Authenticated Group Key Agreement and Friends*

Giuseppe Ateniese† Michael Steiner Gene Tsudik

USC Information Sciences Institute
Marina Del Rey, CA
ateniese@isi.edu

IBM Research Laboratory
Rüschlikon, Switzerland
sti@zurich.ibm.com

USC Information Sciences Institute
Marina Del Rey, CA
gts@isi.edu

Abstract

Many modern computing environments involve dynamic peer groups. Distributed simulation, multi-user games, conferencing and replicated servers are just a few examples. Given the openness of today's networks, communication among group members must be secure and, at the same time, efficient. This paper studies the problem of authenticated key agreement in dynamic peer groups with the emphasis on efficient and provably secure key authentication, key confirmation and integrity. It begins by considering 2-party authenticated key agreement and extends the results to Group Diffie-Hellman key agreement. In the process, some new security properties (unique to groups) are discussed.

1 Introduction

This paper is concerned with security services in the context of dynamic peer groups (DPGs). Such groups are common in many network protocol layers and in many areas of modern computing and the solution to their security needs, in particular key management, are still open research challenges [19]. Examples include replicated servers (such as database, web, time), audio and video conferencing and, more generally, collaborative applications of all kinds. DPGs tend to be relatively small in size, on the order of a hundred members. Larger groups are harder to control on a peer basis and are typically organized in a hierarchy of some sort.

Recently, several key agreement protocols geared for DPGs were proposed in [20]. They were obtained by extending Diffie-Hellman key agreement [11] to n parties. These protocols perform initial key agreement (IKA) within a group. Once a group is formed and the initial key is agreed upon, group members may leave (or be excluded) and new members may join. Moreover, entire groups may join and entire sub-groups may need to be excluded. Any membership change must cause a corresponding group key change in order to preserve *key independence*.[1] Since re-running full IKA for each membership change is expensive, other supporting protocols are necessary. The operations supported by these protocols are collectively called auxiliary key agreement (AKA). AKA protocols, also based on Diffie-Hellman extensions, have been developed in [21]. Both IKA and AKA protocols have been shown secure against passive adversaries.[2]

This paper leverages the results of [20, 21] to develop practical and secure *authenticated* key agreement protocols for DPGs. We also consider other relevant security features such as key confirmation, key integrity and entity authentication. In doing so, we discover that the meaning of these and other familiar notions need to be redefined in a group setting.

Our long-term goal is the development of a comprehensive protocol suite and a toolkit for secure communication in DPGs. Although the focus is on relatively small non-hierarchical groups, no specific communication paradigm (e.g., RPC, connection-oriented) is favored, and no assumptions are made about either the topology or technology of the underlying network.

The remainder of the paper is organized as follows. We first discuss the general requirements and issues in authenticated key agreement. After presenting some necessary terminology in Section 3 and 4 we proceed (in Section 5) to develop a 2-party authenticated key agreement protocol based on the Diffie-Hellman method. We then extend the protocol to n parties (i.e., a DPG) and demonstrate security of the result in Section 6.1. Next, we consider *complete group key authentication* (bilateral among all group members) in Section 6.2 and discuss key integrity and key confirmation features. The paper concludes with the discussion of other group security services that are contingent upon authenticated key agreement.

Disclaimer: most proofs in this paper are fairly *informal* in nature. Work is under way to construct more rigorous formal proofs within the confines of the random oracle model [4] and the 2-party authentication model of Bellare et al.[3].

2 Key Establishment Protocols

Key establishment protocols can be roughly classified in two categories: *key agreement* protocols [21] and centralized *key*

*Work supported by the Defense Advanced Research Project Agency, Information Technology Office (DARPA-ITO), under contract DABT63-97-C-0031.

†Names appear in alphabetical order.

In proceedings of the 5th ACM Conference on Computer and Communication Security, November 2-5, 1998. San Francisco, CA.

[1] Informally, this means that old keys cannot be known to new members and new keys cannot be known to former members.

[2] The security is based on the polynomial indistinguishability of a Diffie-Hellman key from an arbitrary random value.

distribution protocols based on some form of a trusted third party (TTP). Although, in this paper we focus on (contributory) key agreement, we briefly note several features of centralized key distribution that make it unsuitable for DPGs:

- A TTP that generates and distributes keys for a multitude of groups is a single point of failure and a likely performance bottleneck.
- Since all group secrets are generated in one place, a TTP presents a very attractive attack target for adversaries. This is especially the case if a TTP serves as the key generation/distribution center for multiple groups.
- Environments with no hierarchy of trust are a poor match for centralized key transport. (For example, consider a group composed of members in different, and, perhaps competing, organizations or even different countries.)
- Some DPG environments (e.g., *ad hoc* wireless networks) are highly dynamic and no group member is present all the time. However, most key distribution protocols assume fixed centers.
- It might not be acceptable for a single party to generate the group key. For example, every party may need assurance that the resulting group key is *fresh* and *random* (e.g., in case the key is later used for computing digital signatures).
- Achieving perfect forward secrecy (Def. 3.7) and resistance to known-key attacks (Def. 3.8) in an efficient manner is very difficult in the centralized key distribution setting.

Although we argue in favor of distributed, contributory key agreement for DPGs, we also recognize the need for a central point of control for group membership operations such as adding and deleting members. This type of a role (group membership controller) serves only to synchronize the membership operations and prevent chaos. However, the existence and assignment of this role is orthogonal to key establishment and is largely a matter of policy.

3 Goals and Definitions

In addition to *key independence* alluded to above and resistance to all types of passive attacks, desired properties for a practical key agreement protocol typically include the following:

- Perfect Forward Secrecy (PFS)
- Resistance to Known-Key Attacks
- Key Authentication
- Key Confirmation and Key Integrity

All of these are necessary to achieve resistance to *active* attacks mounted by an increasingly powerful adversary. And, as always, ironclad security must be achievable with the lowest possible cost.

We now present some definitions for the above and other terminology used in this paper. (Some of these are adapted from Menezes et al. [18])

Definition 3.1 *A* **key agreement protocol** *is a key establishment technique whereby a shared secret key is derived by two or more specified parties as a function of information contributed by, or associated with, each of these, such that no party can predetermine the resulting value.*

Definition 3.2 *A key agreement protocol is* **contributory** *if each party equally contributes to the key and guarantees its freshness.*

For example, according to this definition, the basic two-party Diffie-Hellman protocol is contributory. On the other hand, the ElGamal one-pass [18] protocol is not contributory as only one of the parties contributes a fresh exponent.

Definition 3.3 *Let $\mathcal{R}$ be an n-party key agreement protocol, $\mathcal{M}$ be the set of protocol parties and let S_n be a secret key jointly generated as a result of $\mathcal{R}$. We say that $\mathcal{R}$ provides* **implicit key authentication** *if each $M_i \in \mathcal{M}$ is assured that no party $M_q \notin \mathcal{M}$ can learn the key S_n (unless aided by a dishonest $M_j \in \mathcal{M}$).*

Definition 3.4 *A protocol provides* **key confirmation** *if a party is assured that its peer (or a group thereof) actually has possession of a particular secret key.*

Definition 3.5 *A contributory key agreement protocol provides* **key integrity** *if a party is assured that its particular secret key is a function of* **only** *the individual contributions of all protocol parties. In particular, extraneous contribution(s) to the group key cannot be tolerated even if it does not afford the attacker(s) with any additional knowledge.*

Definition 3.6 *An* **authenticated group key agreement protocol** *is a key agreement protocol which provides implicit key authentication.*

Definition 3.7 *A protocol offers* **perfect forward secrecy (PFS)** *if compromise of a long-term key(s) cannot result in the compromise of past session keys.*

Definition 3.8 *A protocol is said to be vulnerable to* **known-key attack** *if compromise of past session keys allows: 1) a passive adversary to compromise future session keys, or 2) an active adversary to impersonate one of the protocol parties. (See [6, 5], for details.)*

4 Protocol Preliminaries

The following notation is used throughout the paper:

n	number of protocol parties (group members)
i, j	indices of group members
M_i	i-th group member; $i \in [1, n]$
G	unique subgroup of $\mathbb{Z}_p^*$ of order q with p, q prime
q	order of the algebraic group
α	exponentiation base; generator[3] in group G
x_i	long-term secret key of M_i
r_i	random (secret) exponent $\in \mathbb{Z}_q$ generated by M_i
S_n	group key shared among n members
$S_n(M_i)$	M_i's view on a group key
K_{ij}	long-term secret shared by M_i and M_j, with $i \neq j$

Throughout the paper, all arithmetic is performed in the cyclic group G of prime order q which is a subgroup of $\mathbb{Z}_p^*$ for a prime p such that $p = kq + 1$ for some small $k \in \mathbb{N}$.

[3] α can be computed by repeatedly selecting a random element $b \in \mathbb{Z}_p^*$ and computing $\alpha = b^{(p-1)/q} \bmod p$ until $\alpha \neq 1$.

No practical methods are known to compute partial information with respect to discrete logarithms (DL) in subgroup with this setting. Most DL-based schemes have been designed using a prime order subgroup. One of the advantages of working in such a group is that all the elements (except the unity element) are generators of the subgroup itself. Moreover, using subgroup of prime order seems to be a prudent habit [1]; it also results in increased efficiency.

When operating in subgroups it is important to take into account the attacks outlined in [1, 15]. To prevent masquerading or leaking of (even partial) information of the secret values, each party has to verify that the (purportedly random) values it receives are in fact elements of the subgroup.[4]

Note that p, q and α are public and common to all users. Since they need to be generated only once (or very seldom), it is desirable to make the generation process unpredictable but verifiable to prevent the selection of weak or special primes. One approach is to use the NIST method for selecting DSA primes as described in the FIPS 186 document [13].

In this context, the ability of an active adversary C to modify or inject messages is quite "limited". In fact, any message m can be written as $m = \alpha^c \bmod p$, where α is a generator of the unique cyclic subgroup of $\mathbb{Z}_p^*$ having order q and c some exponent (perhaps unknown). Later on, we will suppose that the adversary C operates on this type of elements.

5 Authenticated 2-party Key Agreement

In this section we develop an extension to the Diffie-Hellman (DH) [11] key agreement protocol that provides key authentication. We explicitly avoid requiring any cryptographic tools (e.g., symmetric encryption) other than those necessary for a normal DH key agreement.

Before turning to the actual protocol, it is important to emphasize that there already exist secure protocols for authenticated DH-based key agreement. However, some are not contributory (such as El Gamal), some require more messages or assume a priori access to certified long-term keys, while others do not offer PFS or are vulnerable to so-called *known-key attacks*. (For example, some of the protocols in the MTI protocol family [17].) An additional goal is to come up with a protocol that is easily extendible from 2- to n-party key agreement. Yet another, perhaps superficial, issue has to do with minimizing the security dependencies of a protocol. For example, an authenticated DH-based key agreement can be easily constructed with the aid of conventional encryption. The security of the underlying protocol would then be dependent not only on the difficulty of, for example, the Diffie-Hellman Decision (DDH) problem (as far as key agreement) but also on the strength of the conventional encryption (as far as key authentication). Ideally, it should be possible to base all the security properties of a given protocol on a *single* hard problem such as the DDH problem in prime-order subgroups.

One protocol that satisfies the above criteria is A-DH, shown in Figure 1. It provides implicit key authentication as stated by the following theorem.

[4] Verifying the order of an element x by checking, for example, that $x^{(p-1)/q} \bmod p \neq 1$, is rather expensive. If p and q are carefully chosen such that the other prime factors of $\phi(p)/2$ are close to the order of q, we can exclude elements of small order in an efficient manner by checking that $x^2 \neq 1 \bmod q$. Although this seems to be sufficient, the security of this method needs further study [15].

Protocol A-DH: Let p, q, G be as defined above, and let α be a generator of G.
Initialization. Let x_1 and x_2 be two integers such that $1 \leq x_1, x_2 \leq q-1$. Let M_1 and M_2 be two parties wishing to share a key and let $(x_1, \alpha^{x_1} \bmod p)$ and $(x_2, \alpha^{x_2} \bmod p)$ be the secret and public keys of M_1 and M_2, respectively. Thus, the public values of the system are $(p, q, \alpha, \alpha^{x_1}, \alpha^{x_2})$. The actual protocol is as follows:
Round 1:
M_1 selects random $r_1 \in_R \mathbb{Z}_q^*$,
$M_1 \longrightarrow M_2 : \alpha^{r_1} \bmod p$
Round 2:
M_2 selects random $r_2 \in_R \mathbb{Z}_q^*$, and computes
$K = F(\alpha^{x_1 x_2} \bmod p)$.
$M_2 \longrightarrow M_1 : \alpha^{r_2 K} \bmod p$
When M_1 receives $J = \alpha^{r_2 K} \bmod p$, computes $K^{-1} \bmod q$ and then $J^{r_1 K^{-1}} \bmod p$. The shared secret key is $S_2 = \alpha^{r_1 r_2} \bmod p$. We can set the function $F()$ such that $F(x) = x \bmod q$ or $F(x) = h(x)$ where h is an appropriate hash function $h : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$.

Figure 1: Authenticated Diffie-Hellman (A-DH)

Theorem 5.1 *The A-DH protocol is a contributory authenticated key agreement protocol.*

Proof (sketch): From the construction of the resultant session key $S_2 = \alpha^{r_1 r_2}$ it is evident that A-DH is *contributory*. Let C be an active adversary able to modify, delay, or inject messages.
Attack on M_2: Let $S_2(M_2)$ be the key computed by M_2. It can be expressed as $S_2(M_2) = \alpha^{c_1 r_2}$ where c_1 is a quantity possibly known to C, i.e., C can substitute the first flow with α^{c_1}. Then, computing $\alpha^{c_1 r_2}$ requires C to compute α^{r_2}. However, the only expression containing α^{r_2} is $\alpha^{r_2 K}$ in the second protocol flow. But, computing α^{r_2} from $\alpha^{r_2 K}$ is intractable without the knowledge of K.
Attack on M_1: The key computed by M_1 is $S_2(M_1) = \alpha^{c_2 r_1 K^{-1}}$ where c_2 is possibly chosen by C.

1: Suppose that $c_2 = c_3 K$ where c_3 is polynomially independent of K and known to C. Then: $S_2(M_1) = \alpha^{c_3 K K^{-1} r_1} = \alpha^{c_3 r_1}$ However, computing $\alpha^{c_3 K}$ such that c_3 is known to C is intractable without computing α^K which, in turn, is intractable without computing the inverse of r_2.

2: Suppose now that c_2 is polynomially independent of K. Since $S_2(M_1)$ is a function of K^{-1}, it is not computable by C. □

On top of implicit key authentication, a practical key agreement protocol must: 1) provide perfect forward secrecy and 2) be resistant to known-key attacks. These two properties are considered in the following theorems.

Theorem 5.2 *The A-DH protocol provides perfect forward secrecy (PFS).*

Proof (sketch): Suppose that the long-term key $K = F(\alpha^{x_1 x_2} \bmod p)$ is compromised. Then, an adversary knows both $\alpha^{r_1} \bmod p$ and $\alpha^{(r_2 K) K^{-1}} \equiv \alpha^{r_2} \bmod p$. Given these, computing the session key $S_2 = \alpha^{r_1 r_2} \bmod p$ is equivalent to solving the DH problem in prime-order subgroups. □

Theorem 5.3 *The A-DH protocol is resistant to known-key attacks.*

Proof (sketch): Let $S_2(M_1)$ and $S_2(M_2)$ be the session keys computed by M_1 and M_2, respectively. We can write $S_2(M_1) = \alpha^{c_1 r_1 K^{-1}}$ and $S_2(M_2) = \alpha^{c_2 r_2}$ where c_1, c_2 are quantities *possibly* known to an active adversary C. Therefore, the only relevant values C can know are: α^{r_1}, $\alpha^{r_2 K}$, $\alpha^{r_1 K^{-1}}$, α^{r_2} and the public keys of M_1 and M_2. Hence, finding K is based on solving the DL problem while computing α^K or $\alpha^{K^{-1}}$ is **at least** as difficult as the DH problem in prime-order subgroups. □

A nice feature of the A-DH protocol is that it does not require *a priori* knowledge of the long-term public keys of the parties involved. In fact, certificates can be piggy-backed onto existing protocol messages. This is a consequence of the protocol's "asymmetry".

6 Authenticated Group Key Agreement

In [20], a class of *generic n-party DH protocols* is defined. The security of the entire protocol class is shown secure against passive adversaries based on the intractability of the Diffie-Hellman Decision (DDH) problem. Several concrete protocols were demonstrated that fit the requirements of DPGs. Moreover, these protocols are shown to be optimal with respect to certain measures of protocol complexity [20, 2]. In this section we extend the GDH protocols to provide implicit key authentication. In doing so, we make use of the A-DH protocol discussed in section 5.

6.1 Authenticated GDH.2 protocol

Two practical protocols: GDH.2 and GDH.3 are defined in [20]. (Another protocol, GDH.1, is used for demonstration purposes only.) The GDH.2 protocol is minimal in terms of the total number of protocol messages. GDH.3, on the other hand, aims to minimize computation costs. Although, the discussion below focuses on extending GDH.2, we note that all of the techniques we consider are easily adapted to GDH.3.

Protocol GDH.2: Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be a set of users wishing to share a key S_n. The GDH.2 protocol executes in n rounds. In the first stage ($n-1$ rounds) contributions are collected from individual group members and then, in the second stage (n-th round) the group keying material is broadcast. The actual protocol is as follows:
Initialization. Let p be a prime and q a prime divisor of $p-1$. Let G be the unique cyclic subgroup of $\mathbb{Z}_p^*$ of order q, and let α be a generator of G.
Round i $(0 < i < n)$**:**
1) M_i selects random $r_i \in_R \mathbb{Z}_q^*$.
2) $M_i \longrightarrow M_{i+1}$: $\{\alpha^{\frac{r_1 \cdots r_i}{r_j}} | j \in [1,i]\}$, $\alpha^{r_1 \cdots r_i}$
Round n**:**
1) M_n selects random $r_n \in_R \mathbb{Z}_q^*$.
2) $M_n \longrightarrow$ ALL M_i: $\{\alpha^{\frac{r_1 \cdots r_n}{r_i}} | i \in [1,n[\}$

Figure 2: Group Diffie-Hellman (GDH.2)

We begin with a brief overview of GDH.2 in Figure 2. This basic protocol can be easily amended to provide implicit key authentication in an efficient manner. This variation (A-GDH.2, shown in Figure 3) differs from the basic protocol only in the last round, hence we are only concerned therewith.

We assume that M_n shares (or is able to share) with each M_i a distinct secret K_{in}.

For example, we can set $K_{in} = F(\alpha^{x_i \cdot x_n} \bmod p)$ with $i \in [1, n-1]$. Where x_i is a secret long term exponent selected by every M_i $(1 \leq x_i \leq q-1)$ and $\alpha^{x_i} \bmod p$ is the corresponding long-term public key of M_i.

Protocol A-GDH.2:
Rounds 1 **to** $n-1$**:** identical to GDH.2
Round n**:**
1) M_n selects random $r_n \in_R \mathbb{Z}_q^*$
2) $M_n \longrightarrow$ ALL M_i: $\{\alpha^{\frac{r_1 \cdots r_n}{r_i} \cdot K_{in}} | i \in [1,n[\}$.
Upon receipt of the above, every M_i computes: $\alpha^{(\frac{r_1 \cdots r_n}{r_i} \cdot K_{in}) \cdot K_{in}^{-1} \cdot r_i} = \alpha^{r_1 \cdots r_n} = S_n$.

Figure 3: Authenticated Group Diffie-Hellman (A-GDH.2)

In this protocol, each group member obtains an (implicitly) authenticated shared key with M_n. Moreover, if we trust M_n to behave correctly, a group member can also be sure the key shared with M_n is the same key M_n shares with all other members.

Theorem 6.1 *A-GDH.2 is a contributory authenticated key agreement protocol.*

Proof (sketch): From the construction of the resultant session key $S_n = \alpha^{r_1 \cdots r_n}$ it is evident that A-GDH.2 is *contributory.*

Let C be an active adversary who can modify, delay, or inject messages. C's goal is to share a key with either M_i, for $i \in [1, n[$, or with M_n by masquerading as some M_i. In case of the former, all considerations of the proof in theorem 5.1 apply.

Assume that C wants to masquerade as M_i. Let $S_n(M_n)$ be the key computed by M_n. It can be expressed as:

$$S_n(M_n) = \alpha^{c_n \cdot r_n}$$

where c_n is a quantity possibly known to C, i.e., in round $n-1$ C can replace $\alpha^{r_1 \cdots r_{n-1}}$ with α^{c_n} in the message from M_{n-1} to M_n. C can also replace the other $(n-1)$ values in the same message:

$$\alpha^{\frac{r_1 \cdots r_{n-1}}{r_j}} \; (j \in [1,n[) \rightarrow \alpha^{c_j}$$

for some known c_j-s. This will cause M_n to output in the last round:

$$\{\, \alpha^{c_j \cdot r_n \cdot K_{jn}} \mid j \in [1,n[\,\}$$

Now, since C knows all c_j, she also knows (or can easily compute) all c_j^{-1}. Hence, C can compute:

$$\{\, \alpha^{r_n \cdot K_{jn}} \mid j \in [1,n[\,\}$$

However, extracting information of $S_n(M_n)$ is intractable if the DDH problem in prime-order subgroup is hard. □

Theorem 6.2 *The A-GDH.2 protocol provides perfect forward secrecy.*

Proof (sketch): Suppose that all long-term keys $\{K_{in} \mid i \in [1,n[\ \}$ are compromised. Then, our adversary is able to compute a subset of $V = \{\alpha^{\Pi(S)} | \ \mathcal{S} \subset \{r_1, \ldots, r_n\}\}$. But, as shown in [20], given V, it is intractable to find information on the group key $S_n = \alpha^{r_1 \cdots r_n}$, if the DDH problem in prime-order subgroup is hard. □

Resistance to known-key attacks. A-GDH.2 is resistant to *passive* known-key attacks since the the session keys do not contain any long-term information. Resistance to *active* known-key attacks, on the other hand, is somewhat dubious for reasons stated below.

Let $S_n(M_i)$ be the session key computed by each M_i. For $0 < i < n-1$ we can re-write it as $\alpha^{c_i r_i K_{in}^{-1}}$. For M_n, $S_n(M_n) = \alpha^{c_n r_n}$ where c_i is a quantity possibly known to the adversary C. C also knows a subset of $\{\alpha^{\Pi(S)} | \ \mathcal{S} \subset \{r_1, \ldots, r_n\}\}$. Using these to find $\alpha^{K_{in}}$ or $\alpha^{K_{in}^{-1}}$ (for $1 \leq i \leq n-1$), is intractable if the DDH problem in prime-order subgroup is hard.

Despite the above, some forms of active known-key attacks are possible. Suppose, for example, that C tries to impersonate M_1. It starts by sending α^{c_1} to M_1 in the last protocol round (where c_1 is selected by C). As a result, M_1 computes: $S_n(M_1) = \alpha^{c_1 r_1 K_{1n}^{-1}}$. Since this key is corrupted (i.e., not shared with any other M_i), we can assume that M_1 will detect the problem and re-run the protocol. Suppose further that C somehow manages to discover this malformed key.[5] In the next protocol run, C can substitute the message from M_{n-1} to M_n with:

$$\alpha^{c_1 r_1 K_{1n}^{-1}}, \ldots, \alpha^{c_1 r_1}$$

In other words, C substitutes only the first and the last sub-keys in the flow; the rest of the values are unchanged. This causes M_n to compute $S_n(M_n) = (\alpha^{c_1 r_1})^{r_n}$. M_n will also compute (as a sub-key for M_1):

$$(\alpha^{c_1 r_1 K_{1n}^{-1}})^{r_n K_{1n}} = \alpha^{c_1 r_1 r_n}$$

and will broadcast this value in the last protocol round. The end-result is that C shares a key with M_n.

There are a few issues with this type of attack. First, it relies on the lack of key confirmation which we discuss later in the paper. Second, it does not fit the usual definition of a known-key attack since C is only able to share a key with M_n, not with the rest of the group. (We note that known-key attacks were only defined in the context of 2-party protocols. Their definition in a group setting remains to be worked out.) Also, as noted in [5], a simple cure for known-key attacks is by setting $S_n = h(S_n(M_i))$ where $h()$ is an appropriate collision-resistant hash function such as SHA [14].

6.2 Complete Group Key Authentication

The above protocol (A-GDH.2) achieves implicit key authentication in a relatively *weak* form since the key is not directly authenticated between an arbitrary M_i and M_j ($i \neq j$). Instead, all key authentication is performed through M_n. This may suffice in some environments, e.g., when the exact membership of the group is not divulged to the individual M_i's. Another reason may be that M_n is an entity trusted by all other members, e.g., M_n is an authentication server.

According to Definition 3.3, A-GDH.2 will result in all participants agreeing on the same key if we assume M_n behaves correctly. However, no one – including M_n – can be sure of other members' participation. In fact, one or more of the intended group members may be "skipped" without detection. Also, a dishonest M_n could partition the group into two without detection by group members. On the one hand, we assume a certain degree of trust in all group members (including M_n), e.g., not to reveal the group key to outsiders. On the other hand, one might want to limit this trust when it comes to group membership, i.e., M_n might not be universally trusted to faithfully include all (and only) group members.

In more concrete terms, our failure model is based on:

> A malicious insider (group member) seeking to alter the group membership by excluding some members – possibly including itself – from participation in key agreement. For example, this may translate into attempting to physically circumvent certain group members or corrupting intermediate values that subsequently contribute to the excluded members' keys.

On the other hand, our failure model specifically **excludes**:

> A malicious insider revealing the group key or any other group (or its own) secrets to outsiders.
>
> An insider (malicious ot otherwise) exhibiting any other form of anomalous behavior.

In order to clarify the above, we introduce the following feature:

Definition 6.3 *Let $\mathcal{R}$ be an n-party key agreement protocol and $\mathcal{M}$ be a set of protocol parties (DPG). We say that $\mathcal{R}$ is a* **complete group key authentication** *protocol if, for every i, j ($0 < i \neq j \leq n$) M_i and M_j compute the same key $S_{i,j}$* **only if** *$S_{i,j}$ has been contributed to by every $M_p \in \mathcal{M}$. (Assuming that M_i and M_j have the* **same view** *of the group membership.)*

An alternative definition for complete group key authentication is as authenticated group key agreement for all (M_i, M_j) pairs ($0 < i \neq j \leq n$).

A-GDH.2 can be augmented to provide complete group key authentication as shown in Figure 4. (To better illustrate SA-GDH.2 and its differences with respect to A-GDH.2, a 4-party example is shown in Figure 5.)

The biggest change in the present protocol, SA-GDH.2, is the requirement for *a priori* availability of all members' long-term credentials. In effect, each M_i is required to have two shared keys (one in each direction) with every other M_j. For every distinct *ordered* pair $< i, j >$ ($0 < i \neq j \leq n$) let $< K_{ij}, K_{ij}^{-1} >$ denote the *unidirectional* key shared by M_i and M_j and its inverse, respectively. Although it may appear otherwise, individual key inverses of the form K_{ij}^{-1} **do not** need to be computed (see below).

Drawbacks: SA-GDH.2 is clearly more expensive than A-GDH.2. First, it requires $n-1$ exponentiations from every M_i during stage 1 as opposed to i in A-GDH.2. Second, if pairwise keys (K_{ij}) are not pre-computed, as many as $(n-1)$ additional exponentiations must be performed. Note that in the last round, only one exponentiation is done since M_i can pre-compute the value: $(K_{i1}^{-1} \cdots K_{in}^{-1}) \cdot r_i$ immediately following the i-th round.

[5] This assumption is what makes active known-key attacks very unlikely in practice.

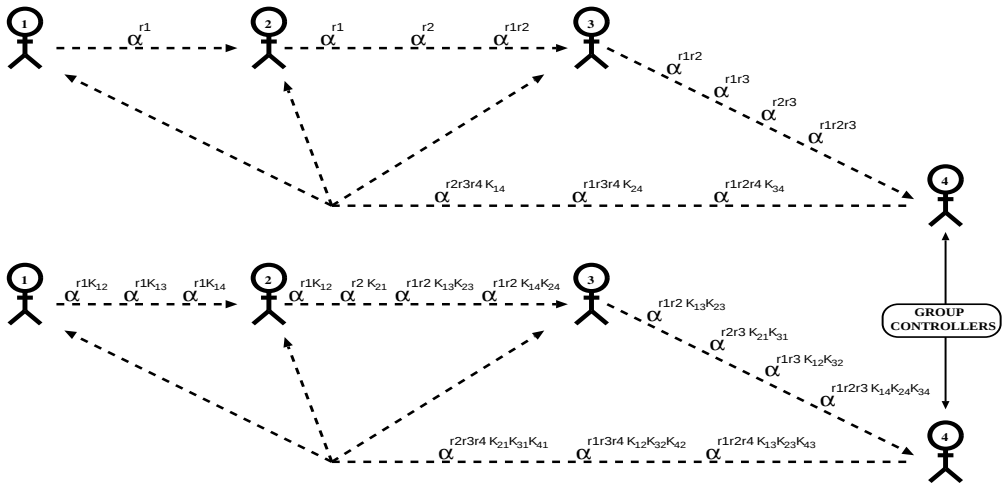

Figure 5: An example/comparison of A-GDH and SA-GDH.2

Advantages: unlike A-GDH.2, SA-GDH.2 allows each member to be explicitly aware of the exact group membership. This may be desired in some environments. Also, the protocol is computationally symmetric, i.e., each member performs the same sequence of computational steps and the same number of exponentiations.

Theorem 6.5 *SA-GDH.2 offers complete group key authentication.*

Proof (sketch): Suppose M_i and M_j compute the same key while following the protocol correctly. Let $K_n = S_n(M_i) = S_n(M_j)$ and, suppose also, that some $M_p \in \mathcal{M}$, $(p \neq i, j)$ has not contributed to this key. Let V_i, V_j denote the values received by M_i and M_j, respectively, in the last round of the protocol. Recall that:

$$S_n(M_i) = (V_i)^{(K_{1i}^{-1} \cdots K_{ni}^{-1}) \cdot r_i}$$

and, similarly:

$$S_n(M_j) = (V_j)^{(K_{1j}^{-1} \cdots K_{nj}^{-1}) \cdot r_j} =$$

Since all other group members have contributed to the key, we can re-write V_i as (V_j is similar):

$$V_i = \alpha^{(\frac{r_1 \cdots r_n}{r_p r_i}) \cdot (\frac{K_{1i}^{-1} \cdots K_{ni}^{-1}}{K_{pi}^{-1}})}$$

Then,

$$S_n(M_i) = \alpha^{(\frac{r_1 \cdots r_n}{r_p}) \cdot K_{pi}^{-1}}$$

which must equal:

$$S_n(M_j) = \alpha^{(\frac{r_1 \cdots r_n}{r_p}) \cdot K_{pj}^{-1}}$$

However, this is impossible since K_{pi}^{-1} and K_{pj}^{-1} are distinct and secret values. □

Remark 6.6 *An interesting feature of SA-GDH.2 is its resistance to known-key attacks. Although we do not to treat this topic in detail, it can be easily observed that an attack of the sort described in Section 6.1 cannot succeed against SA-GDH.2.*

6.3 Efficiency Summary

We now consider the costs incurred by the protocols described above. The following two tables summarize, respectively, the communication and computation overhead of the following:

- GDH.2 – plain group key agreement [20].
- A-GDH.2 - authenticated group key agreement as specified in Section 6.1. Long-term keys K_{in} are assumed to be pre-computed.
- A-GDH.2* - same as A-GDH.2 but long-term keys K_{in} are computed as part of the protocol; this also implies that public exponents of all group members must be accumulated in the course of the protocol.
- SA-GDH.2 – complete group key authentication

The first table illustrates the communication, and the second computation, costs. The latter is broken down into exponentiation, inverse computation and multiplication. Exponentiation is clearly the costliest operation as it requires $O(log^3 p)$ bit operations in $\mathbb{Z}_p^*$. Given a and p, finding the inverse of $a \in \mathbb{Z}_p^*$ requires only $O(log^2 p)$ bit operations (using the extended Euclidean algorithm). Similarly, the multiplication of a and b modulo p requires $O(log^2 p)$ bit operations. (See [12], [18] for a complete treatment of modular operations.)

The only somewhat surprising element of this analysis is the relatively low additional cost of SA-GDH.2 as compared to that of GDH.2 and A-GDH.2. Considering that it offers complete group key authentication and several other useful services (when coupled with key confirmation; see below) the added overhead is well justified.

7 New Services in Group Setting

As mentioned in the introduction, *key confirmation* (Def. 3.4 and [18]) is an important feature in key agreement protocols. Its purpose is to convince one or more parties that its peer (or a group thereof) is in possession of the key. It can be argued that key confirmation is not absolutely necessary if communication immediately follows key agreement, i.e., if a proper key is subsequently used for bi-directional data flows, key confirmation is achieved as a side-effect. However, in

Communication Costs:	Protocols: GDH.2	A-GDH.2	A.GDH.2*	SA-GDH.2
rounds	n	n	n	n
broadcasts	1	1	1	1
total msgs	n	n	n	n
total bandwidth	$(n^2+n)/2-1$	$(n^2+n)/2-1$	n^2	n^2
msgs sent per M_i	1	1	1	1
msgs received per M_i	2	2	2	2

Computation Costs:	Protocols: GDH.2	A-GDH.2	A.GDH.2*	SA-GDH.2
exponentiations for M_i	$i+1$	$i+1$	$i+2$	n
exponentiations for M_n	n	n	$2n-1$	n
total exponentiations	$(n^2+3n)/2-1$	$(n^2+3n)/2-1$	$(n^2+4n)/2-2$	n^2
inverses for M_i			1	1
inverses for M_n				1
total inverses			$n-1$	n
multiplications for M_i		1	1	$2n-2$
multiplications for M_n		$n-1$	$n-1$	$2n-2$
total multiplications		$2n-2$	$2n-2$	$2n^2-2n$

general, it is desirable to bundle key confirmation with key agreement for the following reasons:

1. it makes key agreement both a more robust and more autonomous operation
2. doing otherwise can lead to an incorrectly computed key not being detected later (since there may be a delay between key agreement and actual data communication)

On the other hand, it is not clear what key confirmation means in a peer group setting. Complete key confirmation (in the spirit of complete key authentication) would make it necessary for all group members to compute the key and then confirm to all other members the knowledge of the key. This would entail, at the very least, one round of n simultaneous broadcasts. We take a more practical approach by concentrating on key confirmation emanating from the group controller, the first group member to compute the actual key.

It turns out that the construction of A-GDH.2 (and SA-GDH.2) makes key confirmation fairly easy to add. The only change to both protocols is the addition to the last protocol message (the broadcast from M_n) of:

$$\alpha^{F(S_n(M_n))}$$

where $S_n(M_n)$ denotes the key as computed by M_n and $F()$ is as previously defined.

Upon receipt of the broadcast, each M_i computes its key $S_n(M_i)$ as before. Then, M_i verifies the computed key:

$$\alpha^{F(S_n(M_i))} \stackrel{?}{=} \alpha^{F(S_n(M_n))}$$

In both A-GDH.2 and SA-GDH.2, key confirmation coupled with implicit key authentication, has a nice side-effect of providing *entity authentication* of M_n to all other group members. Informally, this is because the upflow message in round i can be viewed as a random challenge (r_i being M_i's nonce) submitted to M_n (indirectly, through all other $M_j; j > i$). The last broadcast, then, can be viewed as M_n's reply to M_i's challenge encrypted under a secret key shared among M_i and M_n. To support our claim that the above results in entity authentication of M_n we need to show that M_n's reply is *fresh*. (That M_n's reply is *authentic* has been shown in Section 6.1.) Freshness, however, is evident from the way M_i computes the key: by exponentiating the value received from M_n with $(r_i \cdot K_{in}^{-1})$.

Remark 7.1 *In SA-GDH.2, for each M_i, key confirmation also results in* entity authentication *of all M_j, for $i < j \leq n$.*

Including key confirmation in SA-GDH.2 leads us to an interesting observation:

> At the end of the protocol, each M_i knows that the key it holds, $S_n(M_i)$, has been contributed to by every group member.

This follows directly from the *complete group key authentication* property coupled with key confirmation. Recall that the former assures that, if any two distinct parties (M_i and M_j) share a key, that key must be contributed to by every group member. Adding key confirmation allows us to achieve a stronger goal: any group member can unilaterally establish that it is in possession of a correct key which has been contributed to by every member. This is both a novel and important feature of SA-GDH.2 and *a new security service* unique to group key agreement.

Definition 7.2 *(informal) A group key agreement protocol offers* **group integrity** *if each protocol party is assured of every other protocol party's participation in the protocol.*

Group integrity should not be confused with entity authentication. It is a *weaker* notion since group integrity does not guarantee freshness/timeliness. It only guarantees all parties' participation in the protocol and, likewise, all parties' awareness of the group membership.

Definition 7.3 *(informal) A group key agreement protocol is* **verifiable contributory** *if each protocol party is assured of every other protocol party's contribution to the group key.*

Protocol SA-GDH.2:
Round i $(0 < i < n)$**:**
1) M_i receives a set of n intermediate values: $\{V_k | 1 \leq k \leq n\}$. ($M_1$ which can be thought of as receiving an empty set in the first round):

$$V_k = \begin{cases} \alpha^{(\frac{r_1 \cdots r_{i-1}}{r_k}) \cdot (K_{k1} \cdots K_{k(i-1)})} & \textbf{if } k \leq (i-1) \\ \alpha^{(r_1 \cdots r_{i-1}) \cdot (K_{k1} \cdots K_{k(i-1)})} & \textbf{if } k > (i-1) \end{cases}$$

2) M_i updates each V_k as follows:

$$V_k = \begin{cases} (V_k)^{K_{ik} \cdot r_i} = \alpha^{(\frac{r_1 \cdots r_i}{r_k}) \cdot (K_{k1} \cdots K_{ki})} & \textbf{if } k < i \\ (V_k)^{K_{ik} \cdot r_i} = \alpha^{(r_1 \cdots r_i) \cdot (K_{k1} \cdots K_{ki})} & \textbf{if } k > i \\ V_k & \textbf{if } k = i \end{cases}$$

Remark 6.4 In the initial round M_1 sets $V_1 = \alpha^1$.

Round n**:**
1) M_n broadcasts a set of all V_k values to the group.
2) On receipt, each M_i selects the appropriate V_i where:

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

M_i proceeds to compute:

$$(V_i)^{(K_{1i}^{-1} \cdots K_{ni}^{-1}) \cdot r_i} = \alpha^{r_1 \cdots r_n}$$

For the above, instead of computing $n-1$ individual key inverses of the form K_{ji}^{-1}, each M_i computes only a single *compound* inverse $P_i^{-1} = (K_{1i}^{-1} \cdots K_{ni}^{-1})$ where $P_i = (K_{1i} \cdots K_{ni})$

Figure 4: Group Diffie-Hellman with Complete Key Authentication (SA-GDH.2)

Note that *verifiable contributory* implies *group integrity* while the reverse is not true. For example, group integrity can be obtained by requiring every M_i to sign and forward (to all others) a statement certifying to its participation in the protocol. Also, verifiable contributory property does not imply that a group key is *not contributed to by an outside party*. As discussed in the section 7.1, an adversary can still inject some input into the group key.

7.1 The Elusive Key Integrity

Key integrity (Def. 3.5) is orthogonal to both key authentication and key confirmation. A key agreement protocol may offer one or both of the latter while at the same time not guaranteeing key integrity. Consider the following (3-party) SA-GDH.2 example:

This protocol offers complete group key authentication, key confirmation and, entity authentication of M_3. At the end, all parties wind up computing the same key. However, an adversary can exponentiate by a constant all values sent in round 1 (and/or round 2) and remain undetected. Suppose the adversary simply squares all values in round 2. Then, what M_3 actually receives is: $\alpha^{r_1 \cdot K_{12} \cdot 2}$, $\alpha^{r_1 \cdot K_{13} \cdot r_2 \cdot K_{23} \cdot 2}$, $\alpha^{r_2 \cdot K_{21} \cdot 2}$

As a result, M_3 computes $S_3(3) = \alpha^{r_1 \cdot r_2 \cdot r_3 \cdot 2}$ and both M_1 and M_2 compute the same value, i.e., the quadratic residue of the intended key. The key confirmation check does not help since the adversary introduces its input before M_n computes the group key.

Protocol SA-GDH.2 (example):
Round 1: M_1 selects random $r_1 \in_R \mathbb{Z}_q^*$.
$M_1 \longrightarrow M_2 : \alpha^{r_1 \cdot K_{12}}, \alpha^{r_1 \cdot K_{13}}$
Round 2: M_2 selects random $r_2 \in_R \mathbb{Z}_q^*$.
$M_2 \longrightarrow M_3 : \alpha^{r_1 \cdot K_{12}}, \alpha^{r_1 \cdot K_{13} \cdot r_2 \cdot K_{23}}, \alpha^{r_2 \cdot K_{21}}$
Round 3: M_3 selects random $r_3 \in_R \mathbb{Z}_q^*$, computes group key $S_3(3)$ and broadcasts:
$M_3 \longrightarrow M_1, M_2 : \alpha^{r_1 \cdot K_{12} \cdot K_{32} \cdot r_3}, \alpha^{r_2 \cdot K_{21} \cdot K_{31} \cdot r_3}, \alpha^{F(S_3(3))}$
M_2 computes $S_3(2)$, M_1 computes $S_3(1)$ and, finally, both M_1 and M_2 confirm the correctness of their respective keys against $\alpha^{F(S_3(3))}$.

We observe that, in SA-GDH.2, the adversary is only able to introduce multiplicative (in the exponent) input, i.e., it can cause the key to be K^C for some value C. The construction of the protocol precludes the adversary from introducing any other type of input, e.g., additive in the exponent.

This leads us to pose the following question:

> How important is key integrity in a *verifiable contributory* key agreement protocol?

In practice, we expect that key integrity can be easily assured via an external data integrity mechanism (e.g., SSL) used *hop-by-hop* in the upflow stage of the protocol. Consequently, if every protocol message between M_i and M_{i+1} in the i-th $(0 < i < n)$ protocol round is integrity-protected, the adversary is no longer able to introduce any "noise" into the group key. Note that the last, broadcast message does not need to be protected; any modification will be detected by the key confirmation check.

8 Other Security Services

The primary motivation for obtaining a group key (in any manner; whether centralized or contributory) is the ability to communicate **securely and efficiently** once a key is established. If all DPG members share a key, they can communicate using symmetric encryption. This is more efficient than schemes not requiring key establishment.

For example, key establishment can be avoided as follows. A DPG member encrypts a message using a symmetric encryption function with a secret key K and then sends the cipher-text to the entire group along with $n-1$ versions of the key K; each encrypted using a public key of a member. Although this simple scheme has no (cryptographic) startup overhead, it is not contributory and becomes too expensive if the group is large or the volume of message traffic is high. Furthermore, it requires every member to be aware of the exact group membership at all times; something that can (if desired) be avoided with key agreement.

We believe that there are other incentives to consider. In particular, a shared group key can be used to provide a number of useful services (in an efficient manner):

- Authentication to outsiders
- Intra-group authentication
- Non-repudiation of group membership
- Private communication within group
- Private communication between outsiders and group

- Group signatures

For example, we can use a secret group key (such as the one agreed upon in A-GDH.2) to derive a corresponding group Diffie-Hellman public key which can be subsequently embedded in a group certificate. This would allow any group member to use DSA [13] (or any El Gamal family) signatures to authenticate itself (as a group member) to both insiders and outsiders. The same group public key can be viewed as long-term group Diffie-Hellman exponent and outsiders (including other groups) can establish shared keys with the entire group in a trivial manner. Similarly, a group secret key can be used to derive an El Gamal public key-pair; the public component thereof can be embedded in a group certificate. Outsiders can then use this key with El Gamal public key encryption to communicate in secret with the entire group.

9 Group Key Agreement and Byzantine Agreement

Group key agreement (GKA), in general, has similarities to the well-known byzantine agreement (BA) problem ([16]) but there are a number of distinguishing features. The fault model in GKA is not byzantine since we certain degree of trust is assumed among the group members, e.g., not to reveal the group key.

The standard BA requirements are: agreement, validity and termination. The validity requirement usually means: if all honest participants have the same input then they will agree on that value, otherwise they will agree on an an arbitrary value. Although termination and agreement would be required by complete authenticated key agreement too, the validity requirement is quite different, namely that the agreement is private to the participants[6] and that it is both fresh and random. Therefore, we claim that BA alone is not enough to build a robust GKA protocol.[7]

On the other hand, GKA has similarities with secure multiparty computation (SMPC, e.g [9, 10]). In fact, GKA can be viewed as a special case of SMPC. However, we note that general SMPC techniques typically yield highly inefficient protocols.

10 Conclusions: On-going and Future Work

This paper represents the third tier in developing security protocols and services for DPGs. The first tier was provided by group Diffie-Hellman key agreement [20] and the second, by extensions of the latter to support group membership changes [21]. This paper incorporates other important services (key authentication, key confirmation and entity authentication) into group key agreement.

We are currently working on the prototype implementation of the protocols described above. This includes both GDH.2-based and GDH.3-based protocols. (GDH.3 is a key agreement model aimed at minimizing computations by group members [20]; protocols presented above are easily grafted onto GDH.3.) In addition, we are designing authenticated key agreement protocols based on the Burmester-Desmedt model [7, 8] which is very efficient in certain environments, e.g., broadcast LANs. Our long-term goal is to develop a general-purpose toolkit for key agreement and related security services in DPGs. Initial clients for the toolkit may include voice conferencing over IP, replicated Web servers and private (closed) mailing lists.

In summary, this work is merely an initial attempt to analyze the requirements and issues in authenticated, contributory key agreement for DPGs. It is quite likely that the protocols presented here can be improved. We anticipate that practical experience with real DPG applications will result in a better understanding of group security needs and services.

[6] Note that BA protocols in general do not care about confidentiality.

[7] Despite the above, BA could be used for key confirmation (Section 7) but that would represent overkill: BA protocols in the best-possible settings (signatures) require at least $(t+1)$ rounds to tolerate t failures. If we set $t = 0$ (since we do not worry about byzantine faults) we still need a parallel broadcast of n signatures which is rather costly. Moreover, the benefits of BA over the simple key confirmation method sketched in Section 7 are unclear.

11 Acknowledgements

The authors gratefully acknowledge the comments of M. Waidner, M. Reiter and the anonymous referees.

References

[1] R. Anderson and S. Vaudenay. Minding your p's and q's. In *Advances in Cryptology – Asiacrypt'96*, 1996.

[2] C. Becker and U. Wille. Communication complexity of group key distribution. In *ACM Conference on Computer and Communication Security*, November 1998.

[3] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *ACM Symposium on Theory of Computing*, 1998.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO*, 1993.

[5] M. Burmester. On the risk of opening distributed keys. In *Advances in Cryptology – CRYPTO*, 1994.

[6] M. Burmester and Y. Desmedt. Towards practical proven secure authenticated key distribution. In *ACM Conference on Computer and Communication Security*, 1993.

[7] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology – EUROCRYPT*, 1994.

[8] M. Burmester and Y. Desmedt. Efficient and secure conference key distribution. In *Cambridge Workshop on Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 119–129. Springer-Verlag, Berlin Germany, April 1996.

[9] R. Canetti. Studies in secure multiparty computation and applications. PhD Thesis, Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, May 1995.

[10] D. Chaum, C. Crepeau, and I. Damgaard. Multiparty unconditional secure protocols. In *ACM Symposium on Theory of Computing*, 1988.

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

[12] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, Berlin Germany, Berlin, 1987.

[13] NIST Computer Systems Laboratory. Digital signature standard (draft). FIPS PUB 186, May 1994.

[14] NIST Computer Systems Laboratory. Secure hash standard (draft). FIPS PUB 180-1, May 1994.

[15] C. Hoon Lim and P. Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Advances in Cryptology – CRYPTO*, 1997.

[16] N. Lynch. Distributed algorithms. Morgan Kaufmann, San Francisco 1996.

[17] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *Transactions of the IECE*, E69, 1986.

[18] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 0-8493-8523-7.

[19] J. Smith and F. Weingarten. Research challenges for the next generation internet, May 1997. Report from the Workshop on Research Directions for NGI.

[20] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to groups. In *ACM Conference on Computer and Communication Security*, pages 31–37, March 1996.

[21] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A new approach to group key agreement. In *IEEE International Conference on Distributed Computing Systems*, May 1998.