New Multi-party Authentication Services and Key Agreement Protocols

Giuseppe Ateniese, Michael Steiner, Gene Tsudik

Abstract— **Many modern computing environments involve dynamic peer groups. Distributed simulation, multi-user games, conferencing applications and replicated servers are just a few examples. Given the openness of today's networks, communication among peers (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 encountered and discussed.**

Keywords— **Communication system security, Collaborative work, Dynamic group setting, Authentication, Key establishment/agreement protocols, Decision Diffie-Hellman problem.**

I. Introduction

THIS paper is concerned with security services in the context of *dynamic peer groups (DPGs)*. Such groups are common in many layers of the network protocol stack and many application areas of modern computing. Examples of DPGs include replicated servers (such as database, web, time), audio and video conferencing and, more generally, collaborative applications of all kinds. In contrast to large multicast groups, 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.) DPGs typically assume a many-to-many communication pattern rather than one-to-many commonly found in larger, hierarchical groups.

The specific security requirements and needs of dynamic peer groups – in particular, key management – are still considered as open research challenges [1]. Recently, several key agreement protocols geared for DPGs were proposed in [2]. They were obtained by extending the well-known Diffie-Hellman key exchange method [3] to groups of n parties. These protocols perform what we refer as *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 [4]. Both IKA and AKA protocols have been shown secure against passive adversaries. (The security is based on the polynomial indistinguishability of a Diffie-Hellman key from an arbitrary random value.)

This paper leverages the results of [2], [4] to develop practical and secure *authenticated* key agreement protocols for DPGs. Also considered are 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 peer 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 peer 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 as well as previous work on this field. After presenting some necessary terminology in Section IV we proceed (in Section V) 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 VI-A. Next, we consider *complete group key authentication* (bilateral among all group members) in Section VI-B and discuss key integrity and key confirmation features. The paper concludes with the discussion of other group secu-

Giuseppe Ateniese is currently the IBM Zurich Research Laboratory, 8803 Rüschlikon, Switzerland. E-mail: `gat@zurich.ibm.com`. Work done while visiting USC Information Sciences Institute.

Michael Steiner is currently at the Universität des Saarlandes, 66123 Saarbrücken, Germany. E-mail: `steiner@acm.org`. Work done while working for the IBM Zurich Research Laboratory, 8803 Rüschlikon, Switzerland.

Gene Tsudik is with the USC Information Sciences Institute, Marina del Rey, CA 90292, USA. E-mail: `gts@isi.edu`. Research supported by the Defense Advanced Research Project Agency, Information Technology Office (DARPA-ITO), under contract DABT63-97-C-0031.

Appeared in the IEEE Journal of Selected Areas in Communications, Vol 18, No. 4, April 2000. A preliminary version of this paper was presented at the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, November, 1998.

Manuscript received February 7, 1999; revised August 30, 1999. A preliminary version of this paper was presented at the 5th ACM Conference on Computer and Communications Security, San Francisco, CA, November, 1998.

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

rity services that are contingent upon authenticated key agreement.

II. Key Establishment Protocols

Key establishment protocols can be roughly classified in two categories: *key agreement* protocols [4] and (centralized) *key 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 peer group composed of members in different, and perhaps competing, organizations or countries.)
- Some DPG environments (such as *ad hoc* wireless networks) are highly dynamic and no group member can be assumed to be present all the time. However, most key distribution protocols assume fixed centers.
- It might be simply unacceptable for a single party to generate the group key. For example, each 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. IV.7 below) and resistance to known-key attacks (Def. IV.8 below) 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.

III. Previous Work

We now present several prior results in multi-party key agreement protocols. None of them provide any key authentication services and all are resistant only against passive attacks. Moreover, it is unclear how to add authentication services to these schemes.

One mathematically elegant proposal was proposed by Fiat et al. in [5]. A trusted center T selects a RSA-like modulus $n = pq$ and a *secret* element $\alpha \in \mathbb{Z}_n^*$ of large multiplicative order (such that it is hard to compute discrete logarithms). For any $1 \leq i \leq t$, each party M_i receives α^{x_i} mod n from T, with x_i random and relatively prime with x_j for $j \neq i$. In order to establish a secret group key S, each party M_i broadcasts x_i and then, after collecting all messages, computes $S = (\alpha^{x_i})^{x_1 \cdots x_{i-1} x_{i+1} \cdots x_t} \pmod{n}$ (i.e. a Diffie-Hellman key with all the contributions). Drawbacks of this protocol are that 1) it requires a trusted third party (T) and, 2) as shown in [5], two or more parties can collude and recover the secret α.

Another interesting scheme was presented in [6]. Given a $g \in Zp^*$ with p prime, each party M_i computes and broadcasts $y_i = g^{x_i} \pmod{p}$, where x_i is a randomly chosen secret. After receiving all the contributions, M_1 computes the group key:

$$S = g^{x_t g^{x_{t-1} g^{\cdots x_3 g^{x_2 x_1}}}}.$$

Note that to come up with the same group key, the other protocol parties needs to behave differently since the order of exponentiation is right to left, i.e., the protocol is asymmetric. In particular, for $j = 3, \ldots, t-1$, M_j sends $v_j = g^{v_{j-1}^{x_j}}$ to M_{j+1} (where $v_2 = y_1$) and then computes S. The party M_t simply waits v_{t-1} from M_{t-1} and then computes $S = v_{t-1}^{x_t}$.

One notable recent result is due to Burmester and Desmedt [7]. They construct a very efficient protocol (BD) which executes in only three rounds:

1. Each M_i generates its random exponent x_i and broadcasts $z_i = \alpha^{x_i}$.
2. Each M_i computes and broadcasts $W_i = (z_{i+1}/z_{i-1})^{x_i}$
3. Each M_i can now compute the key $S = z_{i-1}^{tx_i} \cdot W_i^{t-1} \cdot W_{i+1}^{t-2} \cdots W_{i-2} \bmod p$

The resulting key is $S = \alpha^{x_1x_2+x_2x_3+\cdots+x_tx_1}$. The protocol is proven secure provided the DH problem is intractable. However, there are some important assumptions underlying this protocol. Specifically, it requires each M_i to broadcast to the rest of the group and to receive $t-1$ messages in a single round. Moreover, the system has to handle t simultaneous broadcasts (in one round).

As mentioned in Section I, Steiner et al. [4] introduced a class of protocols, called *generic* Group Diffie-Hellman (GDH) key agreement. This entire class has been proven resistant against passive attacks. In brief, [4] shows that *if a 2-party DH key is indistinguishable from a random value then a t-party DH key is also indistinguishable from a random value.* Later in this paper, we describe in more detail some of the GDH protocols and extend them to provide authentication services.

IV. Goals, Definitions and Notation

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:

- Key Authentication
- Perfect Forward Secrecy (PFS)
- Resistance to Known-Key Attacks
- 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. [8])

Definition IV.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 IV.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 [8] protocol is not contributory as only one of the parties contributes a fresh exponent.

Definition IV.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 IV.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 IV.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 IV.6: An **authenticated group key agreement protocol** is a key agreement protocol which provides implicit key authentication.

Definition IV.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 IV.8: A protocol is said to be vulnerable to **known-key attack** if compromise of session keys allows: 1) a passive adversary to compromise keys of other sessions, or 2) an active adversary to impersonate one of the protocol parties. (See [9], [10], for details.)

The notation as used throughout the paper is shown in Figure 1.

All arithmetic throughout the paper is performed in a 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 $k \in \mathbb{N}$ (e.g. $k = 2$).

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 [11]; it also results in increased efficiency.

n	number of protocol parties (group members)
i, j	indices of group members
M_i	i-th group member; $i \in [1, n]$
p, q	p, q prime, $q \mid \phi p$
G	unique subgroup of $\mathbb{Z}_p^*$ of order q
α	exponentiation base; generator in group G
x_i	long-term secret key of M_i
r_i	M_i's secret exponent $\in_{\mathcal{R}} \mathbb{Z}_q$
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
$F()$	a function mapping elements from G to $\mathbb{Z}_q$

Fig. 1
Notation

When operating in subgroups it is important to take into account the attacks outlined in [11], [12]. 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.[2]

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 yet 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 $m = \alpha^c \pmod p$, 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.

V. Authenticated 2-party Key Agreement

In this section we develop an extension to the Diffie-Hellman (DH) [3] key agreement method that provides key authentication. We explicitly avoid requiring any cryptographic tools (e.g., symmetric encryption or signatures) other than those necessary for plain DH key agreement.

Before turning to the actual protocol, it is important to emphasize that there 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

[2]Verifying the order of an element x by checking, for example, that $x^{(p-1)/q} \pmod p \neq 1$, 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 $x^2 \neq 1 \pmod q$. Although this seems to be sufficient, the security of this method needs further study [12].

in the MTI protocol family [14].) 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.

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 $(x_1, \alpha^{x_1} \pmod p)$ and $(x_2, \alpha^{x_2} \pmod p)$ 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:

1. M_1 selects $r_1 \in_R \mathbb{Z}_q^*$.
2. $M_1 \longrightarrow M_2 : \alpha^{r_1} \pmod p$

Round 2:

1. M_2 selects $r_2 \in_R \mathbb{Z}_q^*$, and computes $K = F(\alpha^{x_1 x_2} \pmod p)$.
2. $M_2 \longrightarrow M_1 : \alpha^{r_2 K} \pmod p$

When M_1 receives $J = \alpha^{r_2 K} \pmod p$, computes $K^{-1} \pmod q$ and then $J^{r_1 K^{-1}} \pmod p$. The shared secret key is $S_2 = \alpha^{r_1 r_2} \pmod p$. Function $F()$ maps elements from G to $\mathbb{Z}_q$. If p is a safe prime ($p = 2q+1$) then a perfect mapping function would be $F(x)= x$ if $(x < q)$, $q - x$ if $(x >= q)$.

Fig. 2
Authenticated Diffie-Hellman (A-DH)

One protocol that satisfies the above criteria is A-DH, shown in Figure 2. It provides implicit key authentication as stated by the following theorem.

Theorem V.1: The A-DH protocol is a contributory authenticated key agreement protocol.

Proof: From the construction of the resultant session key $S_2 = \alpha^{r_1 r_2}$ it is evident that A-DH is *contributory.* In our model we assume that the two parties M_1 and M_2, wishing to share a secret, behave correctly. Moreover, we assume that any value computed as session key (S_2), by each party, is kept secret, i.e. it is infeasible for an attacker to obtain any (even partial) information about GK_2. Let C be an active adversary able to modify, delay, or inject messages. C's goal is to obtain the secret shared by M_1 and M_2, if any. There are four possible attack scenarios which we treat separately below. To prove our claim we use the following approach: if the attacker C were able to use a nondeterministic Turing machine TM to obtain information about the secret key, then the same TM could be used to solve an instance of the Diffie-Hellman Problem (DHP). The notation $TM(x_1, \ldots, x_n) = y$ means that, given input $x_1, \ldots, x_n$, TM outputs y.

1. **A-DH ends correctly (*passive attack*).** Suppose there exists a nondeterministic Turing machine TM_1 by which the attacker C is able to find the secret session key, i.e. $TM_1(\alpha^{r_1}, \alpha^{r_2 K}) = \alpha^{r_1 r_2}$. Since r_1 and r_2 are randomly chosen, it is easy to see that TM_1 could be used to solve a generic instance of DHP. In fact, given α^a and α^b, with unknown a, b, we would have $TM_1(\alpha^a, \alpha^{bK}) = \alpha^{ab}$.
2. **C substitutes α^{r_1} with α^x.** For simplicity's sake, we can assume that the attacker C knows the value x. The secret computed by M_1 is $\alpha^{r_1 r_2}$. The problem of computing this value has been treated in the previous case.
The secret computed by M_2 is $\alpha^{x r_2}$. The easiest way to compute this value is extracting α^{r_2} from $\alpha^{r_2 K}$. But this is as hard as DH problem if K is the DH value of the public-keys of M_1 and M_2.
3. **C substitutes α^{r_2} with α^y.** Suppose the attacker C knows the value y. The secret computed by M_2 is $\alpha^{r_1 r_2}$. The hardness of computing this value has been already treated in previous cases. C can compute the M_1's secret, $\alpha^{y r_1 K^{-1}}$, trying to get the value $\alpha^{r_1 K^{-1}}$. However, computing $\alpha^{r_1 K^{-1}}$ is as hard as DHP if K is the DH value of the public-keys of M_1 and M_2.
4. **C substitutes α^{r_1}, α^{r_2} with α^x and α^y, respectively.** Since the A-DH protocol's messages are each other independent, this case is analogous to the cases 2 and 3.

qed

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 V.2: The A-DH protocol provides perfect forward secrecy (PFS).

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

Theorem V.3: The A-DH protocol is resistant to known-key attacks.

Proof: The scenario is the following: the attacker C is able to modify both the A-DH protocol's messages and then get the values computed by each party M_1 and M_2 as secret key. From these values, C tries to compute information by which he can impersonate one of the protocol parties. For the sake of simplicity, we can assume that the target of the attacker C is finding α^K by which he can impersonate M_2.

There are four cases to consider:

1. C knows α^{r_1}, $\alpha^{r_2 K}$ and $\alpha^{r_1 r_2}$ (passive known-key attack).
2. C substitutes α^{r_1} with α^{x}, then knows α^{r_1}, $\alpha^{r_2 K}$, $\alpha^{r_1 r_2}$ and α^{r_2}.
3. C substitutes α^{r_2} with α^{y}, then knows α^{r_1}, $\alpha^{r_2 K}$, $\alpha^{r_1 r_2}$ and $\alpha^{r_1 K^{-1}}$.
4. C substitutes both α^{r_1}, α^{r_2} with α^{x} and α^{y}, respectively. Then C knows α^{r_1}, $\alpha^{r_2 K}$, α^{r_2} and $\alpha^{r_1 K^{-1}}$.

Therefore, the general problem that C has to solve is: given α^{a} and α^{ab} getting α^{b} where a, b are unknown random values[3]. It easy to see that, when working in subgroup of prime order, this problem is equivalent to DH problem. □

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".

VI. Authenticated Group Key Agreement

In [2], 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 [2], [15]. 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 V.

A. Authenticated GDH.2 protocol

Two practical protocols: GDH.2 and GDH.3 are defined in [2]. (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.

We begin with a brief overview of GDH.2 in Figure 3. This basic protocol can be easily amended to provide implicit key authentication in an efficient manner. This variation (A-GDH.2, shown in Figure 4) 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 $K_{in} = F(\alpha^{x_i \cdot\, x_n} \pmod p)$ with $i \in [1, n-1]$. Where x_i is a secret long term exponent selected by every M_i ($1 \le x_i \le q-1$) and $\alpha^{x_i} \pmod p$ is the corresponding long-term public key of M_i.

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.

[3]For example, if this problem were easy, the attacker C could get α^{K} from the values α^{r_2}, $\alpha^{r_2 K}$.

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 $r_i \in_R \mathbb{Z}_q^*$.
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 $r_n \in_R \mathbb{Z}_q^*$.
2. $M_n \longrightarrow$ ALL M_i: $\{\alpha^{\frac{r_1 \cdots r_n}{r_i}} | i \in [1, n[\}$

Fig. 3
Group Diffie-Hellman (GDH.2)

Protocol A-GDH.2:

Rounds 1 to $n-1$:
identical to GDH.2

Round n:
1. M_n selects $r_n \in_R \mathbb{Z}_q^*$
2. $M_n \longrightarrow$ ALL $\mathrm{M_i}$: $\{\alpha^{\frac{r_1 \cdots r_n}{r_i} \cdot K_{in}} | i \in [1, n[\}$.

Upon receipt of the above, every M_i computes: $\alpha^{(\frac{r_1 \cdots r_n}{r_i} \cdot K_{in}) \cdot K_{in}^{-1} \cdot r_i} = \alpha^{r_1 \cdots r_n} = S_n$.

Fig. 4
Authenticated Group Diffie-Hellman (A-GDH.2)

Theorem VI.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 V.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 VI.2: The A-GDH.2 protocol provides perfect forward secrecy.

Proof: 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 $V = \{\alpha^{\Pi(\mathcal{S})} | \ \mathcal{S} \subset \{r_1, \ldots, r_n\}\}$. But, as shown in [2], given V, it is intractable to find information on the group key $S_n = \alpha^{r_1, \ldots r_n}$, if the DDH problem in prime-order subgroup is hard. □

A.1 Resistance to known-key attacks.

A-GDH.2 is resistant to *passive* known-key attacks since 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 $\{\alpha^{\Pi(\mathcal{S})} | \ \mathcal{S} \subset \{r_1, \ldots, r_n\}\}$. 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.[4] 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):

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

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

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 [10], 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 [16].

B. Complete (Strong) 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 IV.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 or otherwise) exhibiting any other form of anomalous behavior.

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

Definition VI.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 $(0 < i \neq j \leq n)$.

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 VI.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:

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

Remark VI.5: 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 $P_i = (K_{1i} \cdots K_{ni})$

Fig. 5
Group Diffie-Hellman with Complete Key Authentication (SA-GDH.2)

A-GDH.2 can be augmented to provide complete group key authentication as shown in Figure 5. (To better illustrate SA-GDH.2 and its differences with respect to A-GDH.2, a 4-party example is shown in Figure 6.)

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.
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 VI.6: 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 VI.7: An interesting feature of SA-GDH.2 is its resistance to known-key attacks. Although we do not treat this topic in detail, it can be easily observed that an attack of the sort described in Section VI-A cannot succeed against SA-GDH.2.

C. Efficiency Summary

We now consider the costs incurred by the protocols described above. The two Tables I and II summarize, respectively, the communication and computation overhead of the following:
- GDH.2 – plain group key agreement [2].
- A-GDH.2 - authenticated group key agreement as specified in Section VI-A. 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 [17], [8] 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.

VII. New Services in Group Setting

As mentioned in the introduction, *key confirmation* (Def. IV.4 and [8]) is an important feature in key agreement protocols. Its purpose is to convince one or more parties that its peer (or a group of peers) 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 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 VI-A.) 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 VII.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 VII.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 VII.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.

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 VII-A, an adversary can still inject some input into the group key.

A. The Elusive Key Integrity

Key integrity (Def. IV.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

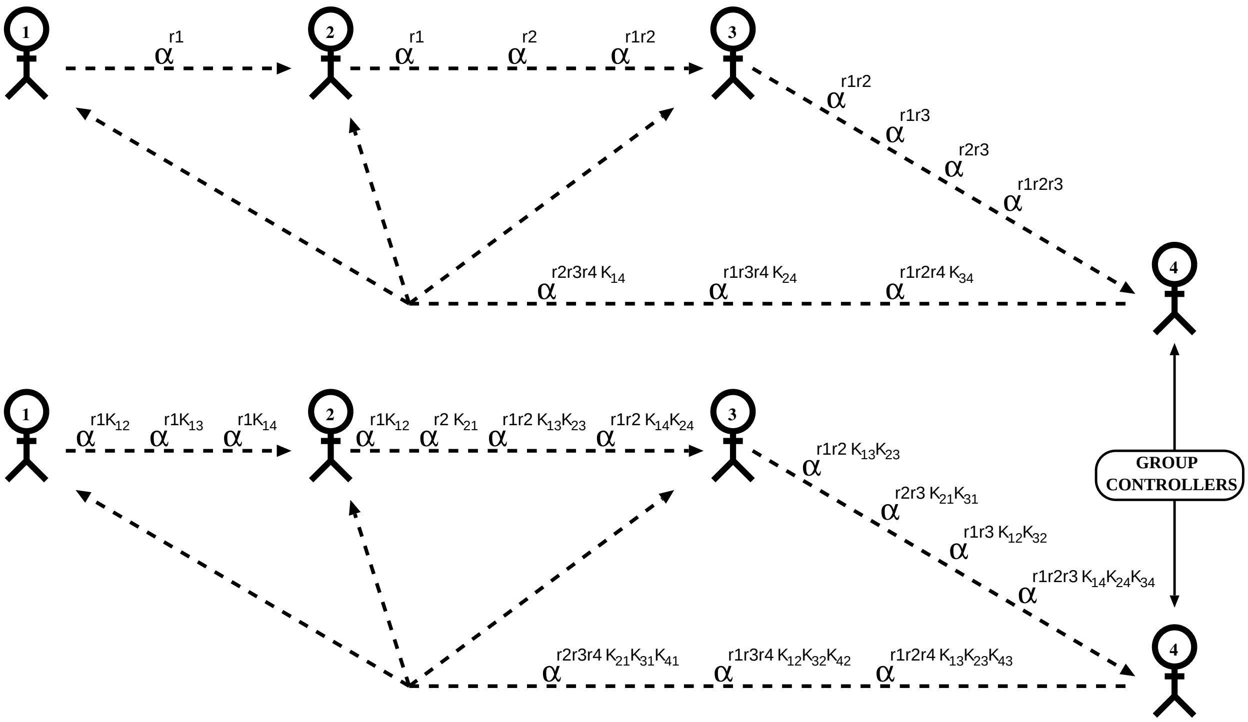


Fig. 6
An example/comparison of A-GDH and SA-GDH.2

TABLE I
Communication costs of 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

TABLE II
Computation costs of 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		$2n-2$	$2n-2$	$2n^2-2n$

time not guaranteeing key integrity. Consider the (3-party) SA-GDH.2 example shown in Figure 7.

Round 1:
M_1 selects $r_1 \in_R \mathbb{Z}_q^*$.
$M_1 \rightarrow M_2 : \alpha^{r_1 \cdot K_{12}}, \alpha^{r_1 \cdot K_{13}}$

Round 2:
M_2 selects $r_2 \in_R \mathbb{Z}_q^*$.
$M_2 \rightarrow M_3 : \alpha^{r_1 \cdot K_{12}}, \alpha^{r_1 \cdot K_{13} \cdot r_2 \cdot K_{23}}, \alpha^{r_2 \cdot K_{21}}$

Round 3:
M_3 selects $r_3 \in_R \mathbb{Z}_q^*$, computes group key $S_3(3)$ and broadcasts:
$M_3 \rightarrow M_1, M_2 : \alpha^{r_1 \cdot K_{12} \cdot K_{32} \cdot r_3}, \alpha^{r_2 \cdot K_{21} \cdot K_{31} \cdot r_3}, \alpha^{F(S_3(3))}$
M_1 and M_2 compute $S_3(1)$ and $S_3(2)$, respectively and confirm the correctness of their respective keys against $\alpha^{F(S_3(3))}$.

Fig. 7
Example run of Protocol SA-GDH.2

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.

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.

B. Dynamic Group Changes

Thus far, we treated groups as static entities. However, more often than not, group changes during the lifetime of a group. New members join in and old members leave (or get evicted from) the group. Moreover, due to "environmental" factors such as network failures groups can be partitioned. Similarly, when network partitions heal, multiple groups need to merge into one.

Therefore, the problem is how to share a secret key (achieve key agreement) in the face of membership changes. As stated in the introduction we require *key independence* and therefore need to compute a new (authenticated and contributory) key. Intuitively, we can approach the problem in two ways:

- **Starting from scratch:** all group members start the key agreement protocol anew and destroy any and all old values.
- **Using previous information:** in order to save computations, recycle old information to the extent possible (and secure) to share a common secret.

It seems clear that the first approach is expensive, unscalable and utterly unsuitable for environments with frequent membership changes.

In [4] we showed how the underlying non-authenticated Group Diffie-Hellman key agreement protocols can be extended to achieve very efficient and flexible dynamic operation, i.e. adding/deleting a single member, fusion/fission of sub-groups and re-keying.

Protocol GDH.2-MA:
Let M_{n+1} be the member to be added and M_n the group controller.

Round 1:
1. M_n selects $\widehat{r_n} \in_R \mathbb{Z}_q^*$
2. $M_n \longrightarrow M_{n+1}$: $\{\alpha^{\widehat{r_n}\frac{r_1 \cdots r_n}{r_i}} | i \in [1, n]\}$, $\alpha^{\widehat{r_n} r_1 \cdots r_n}$

Round 2:
1. M_{n+1} selects $r_{n+1} \in_R \mathbb{Z}_q^*$.
2. $M_{n+1} \longrightarrow M_i$: $\{\alpha^{\widehat{r_n}\frac{r_1 \cdots r_{n+1}}{r_i}} | i \in [1, n+1]\}$

Upon receipt of above, every M_i computes the new key as usual and stores last flow for future AKA protocol runs. Additionally, M_n replaces his key contribution r_n with $\widehat{r_n} r_n$.
Remark VII.4: Note that any group member can take that role; the decision who is group controller is simply a matter of policy.

Fig. 8
Member Addition Protocol for GDH.2

As an example you will find in Figure 8 the (unauthenticated) member addition protocol. The trick in that protocol (as well as in all the other AKA protocols) is the caching by the group members of the keying information (i.e., partial keys) distributed in the broadcast phase of the most recent IKA/AKA protocol run. This information is incrementally updated for an AKA protocol run and redistributed to the new incarnation of the group.

Figure 9 describes the authenticated version of the member addition protocol. The first flow is exactly the same

Protocol A-GDH.2-MA:

Round 1:

1. M_n selects $\widehat{r_n} \in_R \mathbb{Z}_q^*$.
2. $M_n \longrightarrow M_{n+1}$: $\{\alpha^{\widehat{r_n}\frac{r_1\cdots r_n}{r_i}\cdot K_{in}} | i \in [1,n]\}$, $\alpha^{\widehat{r_n} r_1 \cdots r_n}$

Round 2:

1. M_{n+1} selects $r_{n+1} \in_R \mathbb{Z}_q^*$.
2. $M_{n+1} \longrightarrow M_i$: $\{\alpha^{\widehat{r_n}\frac{r_1\cdots r_{n+1}}{r_i}\cdot K_{in} K_{in+1}} | i \in [1,n+1]\}$,

Upon receipt of above, every M_i computes the new key as $\alpha^{(\widehat{r_n}\frac{r_1\cdots r_n}{r_i}\cdot K_{in}K_{in+1})\cdot K_{in}^{-1}K_{in+1}^{-1}\cdot r_i} = \alpha^{\widehat{r_n} r_1\cdots r_{n+1}} = S_{n+1}$. and stores last flow for future AKA protocol runs. Additionally, M_n replaces his key contribution r_n with $\widehat{r_n} r_n$.

Fig. 9
MEMBER ADDITION PROTOCOL FOR A-GDH.2

as in the unauthenticated case except that we use now the broadcast flow of the authenticated protocols which also contain long-term keys K_{in} in the exponent. The Round 2 is essentially the same as the second round of the normal A-GDH.2 protocol and guarantees implicit key authentication. As defined in Figure 9 each member would have to remember the sequence of AKA protocols and the corresponding group controllers to be able to factor out the long-term keys K_{jk}. However, if M_i substitutes his key contribution r_i with $r_i \cdot K_{in}^{-1}$ after each broadcast flow, there is no need to save this history-information and computing the key is exactly the same operation as for IKA protocols.

The extension from unauthenticated to authenticated AKA protocols can be performed in the same straightforward manner also for all other AKA protocol described in [4] and will be omitted in the sequel.

C. 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

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.

VIII. Group Key Agreement and Byzantine Agreement

Group key agreement (GKA), in general, has similarities to the well-known *Byzantine Agreement (BA)* problem ([18]) 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[5] and that it is both fresh and random. Therefore, we claim that BA alone is not enough to build a robust GKA protocol.[6]

On the other hand, GKA has similarities with *secure multiparty computation (SMPC)* [19], [20]. In fact, GKA can be viewed as a special case of SMPC. However, we note that general SMPC techniques typically yield highly inefficient protocols.

IX. Current Status

This paper presented new definitions and protocols geared for the dynamic peer group (DPG) settings. In par-

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

[6] 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.

ticular, it showed how important security services (key authentication, key confirmation and entity authentication) can be incorporated into a particular class of group key agreement protocols.

We are developing a 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 [2]; protocols presented above are easily grafted onto GDH.3.) One of our central goals 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 collaborative (multi-user) simulations. As first tier of this process, we have developed a group key management API called *CLQ_API* [21]. *CLQ_API* implements the functions necessary to compute a group key in a distributed fashion. As it performs no communication on its own, *CLQ_API* requires a group communication layer in order to provide reliable and sequenced message delivery. This approach allows for a small, concise and generic API (it is composed of only seven function calls). Moreover, the purely communication-related issued (such as network partitions and other abnormalities) are taken care by the communication layer.

Finally, we are integrating *CLQ_API* with a reliable group communication layer: SPREAD [22]. Developed at Johns Hopkins University, SPREAD supports process group communication. It essentially provided a multicast communication layer that facilitates the development of fault-tolerant distributed applications in both LANs and WANs. Groups are conveniently identified by a name (ASCII string) selected by the user, such that messages are addressed to the entire group by specifying the group name. Using the group abstraction, the communication subsystem relieves the user from identifying the targets of messages explicitly, and from finding the network routes to them. In addition, it guarantees all-or-none delivery semantics, and handles message losses and transient network failures transparently.

In summary, the work is 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 (e.g., our integration efforts with the SPREAD system) will result in better understanding of group security needs and services.

X. Acknowledgements

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

References

[1] Jean E. Smith and Fred W. Weingarten, Eds., *Research Challenges for the Next Generation Internet.* Computing Research Association, May 1997, Report from the Workshop on Research Directions for the Next Generation Internet.

[2] Michael Steiner, Gene Tsudik, and Michael Waidner, "Diffie-hellman key distribution extended to groups," in *Third ACM Conference on Computer and Communications Security.* Mar. 1996, pp. 31–37, ACM Press.

[3] Whitfield Diffie and Martin Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[4] Michael Steiner, Gene Tsudik, and Michael Waidner, "CLIQUES: A new approach to group key agreement," in *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, Amsterdam, May 1998, pp. 380–387, IEEE Computer Society Press.

[5] Amos Fiat and Moni Naor, "Broadcast encryption," in *Advances in Cryptology – CRYPTO '93*, Douglas R. Stinson, Ed. 1993, vol. 773 of *Lecture Notes in Computer Science*, pp. 480–491, Springer-Verlag, Berlin Germany.

[6] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A secure audio teleconference system," in *Advances in Cryptology – CRYPTO '88*, S. Goldwasser, Ed., Santa Barbara, CA, USA, Aug. 1990, number 403 in Lecture Notes in Computer Science, pp. 520–528, Springer-Verlag, Berlin Germany.

[7] Mike Burmester and Yvo Desmedt, "A secure and efficient conference key distribution system," in *Advances in Cryptology – EUROCRYPT '94*, I.B. Damgard, Ed. 1994, Lecture Notes in Computer Science, Springer-Verlag, Berlin Germany, final version of proceedings.

[8] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press series on discrete mathematics and its applications. CRC Press, 1997, ISBN 0-8493-8523-7.

[9] Mike Burmester and Yvo Desmedt, "Towards practical 'proven secure' authenticated key distribution," in *1st ACM Conference on Computer and Communications Security*, Victoria Ashby, Ed., Fairfax, Virginia, Nov. 1993, ACM Press.

[10] Mike Burmester, "On the risk of opening distributed keys," in *Advances in Cryptology – CRYPTO '94.* 1994, Lecture Notes in Computer Science, pp. 308–317, Springer-Verlag, Berlin Germany.

[11] Ross Anderson and Serge Vaudenay, "Minding your p's and q's," in *Advances in Cryptology: Proceeding of Asiacrypt'96.* 1996, Springer-Verlag, Berlin Germany.

[12] Chae Hoon Lim and Pil Joong Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," in *Advances in Cryptology – CRYPTO '97*, Burton S. Kaliski Jr., Ed. Aug. 1997, number 1294 in Lecture Notes in Computer Science, pp. 249–263, Springer-Verlag, Berlin Germany.

[13] U. S. National Institute of Standards and Technology NIST, "The digital signature standard (DSS)," Dec. 1998.

[14] T. Matsumoto, Y. Takashima, and H. Imai, "On seeking smart public-key-distribution systems," *The Transactions of the IECE of Japan*, vol. E69, pp. 99–106, 1986.

[15] Claus Becker and Uta Wille, "Communication complexity of group key distribution," in *5th ACM Conference on Computer and Communications Security*, San Francisco, California, Nov. 1998, pp. 1–6, ACM Press.

[16] US National Institute of Standards Computer Systems Laboratory and Technology, "Secure hash standard (draft)," Federal Information Processing Standards Publication (FIPS PUB) 180-1, May 1994.

[17] Neal Koblitz, *A Course in Number Theory and Cryptography*, Number GTM 114 in Graduate Texts in Mathematics. Springer-Verlag, Berlin Germany, Berlin, 1987.

[18] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996.

[19] Ran Canetti, *Studies in Secure Multiparty Computation and Applications*, Ph.D. thesis, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Mar 1996, revised version.

[20] David Chaum, Claude Crepeau, and Ivan Damgard, "Multiparty unconditional secure protocols," in *Proceedings of the 20th Symposium on Theory of Computing (STOC)*, New York, 1988, pp. 11–19, ACM.

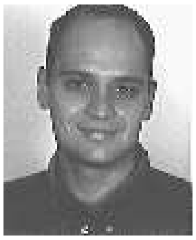
[21] Giuseppe Ateniese, Damian Hasse, Olivier Chevassut, Yongdae Kim, and Gene Tsudik, "The design of a group key agreement api," Research report, IBM Research, 1999.

[22] Y. Amir and J. Stanton, "The spread wide area group communication system," Technical Report CNDS 98-4, The Center for Networking and Distributed Systems, John Hopkins University, 1998.

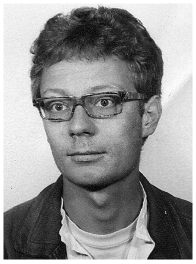
Contents

List of Figures

List of Tables



Giuseppe Ateniese received his Ph.D in Computer Science from the University of Genoa (Italy) in February 2000. From November 1997 to December 1998 he was at the Information Sciences Institute (University of Southern California) working on applied cryptography. From January 1999 to September 1999, he was at the IBM Zurich Research Laboratory working on network security. He joined the Department of Computer Science of the Johns Hopkins University in October 1999 as an Assistant Professor.



Michael Steiner is a research scientist at the Department of Computer Science, Universität des Saarlandes, Saarbrücken and in the network security research group at the IBM Zurich Research Laboratory. His interests include secure and reliable systems as well as cryptography. He received a Diplom in computer science from the Swiss Federal Institute of Technology (ETH) in 1992 and expects to receive a Ph.D. in computer science from the Universität des Saarlandes, Saarbrücken.

Gene Tsudik is a project leader at USC/ISI and a research associate professor in the Computer Science Department at USC. His research interests include network security, applied cryptography and routing in wireless networks. Dr. Tsudik received his Ph.D. in Computer Science from USC in 1991 and spent the next five years at IBM Research working on secure systems, protocols, mobile networks and electronic commerce. His current research at ISI is sponsored by DARPA, NSF and NSA. At USC, he teaches courses in Cryptography, Computer Security and Wireless Networks.