Generic Insecurity of Cliques-Type Authenticated Group Key Agreement Protocols

Olivier Pereira* and Jean-Jacques Quisquater

UCL Crypto Group
Place du Levant, 3
B-1348 Louvain-la-Neuve - Belgium
E-mail: {pereira,quisquater}@dice.ucl.ac.be

Abstract

The A-GDH.2 and SA-GDH.2 authenticated group key agreement protocols showed to be flawed at CSFW 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 describe a proof that it is in fact impossible to design a scalable authenticated group key agreement protocol based on the same building blocks 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). 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 [1, 2] authenticated group key agreement protocols have been shown to be flawed in 2001 [14, 15]. Even though the corresponding attacks (or some variants of them) have been rediscovered in several different frameworks (using the Casper tool [4], rank functions [5] or the constraint solving approach [12] 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 that the method proposed in [15] 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, i.e. 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 security protocols.

In the next section, we will define this family more precisely (Section 2). The next step of our analysis, exposed in Section 3, will consist in the definition of several properties all protocols of our family must verify, 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 will perform in order to obtain the group key can be written as the composition of computations executed by honest users during different protocol sessions, computations of which 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 it is impossible. However, in Section 4, we will prove that it is possible to exploit the result of the previous section 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.

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

2. The GDH Protocols

Authenticated group key agreement protocols are protocols 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.

2.1. Security Properties

The authentication property that is intended for these protocols is the classical implicit key authentication property [11].

Definition 2.1 *A protocol 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 $S_n(M_i)$ (i.e. M_i's view of the session key).*

Besides this main security 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. We do not discuss these properties more in the details 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 Cliques GDH-protocols relies on the assumption that each pair of users (M_i, M_j) share a long-term secret key $K_{ij} \in \mathbb{Z}_q^*$.

During a protocol execution, each group member $M_i \in \mathsf{M}$ selects a random key contribution $r_i \in \mathbb{Z}_q^*$. These assumptions and notations having been introduced, we now define the way the A-GDH.2 protocol is executed.

Protocol 1 : A-GDH.2 Protocol

Round i $(1 \leq i < n)$:
$M_i \rightarrow M_{i+1}: \quad \{\alpha^{\frac{r_1 \ldots r_i}{r_j}} | j \in [1, i]\}, \alpha^{r_1 \ldots r_i}$

Round n:
$M_n \rightarrow$ All M_i: $\quad \{\alpha^{\frac{r_1 \ldots r_n}{r_i} K_{in}} | i \in [1, n[\}$
Upon receipt of the above, every M_i computes the group key as:

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

A typical run of this protocol with 3 participants is represented in Fig. 1.

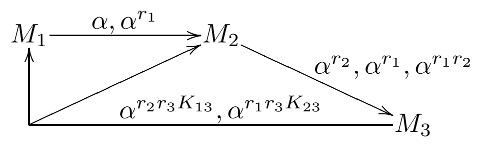


Figure 1. A-GDH.2 Protocol Run with 3 Participants

2.3. An Attack Against the A-GDH.2 Protocol

In order to provide intuitions regarding to the systematic attack construction process we will describe further, we now describe an attack against the A-GDH.2 Protocol.

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 executed by M_1, M_2 and M_3. The goal of the intruder will therefore consist in obtaining a pair of elements of the form $(\alpha^x, \alpha^{x r_2 K_{23}^{-1}})$ and in replacing the second term of M_3's final broadcast with α^x so that M_2 will compute $\alpha^{x r_2 K_{23}^{-1}}$ as group key.

The attacker can obtain such a pair by exploiting what we call *services*. A service is a computation achieved 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 this will allow him to transform a pair of elements he knows into a new pair. All services provided during an A-GDH.2 protocol execution are exponentiation. 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. The services provided by this session participants 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 sending the value $\alpha^{yr'_3K_{23}}$ as second term of the last message M_2 receives in the first protocol session, and M_2 will compute $\alpha^{yr'_3r_2}$ as group key.

We may 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 simply 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 were 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 composed 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?

We now describe the structure of the protocols which we considered as fix candidates for the A-GDH.2 protocol.

A first design assumption we will keep is that we only consider protocols executed by exchanging elements of the public group $\mathcal{G}$, built by exponentiating a public generator α with a product of random values that are generated during the protocol execution and which are only known by the user who generated them; and with a product of long-term shared keys of form K_{ij} where K_{ij} is only known by M_i and M_j. So, for example, elements of $\mathcal{G}$ obtained by multiplying two other elements of the group are not considered.

A second design assumption is that we consider protocols for which the goal is to obtain 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, what 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.

As an example of the protocol family we consider, we suggest a protocol which we will also use to illustrate the definitions, propositions and theorems presented in the next sections.

Example 2.2 We describe here a 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 $\alpha^{r_1r_2r_3}$. Through the rest of this chapter, we will call this protocol the Ex-GDH protocol.

Protocol 2 : Ex-GDH Protocol

Let $r_i, \hat{r}_i \in \mathbb{Z}_q^*$ be random values generated by M_i. The three group members M_1, M_2 and M_3 generate the group key by exchanging the following messages:

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

Upon receipt of the above, M_1 computes the group key $\alpha^{r_1r_2r_3}$ from $\alpha^{\hat{r}_1r_2r_3K_{13}}$, M_2 from $\alpha^{r_1r_3K_{23}^2}$ and M_3 from $\alpha^{r_1r_2}$.

2.5. Modelling the GDH Protocols

We now present our modelling of the protocol family we will consider, and start by defining the set of messages which can be exchanged.

Definition 2.3 *Let:*

1. M *be a set of* n *group members* $\{M_1, \ldots, M_n\}$ *from which the intruder is excluded.*

2. R *be the set of symbols representing random values generated during the protocol execution,* $\mathsf{R}_i \subset \mathsf{R}$ *denoting the set of random values generated by* M_i.

3. K *be the set of symbols representing the long-term shared keys,* $\mathsf{K}_i \subset \mathsf{K}$ *denoting the set of keys known by* M_i *and* $K_{ij} \in (\mathsf{K}_i \cap \mathsf{K}_j)$ *a key shared by* M_i *and* M_j *(for the simplicity of the notations, we will assume that* $K_{ij} = K_{ji}$ *and occasionally write* K_{M_i} *instead of* K_i *or* $K_{M_iM_j}$ *instead of* K_{ij}*).*

4. Atoms *be elements of* $\mathsf{R} \cup \mathsf{K}$.

5. $(\overline{\mathsf{R}}, \cdot)$ *and* $(\overline{\mathsf{K}}, \cdot)$ *be the commutative groups freely generated from* R *and* K *respectively. The unit element of these groups is denoted* 1*. For simplicity, we use multiplicative notations and often write* $a \cdot b$ *as* ab *and* $a \cdot a$ *as* a^2.

6. $(\mathsf{P}, \cdot)$ *be the commutative group isomorphic to* $(\overline{\mathsf{R}} \times \overline{\mathsf{K}})$ *through the morphism* $f(r, K) = r \cdot K$*. It can be noticed from this definition that* P *is free.*

7. p_{R} *and* p_{K} *be the two elements of* P *such that* $p = p_{\mathsf{R}} \cdot p_{\mathsf{K}}$*, with* $p_{\mathsf{R}} \in \bar{\mathsf{R}}$ *and* $p_{\mathsf{K}} \in \bar{\mathsf{K}}$*. Similarly,* p_a *denotes* a^e *where* $p = a^e a_1^{e_1} \cdots a_n^{e_n}$*,* $a \neq (a_i)_{i=1\ldots n}$*, and* $a, (a_i)_{i=1\ldots n}$ *are atoms.*

8. G *be the set that models the finite group* $\mathcal{G}$*. This set is defined through a bijection* $\mathbf{alphaexp} : \mathsf{P} \to \mathsf{G}$ *that represents the exponentiation of the public group generator* α *with some product of random values and keys.*

9. $\alpha = \mathbf{alphaexp}(1)$ *be the symbolic representation of the publicly known generator of* $\mathcal{G}$*.* $\mathbf{alphaexp}(p)$ *will typically be denoted* α^p.

10. $\mathbf{exp} : \mathsf{G} \times \mathsf{P} \to \mathsf{G}$ *be the function which represents the exponentiation of an element of* G *with an element of* P*. If* $g \in \mathsf{G}$ *and* $p \in \mathsf{P}$*,* $\mathbf{exp}(g, p) = \mathbf{alphaexp}(\mathbf{alphaexp}^{-1}(g) \cdot p)$.

We illustrate these definitions through the following example.

Example 2.4 In our Ex-GDH protocol, $\mathsf{M} = \{M_1, M_2, M_3\}$, $\{r_1, \hat{r}_1, r_2, r_3\} \subset \mathsf{R}$ and $\{K_{13}, K_{23}\} \subset \mathsf{K}$; $p = r_1 \cdot r_3 \cdot K_{23}^2$ 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 $\mathbf{exp}(\alpha^{r_1 \cdot r_3 \cdot K_{23}}, K_{23}) = \alpha^{r_1 \cdot r_3 \cdot K_{23}^2}$.

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, in accordance with 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 [8, 16].

We also define a subterm relation $\sqsubset$ as follows:

Definition 2.5 *Let* a *be an atom.*

- $a \sqsubset p \in \mathsf{P}$ *if* $p_a \neq 1$
- $a \sqsubset g \in \mathsf{G}$ *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 definitions.

Example 2.6 Let $g = \alpha^{r_1 K_{23}}$ be an element of G. Then $r_1 \sqsubset g$ and $K_{13} \not\sqsubset g$.

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 model the sending (resp. the reception) of a sequence of n elements of $\mathcal{G}$ as n sending (resp. receptions) of elements of G. So, in our model, all messages (also called *GDH-Terms*) are elements of G.

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 $\langle s, i\rangle$ is the GDH-Term associated with the i-th node of the strand s. A bundle is a directed graph whose edges express the causal dependencies of the nodes (a "$\to$"-edge connects two nodes whose associated GDH-Terms are of form $+t$ and $-t$, while a "$\Rightarrow$"-edge connects two consecutive nodes of a strand). 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).

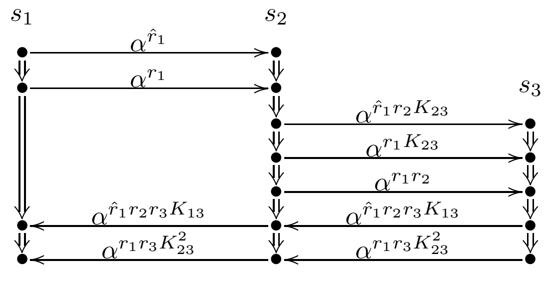


Figure 2. A run of the Ex-GDH protocol

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 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 is linear since the combination of two elements of $\mathcal{G}$ into a third one never occurs, and could therefore be described as a path.

Definition 2.8 *Given a bundle $\mathcal{C}$, a* path π *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* $term(n_m) = -t'$
- $(n_{2j+1}, n_{2j+2}) \in \rightarrow_{\mathcal{C}}$ $(0 \leq j < m/2)$
- $(n_{2j}, n_{2j+1}) \in \Rightarrow^{+}_{\mathcal{C}}$ $(0 < j < m/2)$

We introduce a few more definitions about paths:

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

1. $\pi(j) = n_j$
2. $\langle \pi, j \rangle = uns\_term(n_j) \in \mathsf{G}$*; for the simplicity of the further definitions, the element $\langle \pi, 0 \rangle$ is defined as α*
3. $P(\pi(j)) = p : \langle \pi, j \rangle = \mathbf{exp}(\langle \pi, j-1 \rangle, p)$ $(0 < j \leq m)$
4. $strand(\pi(j)) = strand(n_j)$
5. $Id(\pi(j)) = M_k$ *where M_k is the user executing* $strand(\pi(j))$

From this definition, $\pi(j)$ is the j-th node of π, $\langle \pi, j \rangle$ is the element of G exchanged at the j-th node of π, $P(\pi(j))$ is the value that has to be used for computing $\langle \pi, j \rangle$ from $\langle \pi, j-1 \rangle$, $strand(\pi(j))$ is the strand n_j belongs to, $Id(\pi(j))$ is the identifier of the user executing $strand(\pi(j))$. These notions are exemplified below.

Example 2.10 If we consider the bundle of Example 2.7, a path π describing the history of $\langle s_2, 7 \rangle = \alpha^{r_1 r_3 K_{23}^2}$ is

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

$$\langle \pi, 1 \rangle = \alpha^{r_1}, \langle \pi, 3 \rangle = \alpha^{r_1 K_{23}}, \langle \pi, 5 \rangle = \alpha^{r_1 r_3 K_{23}^2}$$
$$P(\pi(1)) = r_1, P(\pi(2)) = 1, P(\pi(5)) = r_3 K_{23}$$
$$strand(\pi(2)) = s_2, \quad Id(\pi(6)) = M_2$$

As we will use path in order to describe the way messages are transformed along strands, we 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.

Definition 2.11 *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 if there are values for i and j such that p is known on $\pi_i(j)$ and $strand(\pi_i(j)) = s$,*
3. *$p \in \mathsf{P}$ is* locally known *on the strand s if s is the only strand of $\mathcal{C}$ on which p is known,*
4. *$p \in \mathsf{P}$ is* locally known *in $\mathcal{C}$ if p is known on one and only strand of $\mathcal{C}$.*

We can now define the class of protocols we consider.

Definition 2.12 *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. This part of the definition expresses how the GDH-Terms are exchanged.*
2. *a set $\pi = \{\pi_1, \ldots, \pi_n\}$ of n paths in $\mathcal{C}_{GDH}$, these specific paths being called* histories*. These histories express how the exchanged GDH-Terms are computed.*
   Let $n_j^F = \pi_j(length(\pi_j))$ and $\alpha_j^F = \langle \pi_j, length(\pi(j)) \rangle$.
   (a) *M_j computes the group key from α_j^F (so, $strand(n_j^F) = s_j$). Let p_j^F be the element of P such that $\mathbf{exp}(\alpha_j^F, p_j^F) = \alpha^{r_1 \cdots r_n}$*
   (b) *$\langle \pi_j, 2k+1 \rangle$ is computed from $\langle \pi_j, 2k \rangle$ by $Id(\pi_j(2k+1))$*
   (c) *If $a \in \mathsf{R}$ is known on $\pi_j(k)$ then it is locally known*
   (d) *For any $\pi_j \neq \pi_i$, there exists at least one index k such that the contribution r_i is known on $\pi_j(k)$ and $strand(\pi_j(k)) = s_i$*
   (e) *If $a \in \mathsf{K}$ is known on $\pi_j(k)$ then $a \in \mathsf{K}_{Id(\pi_j(k))}$*
   (f) *If $a \in \mathsf{R}$ is known on $\pi_j(k)$ and $a \sqsubset p_l^F$, then $strand(\pi_j(k)) = s_l$*
   (g) *If $a \sqsubset p_j^F$ ($a \in \mathsf{K}$), then $a \in \mathsf{K}_j$*

Example 2.13 Our Ex-GDH protocol is an example of protocol respecting this definition and there is only one way to define π_1, π_2 and π_3 for this protocol:

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

The histories $\pi_1, \ldots, \pi_n$ express how the elements of G that will be used to compute the group key are built (points 2a and 2b). Random contributions generated during an execution of the protocol are assumed to be locally known (point 2c): since they are never communicated in a readable form and are not guessable, they cannot be used to compute elements of G on more than one strand. We also impose that the contribution r_i to the group key is communicated by M_i on the element of G that will be used by the other group members to compute the group key (point 2d). These last two conditions notably impose that r_i must be generated by M_i and be kept secret. In point 2e, we say that the user M_i can only use keys he is supposed to know when he builds new elements of G. Point 2f expresses that the random values used by M_j to compute the group key are not known on any strand executed by an other group member, while point 2g expresses that M_j can only use keys he knows to compute the group key.

We will now introduce a few definitions and notations more before writing properties of GDH-Protocols.

Definition 2.14 *By default, we always refer to a GDH-protocol for a group* $\mathsf{M} = \{M_1, \ldots, M_n\}$ *described through a GDH-Bundle* $\mathcal{C}_{GDH}$ *and through histories* $\pi_1, \ldots, \pi_n$ *. Let:*

1. $C(M_j \rightarrow M_i) = \prod p_k : p_k = P(\pi_i(k))$ *and* $Id(\pi_i(k)) = M_j$ $(0 < k \leq length(\pi_i))$*;* $C(M_j \rightarrow M_i)$ *represents the contribution that* M_j *gives to* α_i^F *through the strand* s_j
2. $F_i = \mathbf{alphaexp}^{-1}(\alpha_i^F)$
3. $R = r_1 \cdots r_n$
4. $R_i = (p_i^F)_{\mathsf{R}} = R \cdot (F_i)_{\mathsf{R}}^{-1}$
5. $K_i = (p_i^F)_{\mathsf{K}} = (F_i)_{\mathsf{K}}^{-1}$

Example 2.15 The first table below indicates the value of $C(M_i \rightarrow M_j)$ for the Ex-GDH protocol in the line M_i of column M_j. The second table indicates the value of F_i, R_i and K_i.

C	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

$F_1 = \hat{r}_1 r_2 r_3 K_{13}$	$R_1 = r_1(\hat{r}_1)^{-1}$	$K_1 = K_{13}^{-1}$
$F_2 = r_1 r_3 K_{23}^2$	$R_2 = r_2$	$K_2 = K_{23}^{-2}$
$F_3 = r_1 r_2$	$R_3 = r_3$	$K_3 = 1$

3. Properties of GDH-Protocols

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. They are considered in the absence of any attacker, and we will show in the next sections how they can be exploited in order to break security properties of such protocols.

It can be observed in the following paragraphs that we never precisely specify to which session of a protocol we refer: we simply state the corresponding group constitution when it is different from M. This is because we will always consider a single protocol execution for each specified group constitution. If, in a different context, a situation imposed us to consider several sessions of a protocol executed by the same group of users, we simply would need to add some supplementary references or indices in order to identify the strands to which we refer for the values local to specific sessions.

We now start our list of properties with two observations that will be used 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* $a \not\sqsubset p_3$*.*

Observation 3.2 *From the definition of* F_j*,*

1. $(F_j)_{\mathsf{R}} = \prod_{i=1\ldots n} C_{\mathsf{R}}(M_i \rightarrow M_j)$
2. $(F_j)_{\mathsf{K}} = \prod_{i=1\ldots n} C_{\mathsf{K}}(M_i \rightarrow M_j)$

This observation can be verified in Example 2.15. We can now write a first proposition about the value of $C_{\mathsf{R}}(M_i \rightarrow M_j)$ when $i \neq j$.

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

Proof. From Observation 3.2 and the definition of R_j, we can write

$$\prod_{i=1\ldots n} C_{\mathsf{R}}(M_i \rightarrow M_j) \cdot R_j = R \tag{1}$$

We can observe that $r_i \sqsubset R$. Furthermore, $r_i \not\sqsubset C_{\mathsf{R}}(M_k \rightarrow M_j)$ $(k \neq i)$ else $\exists l : r_i \sqsubset P(\pi_j(l))$ and $Id(\pi_j(l)) = M_k$ what is impossible given points 2c and 2d of the definition of the GDH-Protocols. Finally, $r_i \not\sqsubset R_j$ given points 2d and 2f of the same definition. We can deduce from these remarks and from Observation 3.1 that $r_i \sqsubset C_{\mathsf{R}}(M_i \rightarrow M_j)$ and that $C_{r_i}(M_i \rightarrow M_j) = (R)_{r_i} = r_i$.

Let us now imagine that $C_{\mathsf{R}}(M_i \rightarrow M_j) = r_i \cdot r$. Then $r_i \not\sqsubset r$. Suppose $r_a \sqsubset r$. From Observation 3.1, $r_a \sqsubset C_{\mathsf{R}}(M_i \rightarrow M_j)$. Since r_a is known on s_i, it is locally known on s_i and is therefore not known on s_k $(k \neq i)$. So, $r_a \not\sqsubset C_{\mathsf{R}}(M_k \rightarrow M_j)$ $(k \neq i)$, $r_a \not\sqsubset R_j$ (from point 2f of Def. 2.12) and $r_a \not\sqsubset R$, what contradicts Observation 3.1 and Equation (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,* $C_{\mathsf{R}}(M_i \to M_i) = r_i \cdot R_i^{-1}$.

Proof. By definition, $R_i = R \cdot (F_i)_{\mathsf{R}}^{-1}$ and $R = \prod_{j=1\ldots n} r_j$. So, by successively exploiting Observation 3.2 and Proposition 3.3, we can write:

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

∎

These two propositions can be checked for the Ex-GDH protocol in the tables of Example 2.15.

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

Proposition 3.5 *For any GDH-Protocol, if* $C_{K_{jk}}(M_j \to M_i) = K_{jk}^a$ $(i \neq j, k)$ *then* $C_{K_{jk}}(M_k \to M_i) = K_{jk}^{-a}$.

Proof. From Observation 3.2, we know that $\prod_{l=1\ldots n} C_{\mathsf{K}}(M_l \to M_i) \cdot K_i = 1$; so the sum of the powers of K_{jk} in the components of the left part of this equation must be null. But $K_{jk} \not\sqsubseteq K_i$ since $K_{jk} \notin \mathsf{K}_i$. Just as $K_{jk} \not\sqsubseteq C_{\mathsf{K}}(M_l \to M_i)$ $(l \neq j, k)$ 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.

∎

Rather than considering the relations between values inside one session of a protocol, we would now like to write a proposition concerning the use of long-term keys in different sessions. To this effect, we introduce a substitution operator: if $p \in \mathsf{P}$ is such that $p_{\mathsf{R}} = 1$ and is a function of elements of a bundle corresponding to a session of a GDH-Protocol, $[M_i \backslash M_I : p]$ (where $M_i \in \mathsf{M}$ and $M_I \notin \mathsf{M}$) refers to the value that p would have in a session where the participants are the same 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* $K_{ij} \not\sqsubseteq K_x$ $(\forall j)$ *then* $[M_i \backslash M_I : p] = \prod_j K_{Ij}^{e_{ij}} \cdot K_x$. *More generally, if* $S = \{M_{i_1}, \ldots M_{i_s}\}$, $[S \backslash M_I : p] = [M_{i_1} \backslash M_I : [(S \backslash \{M_{i_1}\}) \backslash M_I : p]]$.

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, i.e. $C_{\mathsf{K}}(M_j \to M_i)$, with his contribution $[\mathsf{M}_s \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)]$ in a session where a set of honest members $\mathsf{M}_s \subset \mathsf{M}$ 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 M_s will be replaced by keys shared between M_j and M_I.

Proposition 3.8 *Let* $\mathsf{M}_s \subset \mathsf{M}$, $M_j \notin \mathsf{M}_s$. *Then* $C_{\mathsf{K}}(M_j \to M_i) = [\mathsf{M}_s \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] \cdot \prod_{M_k \in \mathsf{M}_s} C_{K_{jk}}(M_j \to M_i) \cdot \prod_{M_k \in \mathsf{M}_s} [\mathsf{M}_s \backslash M_I : C_{K_{jk}}^{-1}(M_j \to M_i)]$.

Proof. $C_{\mathsf{K}}(M_j \to M_i)$ 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 $C_{\mathsf{K}}(M_j \to M_i)$ is therefore $\prod_{M_k \in \mathsf{M}_s} K_{jk}^{e_k} \cdot K_x$ where $K_{jk} \not\sqsubseteq K_x$ for all $M_k \in \mathsf{M}_s$ and K_x is a product of keys in K_j. Definition 3.6 now implies that $[\mathsf{M}_s \backslash M_I : C_{\mathsf{K}}(M_j \to M_i)] = \prod_{M_k \in \mathsf{M}_s} K_{jI}^{e_k} \cdot K_x$.

This proposition results from the fact that $K_{jk}^{e_k}$ can be written as $C_{K_{jk}}(M_j \to M_i)$ and that $K_{jI}^{e_k}$ can be written as $[M_k \backslash M_I : C_{K_{jk}}(M_j \to M_i)]$. ∎

Example 3.9 Consider the Ex-GDH protocol and $\mathsf{M}_s = \{M_1\}$. In this case, $C_{\mathsf{K}}(M_3 \to M_1) = K_{13} K_{23}^{-1}$, $[M_1 \backslash M_I : C_{\mathsf{K}}(M_3 \to M_1)] = K_{I3} K_{23}^{-1}$, $C_{K_{13}}(M_3 \to M_1) = K_{13}$ and $[M_1 \backslash M_I : C_{K_{13}}(M_3 \to M_1)] = K_{I3}$.

All these propositions can be used to prove our main property concerning contributions: the product $R_i \cdot K_i$ 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* $R_i \cdot K_i$ *as a product of contributions* $C(M_j \to M_k)$ $(M_j, M_k \in \mathsf{M} \cup \{M_I\})$ *and of keys known by* M_I.

Proof. (*See [13] for details*)

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

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

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

4. Collecting Contributions

4.1. Introduction

At this point, 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 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.

We will now see how the contributions defined in the proof of Theorem 3.10 can be collected by the intruder, his goal being the obtention of a pair (g_1, g_2) of elements of G such that $g_2 = g_1^{R_i K_i}$.

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 the considered 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.

This algorithm may be justified as follows. Let s_i be a strand that corresponds to M_i's role in an execution of the considered protocol by the group M. 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 $\alpha^{C(M_i \to M_j)}$ into the variable g_1 and $\alpha^{C(M_i \to M_k)}$ into the variable 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.

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**
      $g_1 := \langle \pi_j, x\rangle$
    end if
    if $\exists y : \langle s_i, z\rangle = \pi_k(y)$ and $\pi_j(y) \neq \pi_k(y)$ **then**
      $g_2 := \langle \pi_k, y\rangle$
    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

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 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 (which we will note as n) and the histories we consider:

- if
  - $term(n)$ is negative,
  - $term(n)$ 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 output of this service is not part of both π_j and π_k

  then the intruder provides g_1 (resp. g_2) as input of this service
- if
  - $term(n)$ is positive,
  - $term(n)$ 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 collects the output of this service in g_1 (resp. g_2).

This process always succeeds because when two histories have an element in common, then all preceding elements of these histories are also common (item 2b of Definition 2.12). ■

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 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 α,

$z = 3$ $term(\langle s_2, 3\rangle)$ 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. 3.

4.3. Composing Contributions

As shown in the previous section, 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 $C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)$. We now 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 build more complex pairs; our goal being to obtain a pair of the form described in Theorem 3.10.

This is however not always possible, as we will show through the following example.

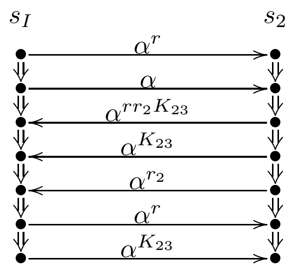


Figure 3. Representation of s_I and s_2

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

Protocol 3 : Tri-GDH Protocol

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

A run of this protocol is represented in Fig. 4. During the protocol first round, the three central messages are exchanged, while the three external ones are computed from those just received and sent during the second round.

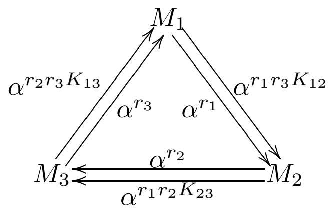


Figure 4. A run of the Tri-GDH protocol

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

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

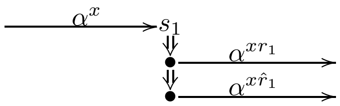
where r_i, r_i', r_i'' represent random values generated during three sessions of the protocol; the participants of these sessions being respectively $\{M_1, M_2, M_3\}$, $\{M_1, M_2, M_I\}$ and $\{M_1, M_I, M_3\}$.

Among these contributions we may consider r'_1, r'^{-1}_2 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 $p = r'_1 r'^{-1}_2 r''_3$, 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 impossible.

Guided by this example, we can observe more generally that we are not usually able to compose two contributions containing initial parts of the corresponding histories if we have to exploit these contributions in the same direction (i.e. if their powers have the same sign).

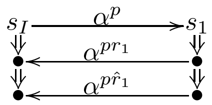
Another kind of services can be problematic: if two services have the same input and two distinct outputs, we may observe that $\pi_j(x) = \pi_k(y)$ for these services input and that the corresponding element of the GDH-Term t will be affected twice in Algorithm 1. This was not a problem when the precondition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

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

We now more precisely define the two problems we just described trough the notions of *starting* and *splitting* points.

Definition 4.5 *Consider a GDH-Protocol with n participants and let $\pi_1, \ldots, \pi_n$ be the n histories given in the definition of this protocol. We define* $\mathrm{start}(M_i)$ *as* $Id(\pi_i(1))$.

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

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

Definition 4.6 *Consider a GDH-Protocol with n participants and $\pi_1, \ldots \pi_n$ the n histories given in the definition of this protocol. We define* $\mathrm{split}(M_i, M_j)$ *as $Id(\pi_i(k))$ where $k = \max_l(\pi_i(l) = \pi_j(l))$ (*$\mathrm{split}(M_i, M_j)$ *is undefined if $\pi_i(l) \neq \pi_j(l)\ \forall l$).*

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

One last definition will be useful for our next proposition.

Definition 4.7 *Consider a GDH-Protocol with n participants and let $\pi_1, \ldots, \pi_n$ be the n histories given in the definition of this protocol. We say that $C(M_i \rightarrow M_j)$* precedes *(written $\preceq$) $C(M_i \rightarrow M_k)$ iff $\forall y :\ Id(\pi_k(y)) = M_i$, $\exists x\ : Id(\pi_j(x)) = M_i$ and $\pi_j(x) \preceq \pi_k(y)$.*

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

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

We may observe that point 2d of Def. 2.12 of GDH-Protocols implies that the precedence relation is always defined in $C(M_i \rightarrow M_j) \preceq C(M_i \rightarrow M_k)$ when $i \neq j$ and $i \neq k$.

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

Proposition 4.8 *Consider a GDH-Protocol with n participants and let $p = \prod_{i\in\mathcal{I}} C^{-1}(M_{j_i} \rightarrow M_{k_i}) \cdot C(M_{j_i} \rightarrow M_{l_i})$ (with $1 \leq j_i, k_i, l_i \leq n$) be a product of contributions such that all pairs of contributions are provided in 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$^+$ and one start$^-$; both occurring for the index $i \in \mathcal{I}$*
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.(*See [13] for details*)
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. ∎

These sufficient conditions can be used to prove that a pair of the form given in Theorem 3.10 can be obtained by the attacker. We will now prove that any GDH-Protocol with at least four participants respects one of these conditions for at least one choice of the indices i, j, k and of the sets S_j and S_k in the equation given in the proof of Theorem 3.10.

Theorem 4.9 *For any GDH-Protocol with at least four 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*

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

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

Proof. (See [13] for details) If we suppress from the product p the factor $\prod_{l \in 1 \ldots n} K_{Il}^{e_l}$ which is known by M_I, we can check that p has the form considered in Proposition 4.8. We will therefore verify that all GDH-Protocols with at least four participants respect at least one of the five sufficient condition of Proposition 4.8 for an adequate choice of M_i, M_j, M_k, S_j and S_k.

The problem we are now confronted to consists in the infinite number of protocols for which we have to check our five conditions. To solve this problem, we will only consider four histories of each protocol (say π_1, π_2, π_3 and π_4), and select M_i, M_j and M_k among the four corresponding group members. We will also consider only two possible choices for S_j and S_k: $\mathsf{S}_j = \mathsf{M} \backslash \{M_i, M_j\}$ and $\mathsf{S}_k = \{M_j\}$ or $\mathsf{S}_j = \{M_k\}$ and $\mathsf{S}_k = \mathsf{M} \backslash \{M_i, M_k\}$.

The five conditions we have to check mainly deal with splitting and starting point of histories. We consider five different values for these specific points: M_1, M_2, M_3, M_4 and M_x which represents users in $\mathsf{M} \backslash \{M_1, M_2, M_3, M_4\}$. The consideration of a single value M_x for all values different of M_1, M_2, M_3 and M_4 is not a restriction since, for the two considered choices of S_j and S_k, the product of contributions of users represented by M_x is always the same ($C^{-1}(M_x \rightarrow M_i) \cdot C(M_x \rightarrow M_j)$ or $C^{-1}(M_x \rightarrow M_i) \cdot C(M_x \rightarrow M_k)$ according to the way S_j and S_k are defined).

Having so limited the number of values to check, we performed an exhaustive search, considering all possible values for the different splitting and starting points. This provided us adequate choices in all cases, except nine.

One of these cases corresponded to protocols such that: $start(M_1) = M_4$, $start(M_2) = M_3$, $start(M_3) = M_2$, $start(M_4) = M_1$. Since the four histories π_1, π_2, π_3 and π_4 have four different starting points, they have no splitting point. If we look at the possible choices for M_i, M_j, M_k, S_j and S_k, we may observe that we always have to choose one $start^+$ and one $start^-$. However, we cannot be sure that the precedence relations of Proposition 4.8's fifth condition are always respected for a specific choice of M_i, M_j, M_k, S_j and S_k. This is why our automated search failed. We now show that this problem can be easily resolved through a little more sophisticated analysis.

Suppose we choose $M_i = M_1$, $M_j = M_2$, $M_k = M_4$, $\mathsf{S}_j = \{M_4\}$ and $\mathsf{S}_k = \mathsf{M} \backslash \{M_1, M_2\}$. This choice implies that the product p contains one start$^+$ (i.e. $C(M_1 \rightarrow M_4)$), one start$^-$ (i.e. $C(M_4 \rightarrow M_1)$), and no splitting point. If this choice satisfies the fifth condition of Proposition 4.8, the attacker is able to obtain the desired pair. If this condition is not verified, we know that $C(M_1 \rightarrow M_2) \preceq C(M_1 \rightarrow M_4)$ and that $C(M_4 \rightarrow M_2) \preceq C(M_4 \rightarrow M_1)$. Furthermore, from the definition of possible histories and from the fact that $C(M_4 \rightarrow M_1)$ is a starting point, we can write:

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

Suppose now we choose $M_i = M_2$, $M_j = M_1$, $M_k = M_3$, $\mathsf{S}_j = \{M_3\}$ and $\mathsf{S}_k = \mathsf{M} \backslash \{M_1, M_2\}$. This choice implies that the product p contains one start$^+$ (i.e. $C(M_2 \rightarrow M_3)$), one start$^-$ (i.e. $C(M_3 \rightarrow M_2)$), and no splitting point. If this choice does not satisfy the fifth condition of Proposition 4.8, $C(M_2 \rightarrow M_1) \preceq C(M_2 \rightarrow M_3)$ and $C(M_3 \rightarrow M_1) \preceq C(M_3 \rightarrow M_2)$. Furthermore, from the definition of possible histories and from the fact that $C(M_3 \rightarrow M_2)$ is a starting point, we can write:

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

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 must verify the fifth condition of Proposition 4.8.

A similar reasoning can be 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 at least four principals.

■

4.4. Fooling $\mathrm{M_i}$ into Computing the Desired Key

In the previous sections, we proved that the attacker is always able to obtain a pair of values (g_1, g_2) such that a selected user 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 that the attacker can always submit g_1:

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

We may check that the first problem cannot occur: the only contribution that uses the strand from which M_i is computing his view of the group key in order to build g_1 is $C(M_i \to M_i)$. However, we can be sure that all nodes which have to be exploited when collecting $C(M_i \to M_i)$ strictly precede the node on which g_1 has to be sent to M_i since it has to be submitted as last element of the history π_i.

Let us now consider the second problem. From the arguments above, we know that it is impossible that we need to submit a specific value instead of the last element of π_i when computing g_1. It is however possible that we would have to use this element 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 may also observe that if the last element of π_i has to be affected when collecting $C(M_i \to M_j)$, then the last element of π_i is also part of π_j and, therefore, $split(M_i, M_j) = M_i$. For that reason, we will solve this last problem by proving that the Theorem 4.9 remains correct if we add a supplementary condition on the choice of M_i, M_j and M_k: we require that $split(M_i, M_j) \neq M_i$. Hopefully, our automated analysis described in the proof of Theorem 4.9 anew provided us adequate choices for M_i, M_j, M_k, S_j and S_k in all concerned cases.

5. Concluding Remarks

5.1. Summary

In this paper, we analyzed a family of authenticated group key agreement protocols, family that we defined as a generalization of the GDH protocols proposed in the context of the Cliques project.

Our main result is the proof that it is impossible to write a protocol of this family providing implicit key authentication as soon as it is executed by at least four participants. This proof being established all along 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 $M_I \notin \mathsf{M}$. The attacker M_I selects:

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

This selection must also respect the two following conditions:

- the product $p =$
 $C^{-1}(M_i \to M_i) \cdot C(M_i \to M_j) \cdot$
 $[\mathsf{S}_j \backslash M_I : C^{-1}(M_i \to M_j) \cdot C(M_i \to M_k)] \cdot$
 $\prod_{M_l \in \mathsf{S}_k} [\mathsf{S}_j \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_k)] \cdot$
 $\prod_{M_l \in \mathsf{S}_j} [\mathsf{S}_k \backslash M_I : C^{-1}(M_l \to M_i) \cdot C(M_l \to M_j)] \cdot$
 $\prod_{M_l \in \mathsf{M}} K_{Il}^{e_l}$
 respects at least one of the conditions described in Proposition 4.8.
- $split(M_i, M_j) \neq M_i$

Theorem 4.9 as well as the discussion of Section 4.4 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, the intruder may 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 replace the value M_i will use to compute the group key with g_1.

At this time, and given that $p = R_i \cdot K_i$ 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.

5.2. Cardinality of the group

Unexpectedly, 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 2-party version of the A-GDH.2 protocol, nor against our Tri-GDH protocol defined in Section 4.3. Our method fails in finding attacks against these two protocols for two different reasons: we are not able to break the 2-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 different choices for such services. However, for this last protocol, we are not able to combine these services in a useful way, as we have to use three starting points.

5.3. Conclusion

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

A practical aspect is that we now know that the A-GDH protocols cannot be corrected without changing the design assumptions at their root. One possible direction to solve this problem would consist in considering the use of a signature scheme or of message authentication codes, what would allow separating the key generation part of the protocol (i.e. the sending of the partial Diffie-Hellman values) from the authentication mechanisms. Such a method has already been exploited in [13] for instance, or in [9] for an extension of the Burmester-Desmedt protocol [3].

A more theoretical aspect concerns the form of our result. If several papers (such as [6, 7, 10, 17]) describe systematic ways to analyze well-defined families of 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 practical families of protocols.

Probably the most closely related results are those concerning the security of ping-pong protocols [6, 7]: 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 (in opposition with computational approaches); notably the highly restricted set of actions that we consider that the intruder can perform, and the fact that 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 interest of using high-level models in the analysis of security protocols.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] 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.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

A. Strand Spaces and Bundles

The following definitions and proposition are taken from [18], 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* $\mathcal{C} = \langle \mathcal{N}_{\mathcal{C}}, (\to_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}) \rangle$ *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* $\mathcal{C} = \langle \mathcal{N}_{\mathcal{C}}, (\to_{\mathcal{C}} \cup \Rightarrow_{\mathcal{C}}) \rangle$, *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 you suppress $\langle s_1, 2 \rangle$ from $\mathcal{N}_{\mathcal{C}}$ as well as the arrows leading to this node from $\to_{\mathcal{C}}$ 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 $\mathcal{N}_{\mathcal{C}}$, $\to_{\mathcal{C}}$ 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.* $\mathcal{S} \subset \to \cup \Rightarrow$, *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 $n \preceq n'$.