keys shared between M_j and users in S will be replaced by keys shared between M_j and M_I.

Proposition 3.8 *Suppose* $\mathsf{S} \subset \mathsf{M}$ *and* $M_j \notin \mathsf{S}$. *Then,*

$$\begin{aligned} C_{\mathsf{K}}(M_j \to M_i) &= [\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] \\ &\quad \cdot \prod_{M_k \in \mathsf{S}} C_{K_{jk}}(M_j \to M_i) \\ &\quad \cdot \prod_{M_k \in \mathsf{S}} [\mathsf{S} \backslash M_I : C_{K_{jk}}^{-1}(M_j \to M_i)]. \end{aligned}$$

Proof. $C_{\mathsf{K}}(M_j \rightarrow M_i)$ is known on s_j, so it can be written as a product of keys of the form K_{jx}. A possible way to write $C_{\mathsf{K}}(M_j \rightarrow M_i)$ is therefore $\prod_{M_k \in \mathsf{S}} K_{jk}^{e_k} \cdot K_x$ where $K_{jk} \not\sqsubseteq K_x$ for all $M_k \in \mathsf{S}$ and K_x is a product of keys in K_j. Definition 3.6 now implies that $[\mathsf{S}\backslash M_I : C_{\mathsf{K}}(M_j \rightarrow M_i)] = \prod_{M_k \in \mathsf{S}} K_{jI}^{e_k} \cdot K_x$.

This proposition now results from the fact that $C_{K_{jk}}(M_j \rightarrow M_i) = K_{jk}^{e_k}$ and $[M_k\backslash M_I : C_{K_{jk}}(M_j \rightarrow M_i)] = K_{jI}^{e_k}$.

∎

Example 3.9 Consider the Ex-GDH protocol and $\mathsf{S} = \{M_1\}$. In this case, $C_{\mathsf{K}}(M_3 \rightarrow M_1) = K_{13}K_{23}^{-1}$, $[M_1\backslash M_I : C_{\mathsf{K}}(M_3 \rightarrow M_1)] = K_{I3}K_{23}^{-1}$, $C_{K_{13}}(M_3 \rightarrow M_1) = K_{13}$ and $[M_1\backslash M_I : C_{K_{13}}(M_3 \rightarrow M_1)] = K_{I3}$. Then we can check that $K_{13}K_{23}^{-1} = K_{I3}K_{23}^{-1} \cdot K_{13} \cdot K_{I3}^{-1}$ as expected from our previous proposition.

All these propositions can be used to prove our main property concerning contributions: the product $P(\pi_i(\bar{0}))$ that user M_i uses when computing his view of the group key can be written as a product of contributions and keys that the intruder knows.

Theorem 3.10 *For any GDH-Protocol executed by a group of users* $\mathsf{M} = \{M_1, \ldots, M_n\}$ *where* $n \geq 3$, *it is possible to write any secret* $P(\pi_i(\bar{0}))$ *as a product of contributions* $C(M_j \rightarrow M_k)$ $(M_j, M_k \in \mathsf{M} \cup \{M_I\})$ *and of keys known by* M_I.

Proof. (*See Appendix B for details*)

Let S_j and S_k be two disjoint sets of users such that $M_k \in \mathsf{S}_j$, $M_j \in \mathsf{S}_k$, $M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$ and $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$. Then, it can be checked that:

$$\begin{aligned} P(\pi_i(\bar{0})) = \; & C^{-1}(M_i \rightarrow M_i) \cdot C(M_i \rightarrow M_j) \\ & \cdot [\mathsf{S}_j\backslash M_I : C^{-1}(M_i \rightarrow M_j) \cdot C(M_i \rightarrow M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j\backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k\backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_j)] \\ & \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l} \end{aligned}$$

This relation was obtained from the observation that $P_{\mathsf{R}}(\pi_i(\bar{0})) = C_{\mathsf{R}}^{-1}(M_i \rightarrow M_i) \cdot C_{\mathsf{R}}(M_i \rightarrow M_j)$, that $P_{\mathsf{K}}(\pi_i(\bar{0})) = \prod_{M_l \in \mathsf{M}} C_{\mathsf{K}}^{-1}(M_l \rightarrow M_i)$ and from the use of the previous propositions. ∎

Example 3.11 If we consider the Ex-GDH Protocol with $i = 1$, $j = 2$ and $k = 3$, we must choose $\mathsf{S}_j = \{M_k\}$ and $\mathsf{S}_k = \{M_j\}$ and, applying

Theorem 3.10, we have that:

$$\begin{aligned} r_1\hat{r}_1^{-1}K_{13}^{-1} &= \hat{r}_1^{-1}\cdot r_1 \\ &\cdot (r_1')^{-1}\cdot r_1' \\ &\cdot (r_2'K_{2I})^{-1}\cdot r_2' \\ &\cdot (r_3''K_{13}K_{I3}^{-1})^{-1}\cdot r_3''K_{I3} \\ &\cdot K_{I3}^{-2}\cdot K_{2I}. \end{aligned}$$

where we considered r_i to be contributions sent in the session executed by the group $\{M_1, M_2, M_3\}$, r_i' to be contributions in the session executed by the group $\{M_1, M_2, M_I\}$, and r_i'' to be contributions in the session executed by the group $\{M_1, M_I, M_3\}$.

3.3 Conclusion

We have shown that, for any GDH-Protocol executed by at least three users, it is possible to write the secret value that each group member will use when computing the group key (i.e. $P(\pi_i(\bar{0}))$) as a product of contributions of different group members during different sessions of the protocol.

In other words, we have shown that the first phase of Section 2.3's attack process can always succeed, provided that we consider at least three group members and some well chosen protocol sessions.

A natural question comes about the security of two-party GDH-Protocols. In fact, it is easy to exhibit a GDH-Protocol for which this first attack phase cannot succeed: the A-GDH.2 protocol with two participants. With our usual notations, this protocol executes as described in Fig. 4.

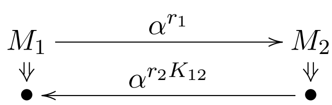


Figure 4: A-GDH.2 Protocol with two parties

Considering the role of M_1, we try to write $P(\pi_1(\bar{0})) = r_1K_{12}^{-1}$ as a product of services and keys the adversary knows. The only service containing r_1 is the service provided by M_1 during the session we try to attack, so we have no choice but using that service. The key part of $P(\pi_1(\bar{0}))$, i.e. K_{12}^{-1}, appears only in services provided by users in sessions executed by the group $\{M_1, M_2\}$ and in sessions executed by the group $\{M_2, M_1\}$. However, this key always comes together with a random value (r_2 in Fig. 4) which only appears in that service, so we will never find another service allowing us to cancel it. This shows that it is impossible to write $P(\pi_1(\bar{0}))$ as a product of services and keys known by the adversary when we only consider the services provided during sessions of the A-GDH.2 protocol with two participants.

It can be easily verified that the same problem occurs with the role of M_2.

4 Collecting Contributions

4.1 Introduction

At this point, we know that the first phase of our attack process described in Section 2.3 always succeed for protocols executed by at least three parties.

We come now to the second phase and will now see how (and if) the required services can be exploited by an intruder who wants to undermine the IKA property. More precisely, we will examine how an intruder who wants to attack M_i can obtain a pair (g_1, g_2) of elements of G such that $g_2 = g_1^{P(\pi_i(\bar{0}))}$ and send g_1 to the node $\pi_i(\bar{1})$.

4.2 Collecting Pairs of Contributions

If we look at Theorem 3.10, we can observe that we are interested in collecting pairs (g_1, g_2) such that $g_2 = g_1^p$ where p is a product of terms of the form $C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)$. The following proposition is a first step in the obtention of such pairs.

Proposition 4.1 *For any session of a GDH-Protocol executed by a group of users* M *of cardinality* n*, an active attacker can obtain a pair* (g_1, g_2) *of elements of* G *such that* $g_2 = g_1^{C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)}$.

Proof. Consider a session of a GDH-Protocol executed by the members of the group M. If we initialize g_1 and g_2 to α, Algorithm 1 gives the intruder a pair (g_1, g_2) of the desired form (actually, in order to prevent the message recipient to observe that the message he receives is simply the group generator α, we also can initialize g_1 and g_2 to any random elements of G, say α^x and α^y, and, at the end of the algorithm, exponentiate g_1 with x^{-1} and g_2 with y^{-1}).

This algorithm can be explained as follows. Let s_i be a strand that corresponds to M_i's role in an execution by the group M of the considered protocol. We proceed by constructing a strand s_I matching s_i (i.e. a strand such that $term(\langle s_i, x\rangle) = -term(\langle s_I, x\rangle)$), while collecting the services on π_j belonging to s_i into g_1 and the services on π_k belonging to s_i into g_2 (excepted for the common parts of π_j and π_k). So, by executing this strand, the intruder will have a conversation with M_i at the end of which M_i will have completed his role in the considered session of the protocol without interacting with any other member of M.

The s_I strand is constructed by receiving the messages M_i sends and by sending a random element of G when M_i is waiting for a message, except

Algorithm 1 Defines a strand s_I which, when executed together with s_i, provides a pair (g_1, g_2) such that $g_2 = g_1^{C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)}$ $(M_j \neq M_k)$ if the precondition $g_1 = g_2$ is verified.

for $z := 1$ to $length(s_i)$ **do**
  if $\exists t : term(\langle s_i, z \rangle) = +t$ **then**
    $term(\langle s_I, z \rangle) := -t$
    if $\exists x : \langle s_i, z \rangle = \pi_j(x)$ and $\pi_j(x) \neq \pi_k(x)$ **then**
      $g_1 := uns\_term(\pi_j(x))$
    end if
    if $\exists y : \langle s_i, z \rangle = \pi_k(y)$ and $\pi_j(y) \neq \pi_k(y)$ **then**
      $g_2 := uns\_term(\pi_k(y))$
    end if
  else
    $t :=$ a random element of G
    if $\exists x : \langle s_i, z \rangle = \pi_j(x)$ and $\pi_j(x+1) \neq \pi_k(x+1)$ **then**
      $t := g_1$
    end if
    if $\exists y : \langle s_i, z \rangle = \pi_k(y)$ and $\pi_j(y+1) \neq \pi_k(y+1)$ **then**
      $t := g_2$
    end if
    $term(\langle s_I, z \rangle) = +t$
  end if
end for

when the considered nodes of s_i are nodes of the histories π_j or π_k. In this last case, different actions are performed according to the sign of the term on the considered node of s_i (to which we will refer as n) and the histories we consider:

- if
  - $term(n)$ is negative and
  - n belongs to π_j (resp. π_k), that is, $term(n)$ is the input of a service that is part of $C(M_i \to M_j)$ (resp. $C(M_i \to M_k)$) and
  - the next node on π_j (resp. on π_k) is not part of both π_j and π_k (i.e. the output of the corresponding service(s) is not part of both π_j and π_k)

  then the term in the corresponding node of s_I is set to g_1 (resp. g_2), that is, the intruder provides g_1 (resp. g_2) as input for this service
- if
  - $term(n)$ is positive and
  - n belongs to π_j (resp. π_k) (that is, $term(n)$ is the output of a service that is part of $C(M_i \to M_j)$ (resp. $C(M_i \to M_k)$)) and

– the output of the considered service is not part of both π_j and π_k,

then the intruder assigns the output of this service to g_1 (resp. g_2).

In fact, when $n = \pi_j(x)$ (resp. $n = \pi_k(x)$), this process allows to perform the operation $g_1 := g_1^{P(\pi_j(x+1))}$ (resp. $g_2 := g_2^{P(\pi_k(x+1))}$), which eventually provides the expected values, as it can be checked from the definition of contributions (Def. 2.16). ∎

Example 4.2 We apply Algorithm 1 in order to obtain a pair (g_1, g_2) such that $g_2 = g_1^{C^{-1}(M_2 \to M_2)\cdot C(M_2 \to M_3)}$ in our Ex-GDH protocol. For that protocol,

$$\begin{aligned} \pi_2 &= \langle\langle s_1, 2\rangle, \langle s_2, 2\rangle, \langle s_2, 4\rangle, \langle s_3, 2\rangle, \langle s_3, 5\rangle, \langle s_2, 7\rangle\rangle \\ \pi_3 &= \langle\langle s_1, 2\rangle, \langle s_2, 2\rangle, \langle s_2, 5\rangle, \langle s_3, 3\rangle\rangle \end{aligned}$$

where s_1, s_2 and s_3 are executed by M_1, M_2 and M_3 respectively.

Our algorithm assumes g_1 and g_2 have been initialized to α and successively considers all the nodes of s_2 in order to build s_I, the variable z indicating the index of the node of s_i which is examined.

$z = 1$ $term(\langle s_2, 1\rangle)$ is negative, so we define $t := \langle \alpha^r \rangle$ (where α^r is a random element of G). The next two tests are false, so $term(\langle s_I, 1\rangle) := +t$.

$z = 2$ $term(\langle s_2, 2\rangle)$ is also negative but $\langle s_2, 2\rangle$ is part of both π_2 and π_3, so $term(\langle s_I, 2\rangle)$ is defined as $+\alpha$, that is, the value to which g_2 has been initialized.

$z = 3$ $term(\langle s_2, 3\rangle)$ is positive, so we define $term(\langle s_I, 3\rangle) := -t$. The next two tests are false.

$z = 4$ $term(\langle s_2, 4\rangle)$ is positive, so we define $term(\langle s_I, 4\rangle) := -t$, where $t = \langle \alpha^{K_{23}} \rangle$. Since the choice $x = 3$ matches the first **if** clause, we update the value of g_1 to $\alpha^{K_{23}}$.

$z = 5$ $term(\langle s_2, 5\rangle)$ is positive, so we define $term(\langle s_I, 5\rangle) := -t$, where $t = \langle \alpha^{r_2} \rangle$. Since the choice $y = 3$ matches the first **if** clause, we update the value of g_2 to α^{r_2}.

$z = 6$ $term(\langle s_2, 6\rangle)$ is negative, and $\langle s_2, 6\rangle$ does not belong to π_2 nor π_3, so we define $term(\langle s_I, 6\rangle) := +\alpha^r$.

$z = 7$ $term(\langle s_2, 7\rangle)$ is also negative, but $\langle s_2, 7\rangle$ is part of π_2, so we define $term(\langle s_I, 7\rangle) := +\alpha^{K_{23}}$.

We can easily verify that $g_2 = g_1^{r_2 K_{23}^{-1}} = g_1^{C^{-1}(M_2 \to M_2)\cdot C(M_2 \to M_3)}$ as expected. The strands s_2 and s_I are represented in Fig. 5.

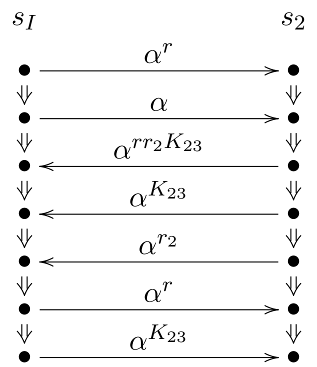


Figure 5: Representation of s_I and s_2

4.3 Composing Contributions

In the previous subsection, we have shown that we can obtain pairs (g_1, g_2) of elements of G such that $g_2 = g_1^p$ where p is a product of terms of the form $C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)$. Now, we would like to be able to reuse Algorithm 1 with the obtained values of g_1 and g_2 as starting values in order to obtain a pair of the more complex form described in Theorem 3.10.

Unfortunately, this is not always possible, as we will show through the following example.

Example 4.3 We introduce a new protocol, which we call the *Tri-GDH* protocol. This protocol can be defined through three strands and three histories:

Protocol 3: Tri-GDH Protocol

$$\begin{array}{rcl}
s_1 & = & \langle +\alpha^{r_1}, -\alpha^{r_3}, +\alpha^{r_1 r_3 K_{12}}, -\alpha^{r_2 r_3 K_{13}} \rangle \\
s_2 & = & \langle +\alpha^{r_2}, -\alpha^{r_1}, +\alpha^{r_1 r_2 K_{23}}, -\alpha^{r_1 r_3 K_{12}} \rangle \\
s_3 & = & \langle +\alpha^{r_3}, -\alpha^{r_2}, +\alpha^{r_2 r_3 K_{13}}, -\alpha^{r_1 r_2 K_{23}} \rangle \\
\pi_1 & = & \langle \langle s_2, 1 \rangle, \langle s_3, 2 \rangle, \langle s_3, 3 \rangle, \langle s_1, 4 \rangle \rangle \\
\pi_2 & = & \langle \langle s_3, 1 \rangle, \langle s_1, 2 \rangle, \langle s_1, 3 \rangle, \langle s_2, 4 \rangle \rangle \\
\pi_3 & = & \langle \langle s_1, 1 \rangle, \langle s_2, 2 \rangle, \langle s_2, 3 \rangle, \langle s_3, 4 \rangle \rangle
\end{array}$$

A run of this protocol is represented in Fig. 6 (we do not used the classical strand notation in order to avoid crossing arrows). During the first round of the protocol, the three central messages are exchanged, while the three external ones are computed from those just received and sent during the second round.

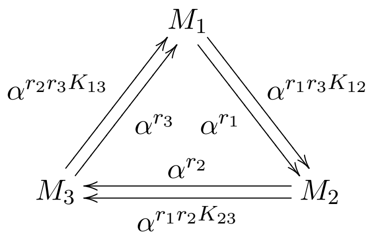


Figure 6: A run of the Tri-GDH protocol

An application of Theorem 3.10 for this protocol with $i = 1$, $j = 2$ and $k = 3$ gives:

$$\begin{aligned} r_1 \cdot K_{13}^{-1} &= 1 \cdot r_1 K_{12} \cdot (r_1' K_{12})^{-1} \cdot r_1' \\ &\quad \cdot {r_2'}^{-1} \cdot r_2' K_{2I} \cdot (r_3'' K_{13})^{-1} \cdot r_3'' \cdot K_{2I}^{-1} \end{aligned}$$

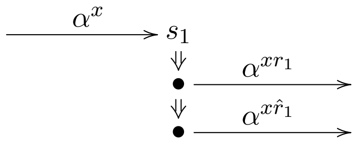
where r_i, r_i', r_i'' represent random values generated during three sessions of the protocol, the participants of these sessions being $\{M_1, M_2, M_3\}$, $\{M_1, M_2, M_I\}$ and $\{M_1, M_I, M_3\}$ respectively.

Among these contributions we first consider r_1', ${r_2'}^{-1}$ and r_3''. These three services are provided as first elements of histories: the values $\alpha^{r_1'}$, $\alpha^{r_2'}$ and $\alpha^{r_3''}$ are provided independently of any input value that the intruder could choose. Unfortunately, in order to build a pair (g_1, g_2) such that $g_2 = g_1^p$ where $p = r_1' {r_2'}^{-1} r_3''$, we would need to submit $\alpha^{r_1'}$ as input of the r_3''-service or, conversely, to submit $\alpha^{r_3''}$ as input of the r_1'-service, which is unfortunately impossible.

Guided by this example, we can more generally observe that we are usually unable to compose two contributions containing initial parts of the corresponding histories if we have to collect these contributions in the same variable, which occurs when their powers have the same sign in the expression of $P(\pi_i(\bar{0}))$. This problem could not occur in the case we considered in the previous section since we had to collect contributions with exponents of different signs.

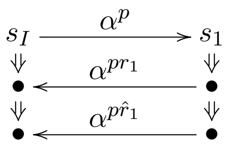
Another kind of services can be problematic: if two services have the same input and two distinct outputs, we may observe that $\pi_j(x) = \pi_k(y)$ for these service's input and that the corresponding element of the GDH-Term t will be affected twice in Algorithm 1. This was not a problem when the precondition $g_1 = g_2$ was verified, but becomes awkward when we try to reuse this algorithm in order to build more complex pairs. Indeed we will lose any non trivial relation that could exist between g_1 and g_2 before starting Algorithm 1. We illustrate this problem in the following example.

Example 4.4 Suppose we applied Algorithm 1 and obtained two values $g_1 = \alpha$ and $g_2 = \alpha^p$. Now, we would like to reuse the same algorithm with the product of contributions $C^{-1}(M_1 \to M_2) \cdot C(M_1 \to M_3)$ (in order to obtain a pair (g_1, g_2) where $g_2 = g_1^{p \cdot C^{-1}(M_1 \to M_2) \cdot C(M_1 \to M_3)}$), the strand s_1 being defined as



and given that $\pi_2(2) = \pi_3(2) = \langle s_1, 1\rangle$, $\pi_2(3) = \langle s_1, 2\rangle$ and $\pi_3(3) = \langle s_1, 3\rangle$.

Applying Algorithm 1 anew provides the following conversation:



The resulting pair will be $(g_1, g_2) = (\alpha^{pr_1}, \alpha^{p\hat{r}_1})$, so we will have $g_2 = g_1^{r_1^{-1}\hat{r}_1}$ instead of the relation $g_2 = g_1^{pr_1^{-1}\hat{r}_1}$ we need to obtain.

We now define the notions of *starting* and *splitting* points of histories, which will be useful to describe the problems we just described. We will use these notions in the next proposition, which gives sufficient conditions for products of contributions to be collectible by the intruder.

Definition 4.5 *Consider a GDH-Protocol executed by n participants and let $s_1, \ldots, s_n$ be the n strands and $\pi_1, \ldots, \pi_n$ be the n histories given in this protocol's definition. We define* $\text{start}(\pi_i)$ *as the first node of π_i, that is, $\pi_i(1)$.*

We then say that the product of contributions $\prod_{i \in \mathcal{I}} C^{e_i}(M_{j_i} \to M_{k_i})$ (with $\mathcal{I}$ a set of indices, $e_i \in \{-1, 1\}$, $1 \le j_i, k_i \le n$) contains x start^+ *(resp.* start^-*) if there exist x indices in $\mathcal{I}$ such that $e_i = 1$ (resp. $e_i = -1$) and start(π_{k_i}) belongs to s_{j_i}.*

By extension, we say that $\prod_{i \in \mathcal{I}} C^{e_i}(M_{j_i} \to M_{k_i})$ contains x starts *(or* starting points*) if it contains x_1 start$^+$, x_2 start$^-$ and $x_1 + x_2 = x$.*

Definition 4.6 *Consider a GDH-Protocol executed by n participants and let $s_1, \ldots, s_n$ be the n strands and $\pi_1, \ldots, \pi_n$ be the n histories given in this protocol's definition. We define* $\text{split}(\pi_i, \pi_j)$ *as the last node which is part of both π_j and π_k, that is, $\pi_i(k)$ where $k = \max_l(\pi_i(l) = \pi_j(l))$ (*$\text{split}(\pi_i, \pi_j)$ *is undefined if $\pi_i(k) \neq \pi_j(k)\ \forall k$).*

We say that the product of contributions $\prod_{i \in \mathcal{I}} C^{-1}(M_{j_i} \to M_{k_i}) \cdot C(M_{j_i} \to M_{l_i})$ (with $\mathcal{I}$ a set of indices, $1 \le j_i, k_i, l_i \le n$) contains x splits *(or* splitting points*) if there exist x indices in $\mathcal{I}$ such that split(π_{k_i}, π_{l_i}) belongs to s_{j_i}.*

One last notion will be useful for our next proposition: the notion of precedence of contributions. We say that contribution $C(M_i \to M_j)$ precedes contribution $C(M_i \to M_k)$ if every node of π_k belonging to s_i is preceded by a node of π_j belonging to the same strand.

Definition 4.7 *Consider a GDH-Protocol executed by n participants and let $s_1, \ldots, s_n$ be the n strands and $\pi_1, \ldots, \pi_n$ be the n histories given in this protocol's definition. We say that $C(M_i \to M_j)$* precedes *(written $\preceq$) $C(M_i \to M_k)$ iff*

$$\forall y \text{ such that } \pi_k(y) \text{ belongs to } s_i, \\ \exists x : \pi_j(x) \text{ belongs to } s_i \text{ and } \pi_j(x) \preceq \pi_k(y).$$

Given a node n on s_i, we also write that $C(M_i \to M_j) \preceq n$ if $\exists x :\ \pi_j(x)$ belongs to s_i and $\pi_j(x) \preceq n$, and that $n \preceq C(M_i \to M_j)$ when $\forall x :\ \pi_j(x)$ belongs to s_i, $n \preceq \pi_j(x)$.

The strict precedence relation $\prec$ corresponds to the precedence relation except that we replace "$\preceq$" with "$\prec$" in its definition.

These definitions are used in the following proposition in which we state sufficient conditions for the possibility of building more complex pairs of elements of G than those described in Proposition 4.1.

Proposition 4.8 *Consider a GDH-Protocol with n participants and let $p = \prod_{i \in \mathcal{I}} C^{-1}(M_{j_i} \to M_{k_i}) \cdot C(M_{j_i} \to M_{l_i})$ (with $1 \leq j_i, k_i, l_i \leq n$ and $\mathcal{I}$ being a set of indices) be a product of contributions such that all pairs of contributions are provided on different strands. Then an active attacker can obtain a pair (g_1, g_2) of elements of G such that $g_2 = g_1^p$ if one of the following conditions is verified:*

1. *p contains at most one splitting point and no starting point,*
2. *p contains no splitting point, one start$^+$ and no start$^-$,*
3. *p contains no splitting point, no start$^+$ and one start$^-$,*
4. *p contains no splitting point, one start$^+$ (for the index $i_+ \in \mathcal{I}$), one start$^-$ (for the index $i_- \in \mathcal{I}$), $k_{i_+} = k_{i_-}$ and $l_{i_+} = l_{i_-}$.*

Proof. Our proof of this proposition proceeds by using Algorithm 1 (or slight variants of it) and by verifying that, when any condition stated above is respected, the resulting pair (g_1, g_2) has the expected form.

1. p contains at most one splitting point and no starting point
Let $m \in \mathcal{I}$ be the index such that $split(\pi_{k_m}, \pi_{l_m})$ belongs to s_{j_m}, or m be a random element of $\mathcal{I}$ if p does not contain any splitting point. If we initialize g_1 and g_2 to α (or to any random value) and execute Algorithm 1 for the product $C^{-1}(M_{j_m} \to M_{k_m}) \cdot C(M_{j_m} \to M_{l_m})$, we obtain a first pair (g_1, g_2).

Then, we may successively apply Algorithm 1 for all products of contributions corresponding to indexes in $i \in \mathcal{I} - \{m\}$, using the values obtained for g_1 and g_2 at the end of each execution as input for the next one.

This procedure provides the expected values because Algorithm 1 will transform pairs (g_1, g_2) such that $g_2 = g_1^{p_x}$ into pairs (g_1, g_2) such that $g_2 = g_1^{p_x \cdot C^{-1}(M_{j_i} \to M_{k_i}) \cdot C(M_{j_i} \to M_{l_i})}$ for each value of $i \in \mathcal{I} - \{m\}$ as long as the considered product of contributions does not contain any splitting or starting point.

2. p contains no splitting point, one start$^+$ and no start$^-$
The process is nearly identical to the one above. Let $m \in \mathcal{I}$ be the index such that $start(\pi_{l_m})$ belongs to s_{j_m}. If we initialize g_1 and g_2 to α and execute Algorithm 1 for the product $C^{-1}(M_{j_m} \to M_{k_m}) \cdot C(M_{j_m} \to M_{l_m})$, we obtain a first pair (g_1, g_2).

Then, we may successively execute Algorithm 1 for all products of contributions corresponding to indexes in $i \in \mathcal{I} - \{m\}$, providing the values obtained for g_1 and g_2 at the end of each execution as input for the next one.

The correctness of this procedure relies on the same observations as above.

3. p contains no splitting point, no start$^+$ and one start$^-$
The procedure is the same as the previous one.

4. p contains no splitting point, one start$^+$ (for the index $i_+ \in \mathcal{I}$), one start$^-$ (for the index $i_- \in \mathcal{I}$), $k_{i_+} = k_{i_-}$ and $l_{i_+} = l_{i_-}$.
Suppose first $i_+ = i_-$. In that case, the process described for the first condition applies.

Suppose now $i_+ \neq i_-$, $k = k_{i_+} = k_{i_-}$ and $l = l_{i_+} = l_{i_-}$. Suppose also $C(M_{j_{i_-}} \to M_k) \prec C(M_{j_{i_-}} \to M_l)$. If this condition holds and if we define $\pi_k(1) = \langle s_{j_{i_-}}, \hat{z} \rangle$, we can proceed as follows. First, initialize g_1 and g_2 to α and apply Algorithm 1 for the product $C^{-1}(M_{j_{i_-}} \to M_k) \cdot C(M_{j_{i_-}} \to M_l)$ until $z = \hat{z}$ (we also execute this algorithm for this value of z). At this point, our precedence assumption guarantees us that $g_1 = uns\_term(\pi_k(1))$ and $g_2 = \alpha$. Then, with the current values of g_1 and g_2, execute the same algorithm for the product $C^{-1}(M_{j_{i_+}} \to M_k) \cdot C(M_{j_{i_+}} \to M_l)$, obtaining a pair $(g_1, g_2) = (uns\_term(\pi_k(1))^{C(M_{j_{i_+}} \to M_k)}, \alpha^{C(M_{j_{i_+}} \to M_l)})$. Now complete the first execution of the algorithm for the values of z going from $\hat{z} + 1$ to $length(s_j)$ with the updated values of g_1 and g_2. Finally, execute Algorithm 1 for the indexes $i \in \mathcal{I} - \{i_+, i_-\}$, always updating g_1 and g_2. This provides the desired pair.

Suppose now $C(M_{j_{i_-}} \to M_k) \not\prec C(M_{j_{i_-}} \to M_l)$. This means that there is an index y such that $\pi_l(y)$ belongs to $s_{j_{i_-}}$ and such that $\pi_l(y) \preceq \pi_k(x)$ for every index x such that $\pi_k(x)$ belongs to $s_{j_{i_-}}$. So, in particular, we have that $\pi_l(y) \preceq \pi_k(1)$ since $\pi_k(1)$ belongs to $s_{j_{i_-}}$. But this last observation guarantees us that $\pi_l(1)$, which belongs to $s_{j_{i_+}}$, strictly precedes all nodes of π_k belonging to that strand, and therefore that $C(M_{j_{i_+}} \rightarrow M_k) \prec C(M_{j_{i_+}} \rightarrow M_l)$. This precedence relation is symmetric to the one above, and allows us to collect g_1 and g_2 by using a symmetric treatment. ∎

All these sufficient conditions can be verified by checking on which strands histories split and start and are independent of the other aspects of the routing of the messages. This will be very convenient to check whether a pair of the form given in Theorem 3.10 can be obtained by the attacker, as we will see.

Unfortunately, in some cases, it will not be possible to be sure that one of these conditions is verified. So, we will define one more sufficient condition. Its verification will be slightly more demanding as it will require to check precedence relations on contributions.

Proposition 4.9 *The following condition is sufficient to make the wording of Proposition 4.8 correct:*

5. *p contains no splitting point, one start$^+$ (for the index $i_+ \in \mathcal{I}$), one start$^-$ (for the index $i_- \in \mathcal{I}$, $i_+ \neq i_-$) and $C(M_{j_{i_-}} \rightarrow M_{k_{i_-}}) \prec C(M_{j_{i_-}} \rightarrow M_{l_{i_-}})$ or $C(M_{j_{i_+}} \rightarrow M_{l_{i_+}}) \prec C(M_{j_{i_+}} \rightarrow M_{k_{i_+}})$*

Proof. This condition explicitly states that one of the precedence conditions we proved when examining the fourth condition of Prop. 4.8 is valid. The desired pair (g_1, g_2) can therefore be built by using exactly the same technique. ∎

An example of the treatment described in these proofs is provided in the full attack process example we propose in Appendix C.

4.4 Attacking GDH-Protocols with four and five participants

We will now prove that, when considering GDH-Protocols with four or five participants, the product of contributions given in the proof of Theorem 3.10 respects one of the conditions of Proposition 4.8 and Proposition 4.9 for at least one choice of the users M_i, M_j, M_k and of the sets S_j and S_k. In order to make it easier to extend this result to an unbounded number of protocol participants, we will in fact consider only two possible choices for the set S_j and S_k:

- $\mathsf{S}_j = \{M_k\}$ and $\mathsf{S}_k = \mathsf{M} - \{M_i, M_k\}$ and
- $\mathsf{S}_j = \mathsf{M} - \{M_i, M_j\}$ and $\mathsf{S}_k = \{M_j\}$.

It is easy to check that these two choices of S_j and S_k respect the conditions given in the proof of Theorem 3.10.

This will prove that the attacker is always able to obtain a pair of values (g_1, g_2) such that M_i would compute g_2 as his view of the group key if he uses

g_1 as input value for this computation. We are not sure however whether the attacker can complete his attack by submitting g_1 to M_i:

1. building g_1 could require the use of services that M_i provides after having computed his view of the group key or
2. M_I could need to use the value that M_i will use to compute the group key in order to build the pair (g_1, g_2).

We can check that the first problem cannot occur: when building g_1, the only contribution that uses the strand from which M_i is computing his view of the group key is $C(M_i \to M_i)$ and we know that all nodes which have to be exploited when collecting $C(M_i \to M_i)$ by using Algorithm 1 strictly precede $\pi_i(\bar{1})$, that is, the node on which g_1 has to be sent to M_i.

Let us now consider the second problem. We already know (from our treatment of the first problem) that we will never need to submit a specific value instead of the last element of π_i when computing g_1. It is however possible that this element has to be used when computing g_2. The only contribution that uses the strand from which M_i is computing his view of the group key in order to build g_2 is $C(M_i \to M_j)$. We can also observe that if the last element of π_i has to be affected to some specific value when collecting $C(M_i \to M_j)$, then the last element of π_i is also part of π_j and, therefore, $split(\pi_i, \pi_j)$ belongs to s_i.

For that reason, instead of simply checking if there is a choice of users M_i, M_j,M_k and of sets S_j and S_k such that the product of contributions given in the proof of Theorem 3.10 respects one of the conditions of Proposition 4.8 and 4.9, we will also require this choice of indices and sets to be such that $split(\pi_i, \pi_j)$ does not belong to s_i.

Obtaining such a result will allow us to conclude that it is impossible to build a secure GDH-Protocol with four or five participants. In Section 4.5, we will show how to extend this result to GDH-Protocols with more than five participants.

Theorem 4.10 *For any GDH-Protocol with four or five participants, it is possible for an active attacker to obtain a pair (g_1, g_2) of elements of G such that $g_2 = g_1^p$ where*

$$\begin{aligned} p \;=\;& C^{-1}(M_i \to M_i) \cdot C(M_i \to M_j) \\ & \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_j)] \\ & \cdot \prod_{l \in 1 \ldots n} K_{Il}^{e_l} \end{aligned}$$

for some choice of M_i, M_j, M_k, S_j, S_k and e_l; where M_i, M_j and M_k are *three different members of the group* M *while* S_j *and* S_k *are two disjoint sets of users defined either as*

- $\mathsf{S}_j = \{M_k\}$ *and* $\mathsf{S}_k = \mathsf{M} - \{M_i, M_k\}$ *or*
- $\mathsf{S}_j = \mathsf{M} - \{M_i, M_j\}$ *and* $\mathsf{S}_k = \{M_j\}$.

Furthermore, it is possible to select M_i *and* M_j *in such a way that* $split(\pi_i, \pi_j)$ *does not belong to* s_i.

Proof. If we suppress the factor $\prod_{l \in 1 \ldots n} K_{Il}^{e_l}$ which is known by M_I from the product p, we can check that p has the form considered in Proposition 4.8. We will therefore try to verify that all GDH-Protocols with at least four participants respect at least one of the four sufficient condition of Proposition 4.8 for an adequate choice of M_i, M_j, M_k, S_j and S_k. We will also require this choice to be such that $split(\pi_i, \pi_j)$ does not belong to s_i, which guarantees us that the attacker will also be able to replace $\pi_i(\bar{1})$ with g_1 in the session from which he is excluded. There will be cases for which it will not be possible to verify one condition of Proposition 4.8. For those cases, we will verify the condition stated in Proposition 4.9.

The verification of the conditions of Proposition 4.8 will be carried out by considering all possible ways for four histories (say π_1, π_2, π_3 and π_4) to split and start in four and five-parties GDH-Protocols. In fact, it can be verified that there are only six ways of making these four histories split and start: we represented them on Fig. 7.[1]

On this figure, we represented the nodes on which these histories start (i.e. $\pi_i(1)$) on the left of each subfigure, then, when it occurs, the nodes on which they split and, finally, the nodes on which they finish (i.e. $\pi_i(\bar{1})$). As an example, the representation of the three histories of the Ex-GDH protocol according to this convention is given in Fig. 8.

Then, for the four histories we consider, we select M_i, M_j and M_k among the four corresponding group members (that is, M_1, M_2, M_3 and M_4) and consider the two possible choices for S_j and S_k described in this theorem's statement.

We then perform an exhaustive search, considering all possible strands for the splitting and starting points and verifying for each of them if one of the sufficient conditions stated in Prop. 4.8 could be verified for at least one specific choice of M_i, M_j, M_k, S_j and S_k. This exhaustive search was performed automatically with the assistance of a small program described in Algorithm 2.

[1] This comes down to generating all binary forests with four leaves. A way to proceed consists in generating all partitions of these four leaves into trees (there are five possible such partitions as $4 = 1+1+1+1 = 1+1+2 = 2+2 = 1+3$; they respectively correspond to one tree with 4 leaves, four trees with 1 leaf, two trees with 1 leaf and one tree with 2 leafs, ...) and, for each of these partitions into trees, generating all trees with the desired number of leaves (the number of such trees is given by the Wedderburn-Etherington numbers).

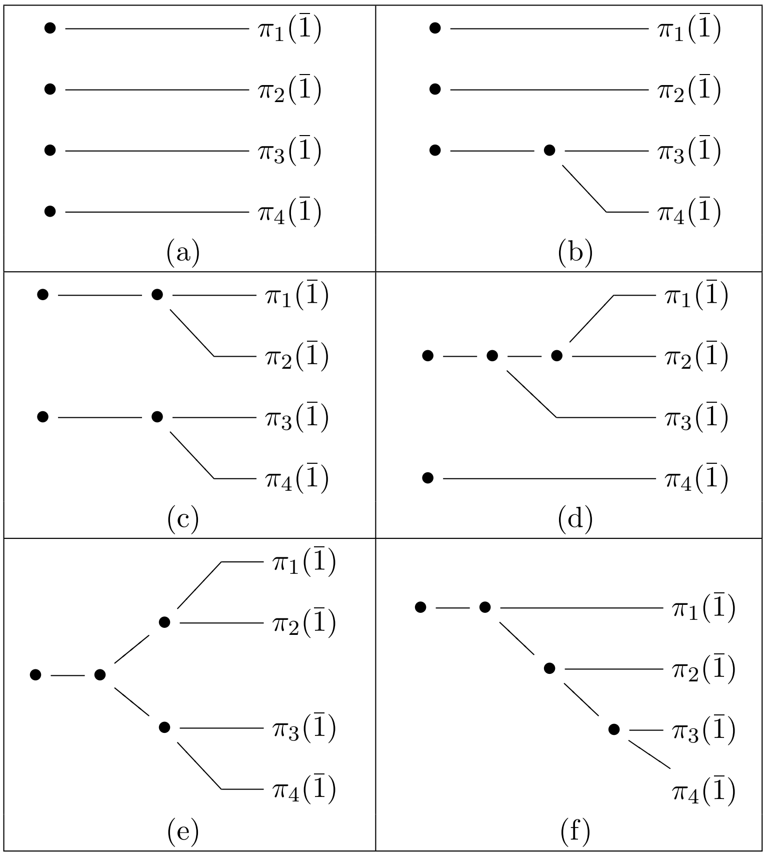


Figure 7: Six varieties of binary forests with four leafs.

This program provided us with an adequate choice in all cases, except nine.

As an example, if we look at the Ex-GDH protocol, such an adequate choice is $M_i = M_3$, $M_j = M_2$, $M_k = M_1$, $\mathsf{S}_j = \{M_1\}$ and $\mathsf{S}_k = \{M_2\}$: in that case, the product

$$\begin{aligned} P(\pi_3(\bar{0})) &= C^{-1}(M_3 \to M_3) \cdot C(M_3 \to M_2) \\ &\cdot [M_1 \backslash M_I : C^{-1}(M_3 \to M_2) \cdot C(M_3 \to M_1)] \\ &\cdot [M_1 \backslash M_I : C^{-1}(M_2 \to M_3) \cdot C(M_2 \to M_1)] \\ &\cdot [M_2 \backslash M_I : C^{-1}(M_1 \to M_3) \cdot C(M_1 \to M_2)] \end{aligned}$$

contains no splitting point and two starting points in the term $[M_2 \backslash M_I : C^{-1}(M_1 \to M_3) \cdot C(M_1 \to M_2)]$. Actually, a more detailed analysis shows that these two starting points do not raise any difficulty as π_2 and π_3 have a splitting point on s_2, which implies that we do not have to use the two corresponding starting services when applying Algorithm 1.

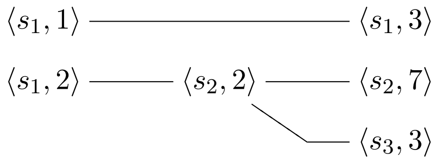


Figure 8: Spitting and starting points in the Ex-GDH Protocol.

As we said, our automatic treatment allowed us to obtain adequate choices for all configurations of histories, except in nine cases, all of them in the forests represented in Fig. 7(a). The strands on which π_1, π_2, π_3 and π_4 start in these nine cases are given in Table 1.

Table 1: Problematic strands for the starting points of π_1, π_2, π_3 and π_4.

	π_1	π_2	π_3	π_4
1)	s_2	s_1	s_4	s_3
2)	s_2	s_3	s_4	s_1
3)	s_2	s_4	s_1	s_3
4)	s_3	s_1	s_4	s_2
5)	s_3	s_4	s_1	s_2
6)	s_3	s_4	s_2	s_1
7)	s_4	s_1	s_2	s_3
8)	s_4	s_3	s_1	s_2
9)	s_4	s_3	s_2	s_1

Unfortunately, in these nine cases, it is not possible to exhibit a choice of M_i, M_j, M_k, S_j and S_k respecting one of the conditions of Proposition 4.8. So, we treated them by hand, each time exhibiting two possible choices of M_i, M_j, M_k, S_j and S_k and verifying that at least one of them verifies the sufficient condition described in Proposition 4.9. As an example, we explain our treatment of the ninth case, where $start(\pi_1)$ belongs to s_4, $start(\pi_2)$ belongs to π_3, $start(\pi_3)$ belongs to s_2 and $start(\pi_4)$ belongs to s_1. The treatment of the other cases is similar, and appropriate choices of M_i, M_j, M_k, S_j and S_k are given in Appendix D.

Since the four histories represented on Fig. 7(a) have no splitting point, $split(\pi_i, \pi_j)$ does obviously not belong to s_i. If we look at the possible choices for M_i, M_j, M_k, S_j and S_k in the case we are examining, we observe that we always have to choose one $start^+$ and one $start^-$. Unfortunately, there is no possible choice verifying the fourth condition of Proposition 4.8.

Algorithm 2 Returns a list of the forests and values of splitting and starting points for which we cannot find a choice of M_i, M_j, M_k, S_j and S_k such that *split*(π_i, π_j) does not belong to s_i and the product p defined in the wording of Thm. 4.10 respects one of the conditions of Proposition 4.8.

---

for all $Forest$ **in** Fig. 7 **do**
  for all strands (s_a, s_b, s_c, s_d) in $\{s_1, s_2, s_3, s_4, s_5\}$ on which the histories of the current forest can split and start **do**
    $SolutionFound$:= False
    for all distinct M_i, M_j, M_k selected in $\{M_1, M_2, M_3, M_4\}$ **do**
      $\mathsf{S}_j := \{M_1, M_2, M_3, M_4, M_5\} \backslash \{M_i, M_j\}$ $\mathsf{S}_k := \{M_j\}$
      if one of the conditions of Prop. 4.8 is verified for this choice of M_i, M_j, M_k, S_j and S_k **and** *split*(π_i, π_j) does not belong to s_i **then**
        $SolutionFound$:= True
      end if
      $\mathsf{S}_j := \{M_k\}$ $\mathsf{S}_k := \{M_1, M_2, M_3, M_4, M_5\} \backslash \{M_i, M_k\}$
      if one of the conditions of Prop. 4.8 is verified for this choice of M_i, M_j, M_k, S_j and S_k **and** *split*(π_i, π_j) does not belong to s_i **then**
        $SolutionFound$:= True
      end if
    end for
    if $SolutionFound$ = False **then**
      Write "The forest $Forest$ with splitting and starting points on s_a, s_b, s_c, s_d is problematic."
    end if
  end for
end for

We now show that it is however possible to verify the condition given in Proposition 4.9.

Suppose we choose $M_i = M_1$, $M_j = M_2$, $M_k = M_4$, $\mathsf{S}_j = \{M_4\}$ and $\mathsf{S}_k = \mathsf{M} - \{M_1, M_4\}$. This choice implies that the product p of this theorem's statement contains one start$^+$ (in the term $C^{-1}(M_1 \to M_2) \cdot C(M_1 \to M_4)$), one start$^-$ (in the term $C^{-1}(M_4 \to M_1) \cdot C(M_4 \to M_2)$), and no splitting point.

Assume first that $C(M_1 \to M_4) \prec C(M_1 \to M_2)$. This relation is one of those described in Prop. 4.9, so it is possible to obtain a pair (g_1, g_2) of the form desired. The same technique can be adopted if $C(M_4 \to M_1) \prec C(M_4 \to M_2)$.

Suppose now that $C(M_1 \to M_4) \not\prec C(M_1 \to M_2)$ and $C(M_4 \to M_1) \not\prec C(M_4 \to M_2)$. From these assumptions, the definition of paths, and the fact that $C(M_4 \to M_1)$ contains a *start*$^+$, we can write:

$$\pi_2(1) \prec C(M_4 \to M_2) \preceq \pi_1(1).$$

We suggest now a second choice of the parameters for p: $M_i = M_2$, $M_j = M_1$, $M_k = M_3$, $\mathsf{S}_j = \{M_3\}$ and $\mathsf{S}_k = \mathsf{M} - \{M_2, M_3\}$. This choice implies that the product p contains one start$^+$ (in the term $C^{-1}(M_2 \rightarrow M_1) \cdot C(M_2 \rightarrow M_3)$), one start$^-$ (in the term $C^{-1}(M_3 \rightarrow M_2) \cdot C(M_3 \rightarrow M_1)$) and no splitting point.

If $C(M_2 \rightarrow M_3) \prec C(M_2 \rightarrow M_1)$ or if $C(M_3 \rightarrow M_2) \prec C(M_3 \rightarrow M_1)$, the condition described in Prop. 4.9 is verified and obtain a pair (g_1, g_2) of the form desired. Suppose now that both these precedence relations are false. Then, the definition of paths and the fact that $C(M_3 \rightarrow M_2)$ contains a *start*$^-$ implies that:

$$\pi_1(1) \prec C(M_3 \rightarrow M_1) \preceq \pi_2(1),$$

which is in contradiction with the relation $\pi_2(1) \prec \pi_1(1)$ obtained above.

Therefore, one of the two choices of M_i, M_j, M_k, S_j and S_k we proposed can be adopted.

A similar reasoning has been carried out for the eight remaining problematic cases. So, we found adequate choices for M_i, M_j, M_k, S_j and S_k for any GDH-Protocol executed by four or five principals. ∎

4.5 Attacking GDH-Protocols with more than five participants

We now try to extend the result we just obtained to an unbounded number of protocol participants. This unbounded number of participants implies that we have to check the five sufficient conditions of Prop. 4.8 and Prop. 4.9 for an unbounded number of contributions and histories. We will see however that the specific choices of S_j and S_k we made will make this difficulty much easier to solve. The following theorem is the same as Theorem 4.10, except that it claims the insecurity of any GDH-Protocols with at least four participants.

Theorem 4.11 *For any GDH-Protocol with $n \geq 4$ participants, it is possible for an active attacker to obtain a pair (g_1, g_2) of elements of G such that $g_2 = g_1^p$ where*

$$\begin{aligned}
p \;=\; & C^{-1}(M_i \rightarrow M_i) \cdot C(M_i \rightarrow M_j) \\
& \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \rightarrow M_j) \cdot C(M_i \rightarrow M_k)] \\
& \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_k)] \\
& \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_j)] \\
& \cdot \prod_{l \in 1 \ldots n} K_{Il}^{e_l}
\end{aligned}$$

for some choice of M_i, M_j, M_k, S_j, S_k *and* e_l*; where* M_i, M_j *and* M_k *are three different members of the group* M *while* S_j *and* S_k *are two disjoint sets of users defined either as*

- $\mathsf{S}_j = \{M_k\}$ *and* $\mathsf{S}_k = \mathsf{M} - \{M_i, M_k\}$ *or*
- $\mathsf{S}_j = \mathsf{M} - \{M_i, M_j\}$ *and* $\mathsf{S}_k = \{M_j\}$.

Furthermore, it is possible to select M_i *and* M_j *in such a way that* $split(\pi_i, \pi_j)$ *does not belong to* s_i.

Proof. We already proved this theorem for four and five participants. We now prove that the four and five-party case implies the validity of the result for any larger number of protocol participants.

Consider a GDH-Protocol GDH_1 with $n > 5$ participants and consider its first four histories π_1, π_2, π_3 and π_4.

Consider now a five-party GDH-Protocol GDH_2, the first four histories of which split and start on the same strands as the GDH_1 protocol, except that all histories splitting or starting on strands in the set $\{s_6, \dots, s_n\}$ in the GDH_1 protocol rather split or start on s_5.

For this protocol, we know from Theorem 4.10 that there is a choice of M_i, M_j and M_k as disjoint users in the set $\{M_1, M_2, M_3, M_4\}$ and a choice of S_j and S_k such that

- $\mathsf{S}_j = \{M_k\}$ and $\mathsf{S}_k = \mathsf{M} - \{M_i, M_k\}$ or
- $\mathsf{S}_j = \mathsf{M} - \{M_i, M_j\}$ and $\mathsf{S}_k = \{M_j\}$

respecting one of the sufficient conditions of Prop. 4.8 and 4.9 and such that $split(\pi_i, \pi_j)$ does not belong to s_i. We call p' the expression of the product p in the GDH_2 protocol, call S the set in $\{\mathsf{S}_j, \mathsf{S}_k\}$ containing more than one protocol participant and M_s the user of the set in $\{\mathsf{S}_j, \mathsf{S}_k\}$ containing one protocol participant.

We now observe that the product

$$\prod_{M_l \in \{M_5, \dots, M_n\}} [\mathsf{S} \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_s)]$$

in the GDH_1 protocol contains at most one *split*, one $start^+$ and one $start^-$. Furthermore, these numbers of splitting and starting points are equal to those in the product

$$[M_5 \backslash M_I : C^{-1}(M_5 \to M_i) \cdot C(M_5 \to M_s)]$$

in the GDH_2 protocol, and the splitting and starting points occur in contributions intended to the same users in the two cases.

So, we know that the products p and p' contain the same number of splitting and starting points, and that these points are part of contributions intended to the same users. This implies that if one of the conditions of Prop. 4.8 is verified in p', it is also verified in p and the selection of M_i,

M_j, M_k, S_j and S_k made for the GDH_2 protocol is also valid for the GDH_1 protocol.

This is only valid for protocols for which we found a choice of M_i, M_j, M_k, S_j and S_k respecting one of the conditions of Prop. 4.8. On the other hand, the protocols which required to use the condition of Prop. 4.9 are such that all splitting and starting points of the first four histories belong to the first four strands and, so, the arguments we used in the proof for four and five participants remain valid. Therefore, at least one of the two possible choices of M_i, M_j, M_k, S_j and S_k we suggest in Appendix D is also valid for the GDH_1 protocol. ∎

This concludes our proof that it is never possible to guarantee the IKA property for all participants of a GDH-Protocol executed by at least four parties.

5 Concluding Remarks

5.1 Summary

In this paper, we analyzed a family of authenticated group key agreement protocols defined as a generalization of the A-GDH protocols proposed in the context of the Cliques project [1, 2].

Our main result is the proof that it is impossible to define a protocol of this family providing implicit key authentication for all group members if it is executed by at least four participants. As we established this proof throughout all the paper, we gather its main points here.

We prove our result by providing a systematic way to set up a scenario that undermines the implicit key authentication property. The process is as follows.

Consider a GDH-Protocol executed by a group M of n users such that $n \geq 4$ and $M_I \notin \mathsf{M}$. If M_I wants to undermine the IKA property in that session, he can select:

- three members of the group M: M_i, M_j and M_k and
- two disjoint sets of users S_j and S_k such that $M_k \in \mathsf{S}_j$, $M_j \in \mathsf{S}_k$, $M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$ and $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$.

This selection must also respect the two following conditions:

- the product

$$
\begin{aligned}
p \;=\; & C^{-1}(M_i \to M_i) \cdot C(M_i \to M_j) \\
& \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)] \\
& \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_k)] \\
& \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_j)] \\
& \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l}
\end{aligned}
$$

  respects at least one of the conditions described in Propositions 4.8 and 4.9.

- $split(\pi_i, \pi_j)$ does not belong to s_i

Theorem 4.11 guarantee that the choice of such M_i, M_j, M_k, S_j and S_k is always possible.

After having selected these values, M_I can build a pair (g_1, g_2) such that $g_2 = g_1^p$ by exploiting a procedure similar to the one described in Algorithm 1 and described in the proof of Prop. 4.8 and 4.9, and replace the value that M_i will use to compute the group key with g_1.

At this time, and given that $p = P(\pi_i(\bar{0}))$ as we proved in Theorem 3.10, M_i will compute g_2 as his view of the group key, which is in contradiction with the implicit key authentication property.

A detailed example of this attack process is given in Appendix C.

5.2 Cardinality of the group

Unexpectedly, and even though the three-party version of the Cliques A-GDH.2 and SA-GDH.2 protocols have been shown to be flawed, our result is found to be only valid for protocols executed by at least four users. This shows that the attacks we discovered are really attacks against group protocols and emphasizes the need to consider these protocols differently than simple extensions of two-party ones.

We think this limit is minimal: we are not able to find any attack against the implicit key authentication property for the two-party version of the A-GDH.2 protocol, nor against our Tri-GDH protocol defined in Section 4.3. Our method fails to find attacks against these two protocols for two different reasons: we are not able to break the two-party version of the A-GDH.2 protocol because we are not able to find services which could be exploited in order to build a pair of the form desired. This is not the case for the Tri-GDH protocol as Theorem 3.10 provides several choices for such services. However, for this last protocol, we are not able to combine these services in a useful way, as we have always need to use the starting point of three histories.

5.3 Conclusion

We think our contribution in this paper has two main aspects.

A first aspect is that we now know that the A-GDH protocols cannot be corrected without changing the design assumptions at their basis. One possible direction to solve this problem would consist in considering the use of a signature scheme or of message authentication codes, what would to separate the key generation part of the protocol (i.e. the sending of the partial Diffie-Hellman values) from the authentication mechanisms. This would allow to make the authentication more explicit, by including the identifiers of the protocol participants and freshness guarantees such as nonces for instance. Such a method has already been exploited in [16] for instance, or by Katz and Yung in [10] for an extension of the Burmester-Desmedt protocol [3] and studied from a more theoretical point of view by Datta & al. in [4] for instance.

A more theoretical aspect concerns the form of our result. While several papers (such as [7, 8, 11, 20]) describe systematic ways to analyze well-defined families of authentication protocols, we do not know any other general impossibility result for such families. It would be interesting to investigate in which measure our result could be transposed to other families of protocols. As our attacks are based on the absence of explicitness of the messages in GDH-Protocols, it would be interesting to see which degree of explicitness would be necessary to obtain secure authenticated group key agreement protocols.

Probably the most closely related results are those concerning the security of ping-pong protocols [7, 8]: as ping-pong protocols, GDH-Protocols are executed by successively applying well-defined transformations on the messages the different users receive (without checking anything about their content). In that sense, we could have used a method similar as their one, but only for obtaining the results of Section 3, i.e. for expressing the secrets of the different users as products of contributions and keys the intruder knows. On the other hand, the routing problems we considered in Section 4 have no correspondence in ping-pong protocols: these protocols consider only one history, and so do not raise the problems we encountered with splitting and starting points.

Our developments rely on several particularities which are only present in Dolev-Yao-type analysis of security protocols (as opposed to computational approaches): we consider a highly restricted set of actions which the intruder can perform, and our analysis method indicates attack scenarios for incorrect protocols rather than leading the analyst to the impossibility of finding a proof. Therefore, we think that our result emphasizes the advantages of using high-level models in the analysis of security protocols.

Acknowledgements

We would like to thank the anonymous reviewers from CSFW-17 and the Journal of Computer Security for very helpful comments on previous versions of this paper. We also would like to thank Sylvie Baudine for her valuable help to finish this paper.

References

[1] G. Ateniese, M. Steiner, and G. Tsudik. Authenticated group key agreement and friends. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 17–26, San Francisco, USA, 1998. ACM Press.

[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. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. De Santis, editor, *Proceedings of Eurocrypt'94*, pages 275–286, Perugia, Italy, 1994. Springer-Verlag - LNCS Vol. 950.

[4] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of the 17-th IEEE Computer Security Foundations Workshop — CSFW'04*, Asilomar, USA, June 2004. IEEE Computer Society Press.

[5] R. Delicata. A security analysis of the CLIQUES protocol suite. Master's thesis, Oxford University Computing Laboratory, 2002.

[6] R. Delicata and S. Schneider. A formal model of Diffie-Hellman using CSP and rank functions. Technical Report CSD-TR-03-05, Department of Computer Science, Royal Holloway, University of London, Jul. 2003.

[7] D. Dolev, S. Even, and R.M. Karp. On the security of ping-pong protocols (extended abstract). In David Chaum, Ronald L. Rivest, , and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of Crypto '82*, pages 177–186, New York, USA, 1982. Plenum Publishing.

[8] S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. Technical Report 285, Technion - Israel Institute of Technology - Computer Science Department, 1983.

[9] S. Hohenberger. The cryptographic impact of groups with infeasible inversion. Master's thesis, MIT, 2003.

[10] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proceedings of Crypto'03*, pages 110–125, Santa Barbara, USA, 2003. Springer-Verlag - LNCS Vol. 2729.

[11] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(2-3):89–146, 1999.

[12] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, July 1999.

[13] J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop — CSFW'03*, Asilomar, USA, 2003. IEEE Computer Society Press.

[14] R. Needham and M. Schroeder. Using encryption in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[15] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987.

[16] O. Pereira. *Modelling and Security Analysis of Authenticated Group Key Agreement Protocols.* PhD thesis, Université catholique de Louvain, 2003.

[17] O. Pereira and J.-J. Quisquater. A security analysis of the Cliques protocols suites. In *Proceedings of the 14-th IEEE Computer Security Foundations Workshop — CSFW'01*, pages 73–81, Cap Breton, Canada, 2001. IEEE Computer Society Press.

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

[19] R. Rivest. On the notion of pseudo-free groups. In M. Naor, editor, *Proceedings of the First Theory of Cryptography Conference - TCC 2004*, pages 505–521. Springer-Verlag - LNCS Vol. 2951, 2004.

[20] S. D. Stoller. A bound on attacks on authentication protocols. In R. Baeza-Yates, U. Montanari, and N. Santoro, editors, *Proc. of the 2nd IFIP International Conference on Theoretical Computer Science: Foundations of Information Technology in the Era of Network and Mobile Computing*, pages 588–600. Kluwer, 2002.

[21] 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.

[22] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 28(3):24–37, Jul. 1994.

A Strand Spaces and Bundles

The following definitions and lemma are taken from [21], Definitions 2.1-2.6 and Lemma 2.7.

Definition A.1 *A signed GDH-Term is a pair $\langle \sigma, t\rangle$ with $t \in \mathsf{G}$ and σ is one of the symbols $+, -$. We will write a signed GDH-Term as $+t$ or $-t$.*

$(\pm\mathsf{G})^*$ *is the set of finite sequences of signed GDH-Terms. We will denote a typical element of* $(\pm\mathsf{G})^*$ *by* $\langle\langle\sigma_1, t_1\rangle, \ldots, \langle\sigma_n, t_n\rangle\rangle$ *or in a shorter way by* $\langle\sigma_1 t_1, \ldots, \sigma_n t_n\rangle$.

Definition A.2 *A* strand space *over* G *is a set* Σ *with a trace mapping* $\mathrm{tr} : \Sigma \to (\pm\mathsf{G})^*$.

By abuse of language, we will still treat signed GDH-Terms as ordinary GDH-Terms. For instance, we shall refer to subterms of signed GDH-Terms. We will also usually refer to GDH-Terms simply as terms.

A strand space will usually be represented by its underlying set of strands Σ.

Definition A.3 *Fix a strand space* Σ.

1. *A node is a pair* $\langle s, i\rangle$*, with* $s \in \Sigma$ *and* i *an integer satisfying* $1 \leq i \leq length(tr(s))$*. The set of nodes is denoted* $\mathcal{N}$*. We will say the node* $\langle s, i\rangle$ *belongs to strand* s*. Clearly, every node belongs to a unique strand.*

2. *If* $n = \langle s, i\rangle \in \mathcal{N}$ *then* $index(n) = i$ *and* $strand(n) = s$*. Define* $term(n)$ *to be* $(tr(s))(i)$*, i.e. the* i*-th signed term in the trace of* s*. Similarly,* $uns\_term(n)$ *is* $((tr(s))(i))_2$*, i.e. the unsigned part of the* i*-th signed term in the trace of* s*.*

3. *There is an edge* $n_1 \to n_2$ *if and only if* $term(n_1) = +t$ *and* $term(n_2) = -t$ *for some* $t \in \mathsf{G}$*. Intuitively, the edge means that* n_1 *sends the message* t*, which is received by* n_2*, recording a potential causal link between those strands.*

4. *When* $n_1 = \langle s, i\rangle$ *and* $n_2 = \langle s, i+1\rangle$ *are members of* $\mathcal{N}$*, there is an edge* $n_1 \Rightarrow n_2$*. Intuitively, the edge expresses that* n_1 *is an immediate causal predecessor of* n_2 *on the strand* s*. We write* $n' \Rightarrow^+ n$ *to mean that* n' *precedes* n *(not necessarily immediately) on the same strand.*

$\mathcal{N}$ together with both sets of edges $n_1 \to n_2$ and $n_1 \Rightarrow n_2$ is a directed graph $\langle\mathcal{N}, (\to \cup \Rightarrow)\rangle$.

A *bundle* is a finite subgraph of $\langle\mathcal{N}, (\to \cup \Rightarrow)\rangle$ for which we can regard the edges as expressing the causal dependencies of the nodes.

Definition A.4 *Suppose* $\to_{\mathcal{C}} \subset \to$*; suppose* $\Rightarrow_{\mathcal{C}} \subset \Rightarrow$*; and suppose that* $\mathcal{C} = \langle\mathcal{N}_{\mathcal{C}}, (\to_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}})\rangle$ *is a subgraph of* $\langle\mathcal{N}, (\to \cup \Rightarrow)\rangle$*.* $\mathcal{C}$ *is a bundle if:*

1. $\mathcal{N}_{\mathcal{C}}$ *and* $\to_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}$ *are finite;*

2. *if* $n_2 \in \mathcal{N}_{\mathcal{C}}$ *and* $term(n_2)$ *is negative, then there is a unique* n_1 *such that* $n_1 \to_{\mathcal{C}} n_2$*;*

3. *if* $n_2 \in \mathcal{N}_{\mathcal{C}}$ *and* $n_1 \Rightarrow n_2$ *then* $n_1 \Rightarrow_{\mathcal{C}} n_2$*;*

4. $\mathcal{C}$ is acyclic.

In conditions (2) and (3), it follows that $n_1 \in \mathcal{N}_\mathcal{C}$, because $\mathcal{C}$ is a graph.

Definition A.5 *A node n is in a bundle $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, (\rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C})\rangle$, written $n \in \mathcal{C}$, if $n \in \mathcal{N}_\mathcal{C}$; a strand s is in $\mathcal{C}$ if all of its nodes are in $\mathcal{N}_\mathcal{C}$.*

If $\mathcal{C}$ is a bundle, then the $\mathcal{C}$-height of a strand s is the largest i such that $\langle s, i\rangle \in \mathcal{C}$.

Example A.6 The scheme of Example 2.7 represents a bundle $\mathcal{C}$ and it remains a bundle if we suppress $\langle s_1, 4\rangle$ from $\mathcal{N}_\mathcal{C}$ as well as the arrows leading to this node from $\rightarrow_\mathcal{C}$ and $\Rightarrow_\mathcal{C}$. However, it is not a bundle anymore if $\langle s_2, 1\rangle$ and the arrows leading to and starting from this node are suppressed from $\mathcal{N}_\mathcal{C}$, $\rightarrow_\mathcal{C}$ and $\Rightarrow_\mathcal{C}$ since $\langle s_2, 2\rangle \in \mathcal{C}$ and $\langle s_2, 1\rangle \Rightarrow \langle s_2, 2\rangle$.

Definition A.7 *If $\mathcal{S}$ is a set of edges, i.e. $\mathcal{S} \subset \rightarrow \cup \Rightarrow$, then $\prec_\mathcal{S}$ is the transitive closure of $\mathcal{S}$ and $\preceq_\mathcal{S}$ is the reflexive, transitive closure of $\mathcal{S}$.*

The relations $\prec_\mathcal{S}$ and $\preceq_\mathcal{S}$ are each subsets of $\mathcal{N}_\mathcal{S} \times \mathcal{N}_\mathcal{S}$, where $\mathcal{N}_\mathcal{S}$ is the set of nodes incident with any edge in $\mathcal{S}$.

Lemma A.8 *Suppose $\mathcal{C}$ is a bundle. Then $\preceq_\mathcal{C}$ is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in $\mathcal{C}$ has $\preceq_\mathcal{C}$-minimal members.*

We regard $\preceq_\mathcal{C}$ as expressing causal precedence, because $n \preceq_\mathcal{C} n'$ holds only when n's occurrence causally contributes to the occurrence of n'. When a bundle $\mathcal{C}$ is understood, we will simply write $\preceq$. Similarly, we will say that a node n *precedes* a node n' if $n \preceq n'$.

B Proof of Theorem 3.10

Several lemmas will be useful to prove Theorem 3.10, which comes down to proving that, for any GDH-Protocol,

$$\begin{aligned} P(\pi_i(\bar{0})) = {} & C^{-1}(M_i \rightarrow M_i) \cdot C(M_i \rightarrow M_j) \\ & \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \rightarrow M_j) \cdot C(M_i \rightarrow M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \rightarrow M_i) \cdot C(M_l \rightarrow M_j)] \\ & \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l} \end{aligned} \tag{2}$$

if S_j and S_k are two disjoint sets of users such that $M_k \in \mathsf{S}_j$, $M_j \in \mathsf{S}_k$, $M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$ and $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$.

Our first lemma says that the key part of the contribution of M_i to M_j in a session executed by the group of users M is a product of keys that M_i shares with the other group members, and only of such keys (that is, it does not contain keys which M_i shares with users outside the group).

Lemma B.1 *For any GDH-protocol, if $i \neq j$,*

$$C_{\mathsf{K}}(M_i \to M_j) = \prod_{k=1\dots n, k\neq i} C_{K_{ik}}(M_i \to M_j).$$

Proof. $C_{\mathsf{K}}(M_i \to M_j)$ can only contain keys of the form $K_{ix} \in \mathsf{K}_i$ since these are the only known on s_i. From Proposition 3.5, we can observe that $C_{K_{ii}}(M_i \to M_j) = 1$. Furthermore, the same proposition guarantees that $C_{K_{ik}}(M_i \to M_j) = 1$ when $k \notin \{1\dots n\}$ (that is, that $C_{\mathsf{K}}(M_i \to M_j)$ does not contain keys shared between M_i and users which are not expected to take part to the protocol execution) since $C_{K_{ik}}(M_k \to M_i)$ is undefined when $M_k \notin \mathsf{M}$. ∎

Our second lemma provides a relation which will be very useful when we will have to prove Lemma B.3. This third lemma will provide an expression of $P_{\mathsf{K}}(\pi_i(\bar{0}))$ as a product of contributions which are a subset of the contributions included in the expression of p given in Equation (2). So, our goal after having proved that lemma will be to prove that the product of the key part of all other contributions included in the product of Equation 2 is known by M_I.

Lemma B.2 *Consider a GDH-Protocol executed by a group of users $\mathsf{M} = \{M_1, \dots, M_n\}$ where $n \geq 3$ and let S_j and S_k be two disjoint sets of users such that $M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$ and $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$. Then*

$$\prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] = \prod_{M_l \in \mathsf{S}_k} C_{K_{li}}(M_l \to M_i) \cdot \prod_{l \in 1\dots n} K_{Il}^{e_l}$$

Proof. Let $M_l \in \mathsf{S}_k$. From Lemma B.1, $[\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] = [\mathsf{S}_j \backslash M_I : \prod_{M_m \in \mathsf{M}-\{M_l\}} C_{K_{lm}}(M_l \to M_i)]$. If $M_m \in \mathsf{S}_j$, then $[\mathsf{S}_j \backslash M_I : C_{K_{lm}}(M_l \to M_i)]$ is a term of the form $K_{lI}^{e_l}$, and is therefore known by M_I. So, $[\mathsf{S}_j \backslash M_I : \prod_{M_m \in \mathsf{M}-\{M_l\}} C_{K_{lm}}(M_l \to M_i)] = [\mathsf{S}_j \backslash M_I : \prod_{m \in \mathsf{S}_k \cup \{M_i\} \backslash \{M_l\}} C_{K_{lm}}(M_l \to M_i)] \cdot K_{lI}^{e_l}$.

Finally, if we simplify the terms in

$$\prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : \prod_{M_m \in \mathsf{S}_k \cup \{M_i\} \backslash \{M_l\}} C_{K_{lm}}(M_l \to M_i)]$$

by using Proposition 3.5, we can observe that the only remaining terms are those of the form $C_{K_{li}}(M_l \to M_i)$, which proves our identity. ∎

Lemma B.3 *Consider a GDH-Protocol executed by a group of users* $\mathsf{M} = \{M_1, \ldots, M_n\}$ *where* $n \geq 3$*. Let* M_i*,* M_j *and* M_k *be three different members of the group* M*. Let* S_j *and* S_k *be two disjoint sets of users such that* $M_k \in \mathsf{S}_j$*,* $M_j \in \mathsf{S}_k$*,* $M_i \notin \mathsf{S}_j$*,* $M_i \notin \mathsf{S}_k$ *and* $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$*. Then*

$$\begin{aligned} P_{\mathsf{K}}(\pi_i(\bar{0}))^{-1} &= C_{\mathsf{K}}(M_i \to M_i) \\ &\cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] \\ &\cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] \cdot \prod_{m \in 1 \ldots n} K_{Im}^{e_m} \end{aligned}$$

Proof. Observation 3.2 tells us that

$$P_{\mathsf{K}}(\pi_i(\bar{0}))^{-1} = \prod_{j=1 \ldots n} C_{\mathsf{K}}(M_j \to M_i)$$

Then we observe that

$$\prod_{M_j \in \mathsf{M} - \{M_i\}} C_{\mathsf{K}}(M_j \to M_i) = \prod_{M_j \in \mathsf{M} - \{M_i\}} C_{K_{ij}}(M_j \to M_i)$$

Actually, from Lemma B.1, $C_{\mathsf{K}}(M_j \to M_i) = \prod_{M_m \in \mathsf{M} - \{M_j\}} C_{K_{jm}}(M_j \to M_i)$ and, from Prop. 3.5, $C_{K_{jm}}(M_j \to M_i) = C^{-1}_{K_{jm}}(M_m \to M_i)$. So, the only terms that are not inverted in $\prod_{M_j \in \mathsf{M} - \{M_i\}} \prod_{M_m \in \mathsf{M} - \{M_j\}} C_{K_{jm}}(M_j \to M_i)$ are those of the form $C_{K_{ji}}(M_j \to M_i)$ (for $M_j \in \mathsf{M} - \{M_i\}$), what proves our identity.

The last step of our proof consists in observing that

$$\prod_{M_j \in \mathsf{M} - \{M_i\}} C_{K_{ij}}(M_j \to M_i) = \prod_{M_j \in \mathsf{S}_j} C_{K_{ij}}(M_j \to M_i) \cdot \prod_{M_j \in \mathsf{S}_k} C_{K_{ij}}(M_j \to M_i)$$

and in using Lemma B.2 on the two terms of the product on the right of this equation. ∎

As we proved that

$$\begin{aligned} P_{\mathsf{K}}(\pi_i(\bar{0}))^{-1} &= C_{\mathsf{K}}(M_i \to M_i) \\ &\cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] \\ &\cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_i)] \cdot \prod_{m \in 1 \ldots n} K_{Im}^{e_m} \end{aligned}$$

we now have to prove that the product

$$\begin{aligned} &C_{\mathsf{K}}(M_i \to M_j) \cdot [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}^{-1}(M_i \to M_j)] \\ &\cdot \prod_{M_l \in (\mathsf{M} - \mathsf{S}_j)} [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_k)] \\ &\cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_j)] \end{aligned}$$

containing the remaining contributions in the expression of $P(\pi_i(\bar{0}))$ we try to prove, is a product of keys that M_I knows.

To this purpose, we prove one last expression, which is very close from the one we proved in Lemma B.2.

Lemma B.4 *Consider a GDH-Protocol executed by a group of users* $\mathsf{M} = \{M_1, \ldots, M_n\}$ *where* $n \geq 3$ *and let* $\mathsf{S} \subset \mathsf{M}$ *be a subgroup of* M *such that* $M_i \in \mathsf{S}$*. Then*

$$\prod_{M_j \in \mathsf{M} \backslash \mathsf{S}} [\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] = \prod_{j \in 1 \ldots n} K_{Ij}^{e_j}$$

Proof. We first note that

$$\prod_{M_j \in \mathsf{M} \backslash \mathsf{S}} [\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] = \prod_{M_j \in \mathsf{M}} [\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] \cdot \prod_{j \in 1 \ldots n} K_{Ij}^{e'_j}$$

as, if $M_j \in \mathsf{S}$, we can be sure that $[\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)]$ is a product of keys known by M_I.

Then, we note that

$$\prod_{M_j \in \mathsf{M}} [\mathsf{S} \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] = [\mathsf{S} \backslash M_I : P_{\mathsf{K}}(\pi_i(\bar{0}))^{-1}]$$

which proves our lemma since $M_i \in \mathsf{S}$. ■

This lemma proves that the term $\prod_{M_l \in (\mathsf{M} - \mathsf{S}_j)} [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}(M_l \to M_k)]$ in the product above is a product of keys the adversary knows.

We now provide a proof of Theorem 3.10.

Theorem B.5 *For any GDH-Protocol executed by a group of users* $\mathsf{M} = \{M_1, \ldots, M_n\}$ *where* $n \geq 3$*, if* S_j *and* S_k *are two disjoint sets of users such that* $M_k \in \mathsf{S}_j$*,* $M_j \in \mathsf{S}_k$*,* $M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$ *and* $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$*. Then,*

$$\begin{aligned} P(\pi_i(\bar{0})) \quad = \quad & C^{-1}(M_i \to M_i) \cdot C(M_i \to M_j) \\ & \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_j)] \\ & \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l} \end{aligned}$$

Proof. We proceed in two steps. In the first one, we prove that the expression above is true for the random part of the product (i.e. $P_{\mathsf{R}}(\pi_i(\bar{0}))$), then we prove that it is correct for the key part of the product (i.e. $P_{\mathsf{K}}(\pi_i(\bar{0}))$).

1. R-part

If we except the last line of the expression of $P(\pi_i(\bar{0}))$ above which is a product of keys, this expression only contains products of pairs of contributions. Proposition 3.3 guarantees us that the random part of the last three lines of products of contributions must be equal to one. Finally, combining the expressions proved in Propositions 3.3 and 3.4 for the product $C_{\mathsf{R}}^{-1}(M_i \to M_i) \cdot C_{\mathsf{R}}(M_i \to M_j)$ guarantees us that it is equal to $P_{\mathsf{R}}(\pi_i(\bar{0}))$.

1. K-part

The use of Lemma B.3 combined with the use of Lemma B.4 says us that our result is proved if the product

$$C_{\mathsf{K}}(M_i \to M_j) \cdot [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}^{-1}(M_i \to M_j)] \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_j)] \tag{3}$$

only contains keys known by M_I.

Using Lemma B.1 with the first two terms of this product provides

$$\begin{aligned} & C_{\mathsf{K}}(M_i \to M_j) \cdot [\mathsf{S}_j \backslash M_I : C_{\mathsf{K}}^{-1}(M_i \to M_j)] \\ &= \prod_{k=1 \ldots n, k \neq i} C_{K_{ik}}(M_i \to M_j) \cdot [\mathsf{S}_j \backslash M_I : \prod_{k=1 \ldots n, k \neq i} C_{K_{ik}}^{-1}(M_i \to M_j)] \\ &= \prod_{M_k \in \mathsf{S}_j} C_{K_{ik}}(M_i \to M_j) \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e'_l} \\ &= [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_i \to M_j)] \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e''_l} \end{aligned}$$

Inserting this last expression in 3 provides:

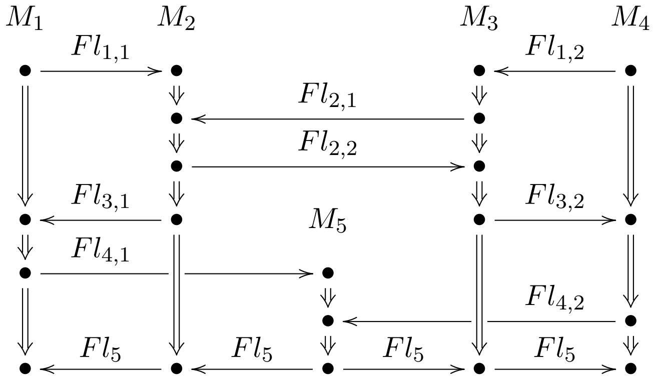
$$\begin{aligned} (3) \quad &= \quad [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_i \to M_j)] \\ & \qquad \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_j)] \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e''_l} \\ &= \quad \prod_{M_l \in \mathsf{M} - \mathsf{S}_k} [\mathsf{S}_k \backslash M_I : C_{\mathsf{K}}(M_l \to M_j)] \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e''_l} \end{aligned}$$

and this last product is a product of keys M_I knows, as we proved in Lemma B.4. ∎

C Illustration of the Attack Process

We illustrate the full attack construction process developed all along this paper. To this purpose, we define a deliberately intricate protocol, the Int-GDH protocol, which will allow us to illustrate our attack construction process more completely than if we considered a simple, regular protocol.

A typical execution of the Int-GDH protocol is represented in the strand space of Fig. 9, where, in order to keep our figure in a reasonable dimension, we represented the transmission of a sequence of elements of $\mathcal{G}$ as a single arrow between two nodes instead of splitting these transmissions into sequences of transmissions of a single element of $\mathcal{G}$.



$$
\begin{aligned}
Fl_{1,1} &= \alpha^{r_1}, \alpha^{r_1 K_{15}} \\
Fl_{1,2} &= \alpha^{r_4} \\
Fl_{2,1} &= \alpha^{r_4}, \alpha^{r_3} \\
Fl_{2,2} &= \alpha^{r_2 K_{12} K_{25}}, \alpha^{r_1}, \alpha^{r_1 r_2 K_{15} K_{25}} \\
Fl_{3,1} &= \alpha^{r_2 r_4}, \alpha^{r_2 r_3} \\
Fl_{3,2} &= \alpha^{r_2 r_3 K_{12} K_{25}}, \alpha^{r_1 r_3}, \alpha^{r_1 r_2 r_3 K_{15} K_{25} K_{35}} \\
Fl_{4,1} &= \alpha^{r_1 r_2 r_4}, \alpha^{r_1 r_2 r_3} \\
Fl_{4,2} &= \alpha^{r_2 r_3 r_4 K_{12} K_{25}}, \alpha^{r_1 r_3 r_4}, \alpha^{r_1 r_2 r_3 r_4 K_{15} K_{25} K_{35} K_{45}} \\
Fl_5 &= \alpha^{r_2 r_3 r_4 r_5 K_{15}}, \alpha^{r_1 r_3 r_4 r_5 K_{25}}, \alpha^{r_1 r_2 r_4 r_5 K_{35}}, \alpha^{r_1 r_2 r_3 r_5 K_{45}}
\end{aligned}
$$

Figure 9: A run of the Int-GDH protocol

Even though the five corresponding histories can be easily deduced from the strand definitions, we provide them in Table 2, where the notation $(\langle s_i, j \rangle, k)$ refers to the k-th element of $\mathcal{G}$ transmitted on $\langle s_i, j \rangle$.

We now have a complete definition of the Int-GDH protocol: strands inform us about the way messages are (normally) exchanged, while histories indicate us how they are computed.

We will now build an attack against this protocol.

We first have to select:

- three group members: M_i, M_j and M_k
- two disjoint sets of users S_j and S_k such that $M_k \in \mathsf{S}_j$, $M_j \in \mathsf{S}_k$,

Table 2: Histories in the Int-GDH Protocol

π_1	π_2	π_3	π_4	π_5
$(\langle s_2, 3\rangle, 1)$	$(\langle s_1, 1\rangle, 1)$	$(\langle s_4, 1\rangle, 1)$	$(\langle s_3, 2\rangle, 2)$	$(\langle s_1, 1\rangle, 2)$
$(\langle s_3, 3\rangle, 1)$	$(\langle s_2, 1\rangle, 2)$	$(\langle s_3, 1\rangle, 1)$	$(\langle s_2, 2\rangle, 2)$	$(\langle s_2, 1\rangle, 2)$
$(\langle s_3, 4\rangle, 1)$	$(\langle s_2, 3\rangle, 2)$	$(\langle s_3, 2\rangle, 1)$	$(\langle s_2, 4\rangle, 2)$	$(\langle s_2, 3\rangle, 3)$
$(\langle s_4, 2\rangle, 1)$	$(\langle s_3, 3\rangle, 2)$	$(\langle s_2, 2\rangle, 1)$	$(\langle s_1, 2\rangle, 2)$	$(\langle s_3, 3\rangle, 3)$
$(\langle s_4, 3\rangle, 1)$	$(\langle s_3, 4\rangle, 2)$	$(\langle s_2, 4\rangle, 1)$	$(\langle s_1, 3\rangle, 2)$	$(\langle s_3, 4\rangle, 3)$
$(\langle s_5, 2\rangle, 1)$	$(\langle s_4, 2\rangle, 2)$	$(\langle s_1, 2\rangle, 1)$	$(\langle s_5, 1\rangle, 2)$	$(\langle s_4, 2\rangle, 3)$
$(\langle s_5, 3\rangle, 1)$	$(\langle s_4, 3\rangle, 2)$	$(\langle s_1, 3\rangle, 1)$	$(\langle s_5, 3\rangle, 4)$	$(\langle s_4, 3\rangle, 3)$
$(\langle s_1, 4\rangle, 1)$	$(\langle s_5, 2\rangle, 2)$	$(\langle s_5, 1\rangle, 1)$	$(\langle s_4, 4\rangle, 4)$	$(\langle s_5, 2\rangle, 3)$
	$(\langle s_5, 3\rangle, 2)$	$(\langle s_5, 3\rangle, 3)$		
	$(\langle s_2, 5\rangle, 2)$	$(\langle s_3, 5\rangle, 3)$		

$M_i \notin \mathsf{S}_j \cup \mathsf{S}_k$, $\mathsf{S}_j \cup \mathsf{S}_k \cup \{M_i\} = \mathsf{M}$.

This selection must also respect the two following conditions:

- $split(\pi_i, \pi_j)$ does not belong to s_i
- the product

$$\begin{aligned} p \;=\; & C^{-1}(M_i \to M_i) \cdot C(M_i \to M_j) \\ & \cdot [\mathsf{S}_j \backslash M_I : C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_k)] \\ & \cdot \prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_j)] \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l} \end{aligned}$$

  respects at least one of the sufficient conditions described in Proposition 4.8.

We first observe that the five histories of the Int-GDH protocol have no common part, so that there are no splitting point.

As a first try, we consider the choice $M_i = M_1$, $M_j = M_2$ and $M_k = M_3$. Whatever choice we do for S_j and S_k, we can verify that the product p will contain at least three starting points: $C(M_1 \to M_2)$, $[\mathsf{S}_j \backslash M_I : C^{-1}(M_1 \to M_2)]$ and $[\mathsf{S}_j \backslash M_I : C^{-1}(M_2 \to M_1)]$. These values of M_i, M_j and M_k are therefore not admissible.

As a second attempt, we consider the choice $M_i = M_1$, $M_j = M_3$, $M_k = M_2$ while $\mathsf{S}_j = \{M_2\}$ and $\mathsf{S}_k = \{M_3, M_4, M_5\}$. This solution implies that p contains one $start^+$: $[\mathsf{S}_j \backslash M_I : C(M_1 \to M_2)]$ and one $start^-$: $[\mathsf{S}_k \backslash M_I : C^{-1}(M_2 \to M_1)]$. As expressed in our fifth condition, this is acceptable only if $C(M_1 \to M_2) \prec C(M_1 \to M_3)$ or $C(M_2 \to M_1) \prec C(M_2 \to M_3)$. Checking these conditions in Table 2 shows that $C(M_2 \to M_1) \not\prec C(M_2 \to M_3)$ because $\langle s_2, 2\rangle \prec \langle s_2, 3\rangle$. However, we can verify that $C(M_1 \to M_2) \prec C(M_1 \to M_3)$ since $\langle s_1, 1\rangle$ strictly precedes all nodes of α_3 belonging to s_1. We are therefore able to build an attack for this selection of values.

A simple way to construct our attack consist in following the procedure explained in Proposition 4.9's proof.

The first step in this procedure consists in defining $\hat{z}$ as the index of the starting point of α_2 in s_1 (given that $C(M_1 \to M_2) \prec C(M_1 \to M_3)$). A simple examination shows that $\hat{z} = 1$.

We now have to execute Algorithm 1 for the product of contributions $[M_2 \backslash M_I : C^{-1}(M_1 \to M_3) \cdot C(M_1 \to M_2)]$ and for values of z ranging from 1 to $\hat{z}$, what means that we will execute this algorithm for only one step of the **for** loop. The values g_1 and g_2 are initialized to α and, for the simplicity of the writings, we always select α^x as random element of G. The random contribution of M_i during the session we are attacking will be written r_i, while his contribution during the session where the intruder replaces the users included in S_j will be denoted r'_i, and we will use the letter r''_i to write the contribution M_i generated during the session where the intruder replaces the users included in S_k. The strand space resulting from this partial execution of Algorithm 1 is represented in Fig. 10. The current values of g_1 and g_2 are indicated as well.

$$M_1 \xrightarrow{\quad \alpha^{r'_1}, \alpha^{r'_1 K_{15}} \quad} M_I$$

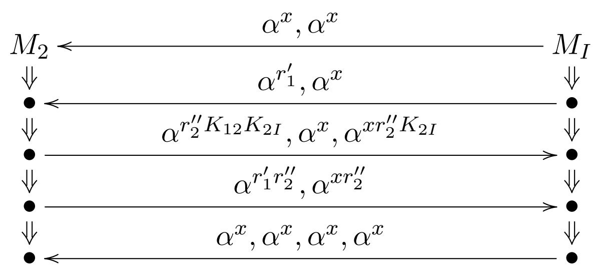
$$g_1 = \alpha, \; g_2 = \alpha^{r'_1}$$

Figure 10: First Step

Always following the procedure indicated in the fifth part of the proof of Proposition 4.8, we now have to execute Algorithm 1 for the product $[\{M_3, M_4, M_5\} \backslash M_I : C^{-1}(M_2 \to M_1) \cdot C(M_2 \to M_3)]$, keeping the current values of g_1 and g_2 as initial values. The resulting strand space is represented in Fig. 11.

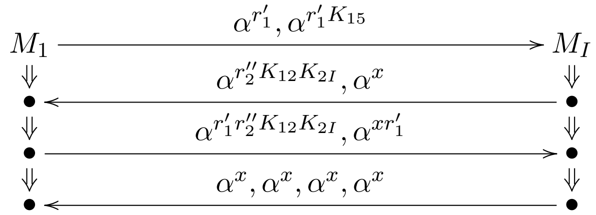
The next step in the procedure described in the fifth part of the proof of Proposition 4.8 consists in completing the execution of Algorithm 1 for the product $[M_2 \backslash M_I : C^{-1}(M_1 \to M_3) \cdot C(M_1 \to M_2)]$. The result of this execution (with the updated values of g_1 and g_2) is represented in Fig. 12.

We now have to execute Algorithm 1 for the remaining products of pairs



$$g_1 = \alpha^{r''_2 K_{12} K_{2I}}, \; g_2 = \alpha^{r'_1 r''_2}$$

Figure 11: Second Step



$$g_1 = \alpha^{r'_1 r''_2 K_{12} K_{2I}}, \; g_2 = \alpha^{r'_1 r''_2}$$

Figure 12: Third Step

of contributions in

$$\begin{aligned} p \;=\; & C^{-1}(M_1 \to M_1) \cdot C(M_1 \to M_3) \\ & \cdot [M_2 \backslash M_I : C^{-1}(M_1 \to M_3) \cdot C(M_1 \to M_2)] \\ & \cdot \prod_{M_l \in \{M_3, M_4, M_5\}} [M_2 \backslash M_I : C^{-1}(M_l \to M_1) \cdot C(M_l \to M_2)] \\ & \cdot [\{M_3, M_4, M_5\} \backslash M_I : C^{-1}(M_2 \to M_1) \cdot C(M_2 \to M_3)] \cdot \prod_{M_l \in \mathsf{M}} K_{Il}^{e_l} \end{aligned}$$

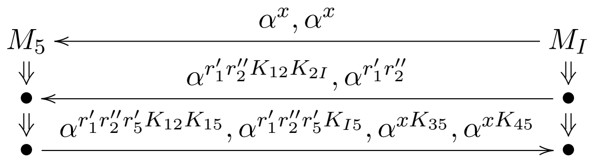
A direct observation however shows that

$$[M_2 \backslash M_I : C^{-1}(M_3 \to M_1) \cdot C(M_3 \to M_2)] = 1$$

$$[M_2 \backslash M_I : C^{-1}(M_4 \to M_1) \cdot C(M_4 \to M_2)] = 1$$

so we do not need to apply Algorithm 1 for these products in our protocol.

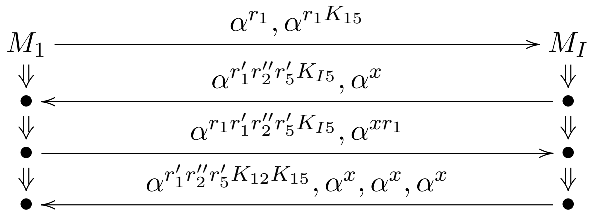
We however have to collect the products $[M_2 \backslash M_I : C^{-1}(M_5 \to M_1) \cdot C(M_5 \to M_2)]$ and $C^{-1}(M_1 \to M_1) \cdot C(M_1 \to M_3)$ by applying the same process as before. Always keeping the current values of g_1 and g_2, the strand space obtained for the product $[M_2 \backslash M_I : C^{-1}(M_5 \to M_1) \cdot C(M_5 \to M_2)]$ is represented in Fig. 13.



$$g_1 = \alpha^{r_1' r_2'' r_5' K_{12} K_{15}}, \; g_2 = \alpha^{r_1' r_2'' r_5' K_{I5}}$$

Figure 13: Fourth Step

In order to complete our attack, we still have to execute Algorithm 1 for the product $C^{-1}(M_1 \to M_1) \cdot C(M_1 \to M_3)$ and to send g_1 as the value M_1 will use to compute the group key. This is represented in Fig. 14.



$$g_1 = \alpha^{r_1' r_2'' r_5' K_{12} K_{15}}, \; g_2 = \alpha^{r_1 r_1' r_2'' r_5' K_{I5}}$$

Figure 14: Last Step

At the end of this process, when M_1 will compute his view of the group key, he will exponentiate $g_1 = \alpha^{r_1' r_2'' r_5' K_{12} K_{15}}$ with $r_1 K_{12}^{-1} K_{15}^{-1}$ and obtain $\alpha^{r_1 r_1' r_2'' r_5'} = g_2^{K_{I5}^{-1}}$. The intruder can therefore compute a key that would normally have to be kept secret.

D Possible choices for M_i, M_j, M_k, S_j and S_k

The following table provides adequate choices for M_i, M_j, M_k, S_j and S_k in the nine problematic cases discussed in the proof of Theorem 4.10. The first column explains on which strands the four histories start, while the second column provides the corresponding choices of M_i, M_j, M_k, S_j and S_k.

Table 3: Possible choices for M_i, M_j, M_k, S_j and S_k

	π_1	π_2	π_3	π_4	M_i	M_j	M_k	S_j	S_k
1)	s_2	s_1	s_4	s_3	M_1	M_3	M_2	$\{M_2\}$	$\mathsf{M}\backslash\{M_1, M_2\}$
					M_3	M_1	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_3, M_4\}$
2)	s_2	s_3	s_4	s_1	M_3	M_1	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_3, M_4\}$
					M_3	M_4	M_1	$\mathsf{M}\backslash\{M_3, M_4\}$	$\{M_4\}$
3)	s_2	s_4	s_1	s_3	M_1	M_2	M_4	$\mathsf{M}\backslash\{M_1, M_2\}$	$\{M_2\}$
					M_1	M_4	M_2	$\{M_2\}$	$\mathsf{M}\backslash\{M_1, M_2\}$
4)	s_3	s_1	s_4	s_2	M_1	M_4	M_3	$\{M_3\}$	$\mathsf{M}\backslash\{M_1, M_3\}$
					M_1	M_3	M_4	$\mathsf{M}\backslash\{M_1, M_3\}$	$\{M_3\}$
5)	s_3	s_4	s_1	s_2	M_1	M_2	M_3	$\{M_3\}$	$\mathsf{M}\backslash\{M_1, M_3\}$
					M_2	M_1	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_2, M_4\}$
6)	s_3	s_4	s_2	s_1	M_1	M_2	M_3	$\{M_3\}$	$\mathsf{M}\backslash\{M_1, M_3\}$
					M_1	M_3	M_2	$\mathsf{M}\backslash\{M_1, M_3\}$	$\{M_3\}$
7)	s_4	s_1	s_2	s_3	M_1	M_3	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_1, M_4\}$
					M_1	M_4	M_3	$\mathsf{M}\backslash\{M_1, M_4\}$	$\{M_4\}$
8)	s_4	s_3	s_1	s_2	M_1	M_2	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_1, M_4\}$
					M_1	M_4	M_2	$\mathsf{M}\backslash\{M_1, M_4\}$	$\{M_4\}$
9)	s_4	s_3	s_2	s_1	M_1	M_2	M_4	$\{M_4\}$	$\mathsf{M}\backslash\{M_1, M_4\}$
					M_2	M_1	M_3	$\{M_3\}$	$\mathsf{M}\backslash\{M_2, M_3\}$