Diffie-Hellman Key Distribution Extended to Group Communication

Michael Steiner Gene Tsudik Michael Waidner
IBM Zürich Research Laboratory
CH-8803 Rüschlikon, Switzerland
{sti,gts,wmi}@zurich.ibm.com

Abstract

Ever since 2-party Diffie-Hellman key exchange was first proposed in 1976, there have been efforts to extend its simplicity and elegance to a group setting. Notable solutions have been proposed by Ingemarsson et al. (in 1982) and Burmester/Desmedt (in 1994). In this paper, we consider a class of protocols that we call *natural* extensions of Diffie-Hellman to the n-party case. After demonstrating the security of the entire class based on the intractability of the Diffie-Hellman problem we introduce two novel and practical protocols and compare them to the previous results. We argue that our protocols are optimal with respect to certain aspects of protocol complexity.

1 Introduction

It has been almost twenty years since Diffie-Hellman (DH) 2-party key exchange was first proposed in [1]. In the meantime, there have been many attempts to extend its elegance and simplicity to the group setting. The main motivating factor is the increasing popularity of various types of groupware applications and the need of doing it securely. Since key distribution is the cornerstone of secure group communication, it has naturally received a lot of attention. (See, for example: [2], [3], [4], [5], [6], [7], [8], [9].) Unfortunately some of the results are of only theoretical interest, while the security of some others remains unproven.

In this paper we consider a class of protocols that we call "natural" extensions of the 2-party Diffie-Hellman key exchange. We define a generic protocol of this class and prove its security; provided, of course, that the 2-party Diffie-Hellman decision problem is hard. This result allows us to craft a number of protocols without having to be concerned for their individual security. In particular, we present three new protocols, each optimal with respect to certain aspects of protocol efficiency.

This paper is organized as follows. We begin in section 2 by defining a *generic* group Diffie-Hellman protocol and proving its security. We then introduce three new group key distribution protocols in Sections 3.1-3.4 and discuss their relative merits and drawbacks. Next, in Section 4, we briefly review some notable previous results. The paper concludes with the summary/comparison of all current solutions and some directions for future work.

In proceedings of the 3rd ACM Conference on Computer and Communications Security, March 14-16 1996, New Delhi, India.

2 Generic n-Party Diffie-Hellman Key Distribution

2.1 Notation

The following notation is used throughout the paper:

n	number of participants in the protocol
i, j, k	indices of group members (ranging in $[1, n]$)
M_i	i-th group member; $i \in [1, n]$
q	order of the algebraic group
α	exponentiation base; generator in the algebraic group delimited by q
N_i	random exponent generated by group member M_i
$\mathcal{S}, \mathcal{T}$	subsets of $\{N_1, \ldots, N_n\}$
$\Pi(\mathcal{S})$	product of all elements in subset $\mathcal{S}$
K_n	group key shared among n members (we also use K when n is obvious)

2.2 Generic Protocol

We consider a family of protocols that we refer to as "natural" extensions of the original, 2-party Diffie-Hellman key exchange [1] to n parties.

Like in the 2-party case, all participants $M_1, \ldots, M_n$ agree *a priori* on a cyclic group, G, of order q, and a generator, α, of this group G. For each key exchange, each member, M_i, choses randomly a value $N_i \in G$.

In the 2-party case, M_i sends α^{N_i} to M_{3-i} and computes the common key $K = (\alpha^{N_{3-i}})^{N_i}$. For appropriately chosen G (see below) it is reasonable to assume that an adversary observing $(\alpha^{N_1}, \alpha^{N_2})$ cannot distinguish K from a random value $y \in G$.

All our protocols are based on distributively computing a subset of $\{\alpha^{\Pi(\mathcal{S})} \mid \mathcal{S} \subset \{N_1, \ldots, N_n\}\}$ From $\alpha^{N_1 \cdots N_{i-1} N_{i+1} \cdots N_n}$, member M_i can easily compute the shared key $K = \alpha^{N_1 \cdots N_n}$.

We call the protocol that reveals *all* these subsets the *generic n-party DH protocol.* Before presenting our protocols, we will prove that this generic protocol is secure. In this context, security means:

if a 2-party key is indistinguishable from a random value, the same is true for n-party keys.

Obviously, this will prove the security of all of our protocols at once.

2.3 Security of the Generic Protocol

Let k be a security parameter. All algorithms receive k as first input, implicitly, and will be polynomially bounded by k. even if the input itself is not bounded.

For concreteness, we consider a specific class of algebraic groups for which it is commonly assumed that the 2-party key is computationally indistinguishable from a random value: On input k, algorithm *gen* choses randomly a pair (q, α) such that q has length k bit, q and $q' = 2q + 1$ are both prime, and α generates the unique subgroup G of $\mathbb{Z}^*_{q'}$ of order q. Groups of this type are used, e.g., in [10] and [11]. The indistinguishability of the 2-party key is considered, e.g., in [12].

For $(q, \alpha) \leftarrow gen(k)$, $n \in \mathbb{N}$, and $X = (N_1, \dots, N_n)$ for $N_i \in \mathbb{Z}_q$, let

- $view(q, \alpha, n, X) :=$ the ordered set of all $\alpha^{N_{i_1} \cdots N_{i_m}}$ for all proper subsets $\{i_1, \dots, i_m\}$ of $\{1, \dots, n\}$,
- $K(q, \alpha, n, X) := \alpha^{N_1 \cdots N_n}$.

If (q, α) are obvious from the context, we omit them in $view()$ and $K()$. Note that $view(n, X)$ is exactly the view of an adversary in the generic n-party DH-protocol, where the final secret key is $K(n, X)$. Let the following two random variables be defined by generating $(q, \alpha) \leftarrow gen(k)$ and choosing X randomly from $(\mathbb{Z}_q)^n$:

- $A_n := (view(n, X), y)$, for a randomly chosen $y \in G$,
- $D_n := (view(n, X), K(n, X))$.

Let "$\approx_{\text{poly}}$" denote polynomial indistinguishability.

Theorem:
For each constant n, $A_2 \approx_{\text{poly}} D_2$ implies $A_n \approx_{\text{poly}} D_n$.

Proof (by induction on n):
Assume that $A_2 \approx_{\text{poly}} D_2$ and $A_{n-1} \approx_{\text{poly}} D_{n-1}$. Thus, we have to show $A_n \approx_{\text{poly}} D_n$. We do this by defining random variables B_n, C_n, and showing $A_n \approx_{\text{poly}} B_n \approx_{\text{poly}} C_n \approx_{\text{poly}} D_n$, which immediately yields: $A_n \approx_{\text{poly}} D_n$.

We can rewrite $view(n, (N_1, N_2, X))$ with $X = (N_3, \dots, N_n)$ as a permutation of:

$$\begin{array}{c}\big(\; view(n-1, (N_1, X)), K(n-1, (N_1, X)), \\ view(n-1, (N_2, X)), K(n-1, (N_2, X)), \\ view(n-1, (N_1 N_2, X)) \;\big)\end{array}$$

and $K(n, (N_1, N_2, X))$ as $K(n-1, (N_1 N_2, X))$.

We use this to redefine A_n and D_n. All in all, we consider the following four distributions. All of them are defined by $(q, \alpha) \leftarrow gen(k)$, choosing $c, N_1, N_2 \in \mathbb{Z}_q$ and $X \in (\mathbb{Z}_q)^{n-2}$ and $y \in G$ randomly.

- $A_n := (view(n-1, (N_1, X)), K(n-1, (N_1, X)),$ $view(n-1, (N_2, X))), K(n-1, (N_2, X)),$ $view(n-1, (N_1 N_2, X)), y)$
- $B_n := (view(n-1, (N_1, X)), K(n-1, (N_1, X)),$ $view(n-1, (N_2, X))), K(n-1, (N_2, X)),$ $view(n-1, (c, X)), y)$
- $C_n := (view(n-1, (N_1, X)), K(n-1, (N_1, X)),$ $view(n-1, (N_2, X))), K(n-1, (N_2, X)),$ $view(n-1, (c, X)), K(n-1, (c, X)))$
- $D_n := (view(n-1, (N_1, X)), K(n-1, (N_1, X)),$ $view(n-1, (N_2, X))), K(n-1, (N_2, X)),$ $view(n-1, (N_1 N_2, X)), K(n-1, (N_1 N_2, X)))$

Note that only the last two components vary.

$A_n \approx_{\text{poly}} B_n$ **follows from** $A_2 \approx_{\text{poly}} D_2$**:**
Assume that adv distinguishes A_n and B_n, and let (u, v, w) be an instance of $A_2 \approx_{\text{poly}} D_2$. We produce an instance for adv by using u for α^{N_1}, v for α^{N_2}, and w for $\alpha^{N_1 N_2}$ (or α^c), and choosing X and y randomly. If (u, v, w) belongs to A_2 (D_2), this new distribution belongs to A_n (D_n).

$B_n \approx_{\text{poly}} C_n$ **follows from** $A_{n-1} \approx_{\text{poly}} D_{n-1}$**:**
Assume that adv distinguishes B_n and C_n, and (ignoring a necessary permutation in order) let: $(view(n-1, (c, X)), y)$ be an instance for $A_{n-1} \approx_{\text{poly}} D_{n-1}$ (i.e., the problem is to decide whether $y = K(n-1, (c, X))$.) We produce an instance for adv by choosing N_1, N_2 randomly, and computing $(view(n-1, (N_i, X)), K(n-1, (N_i, X)))$ based on those values in $view(n-1, (c, X))$ that do not contain c as an exponent. The rest follows as in the last case.

$C_n \approx_{\text{poly}} D_n$ follows from $A_2 \approx_{\text{poly}} D_2$, almost exactly like the first statement. The only difference is that we do not choose y randomly, but as $K(n-1, (w, X))$.
□

3 Group Key Distribution Protocols

Having demonstrated the security of the generic protocol, we now turn to the specific examples drawn from the "natural" protocol family.

3.1 Group Key Distribution: GDH.1

The protocol (GDH.1) depicted in Figure 1 is quite simple and straight-forward. It consists of two stages: upflow and downflow. The purpose of the upflow stage is to collect contributions from all group members. As shown in the figure, M_i receives a collection of intermediate values. The task of each M_i on the upflow is to compute $\alpha^{N_1 \cdots N_i}$ by raising $\alpha^{N_1 \cdots N_{i-1}}$ – the highest numbered incoming intermediate value – to the power of N_i, append it to the incoming flow and forward all to M_{i+1}. For example, M_4 receives the set $\{\alpha^{N_1}, \alpha^{N_1 N_2}, \alpha^{N_1 N_2 N_3}\}$ and forwards to M_5: $\{\alpha^{N_1}, \alpha^{N_1 N_2}, \alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2 N_3 N_4}\}$.

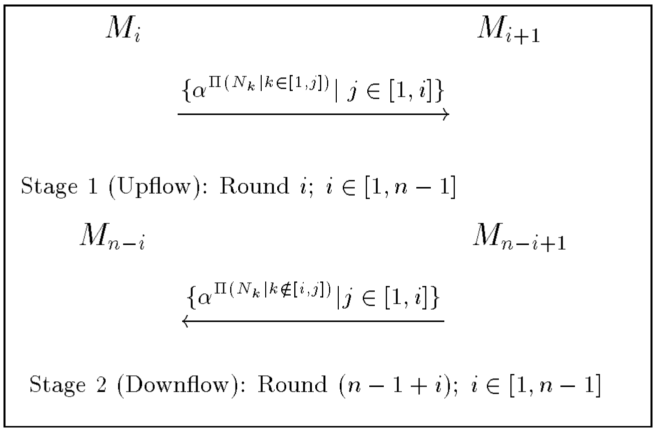

Figure 1: Group Key Distribution – GDH.1

To summarize the upflow stage, each group member performs one exponentiation and an upflow message between M_i and M_{i+1} contains i intermediate values.

The final transaction in the upflow stage takes place when the highest-numbered group member M_n receives the upflow message and computes $(\alpha^{N_1 \cdots N_{n-1}})^{N_n}$ which is the intended group key K_n.

After obtaining K_n, M_n initiates the downflow stage. In this final stage each M_i performs i exponentiations: one to compute K_n and $(i-1)$ to provide intermediate values to subsequent (lower-indexed) group members. For example, assuming $n=5$, M_4 receives a downflow message: $\{\alpha^{N_5}, \alpha^{N_1 N_5}, \alpha^{N_1 N_2 N_5}, \alpha^{N_1 N_2 N_3 N_5}\}$. First, it uses the last intermediate value in the set to compute K_n. Then, it raises all remaining values to the power of N_4 and forwards the resulting set: $\{\alpha^{N_5 N_4}, \alpha^{N_1 N_5 N_4}, \alpha^{N_1 N_2 N_5 N_4}\}$ to M_3. (In general, the size of a downflow message decreases on each link; a message between M_{i+1} and M_i includes i intermediate values.)

In summary, GDH.1 has following characteristics:

rounds	$2(n-1)$
messages	$2(n-1)$
combined message size	$(n-1)n$
exponentiations per M_i	$(i+1)$ for $i<n$, n for M_n
total exponentiations	$\frac{(n+3)n}{2}-1$

The main drawback of GDH.1 is its relatively large number of rounds. At the same time, GDH.1 imposes no special communication requirements, i.e., no broadcasting or synchronization is necessary.

3.2 Group Key Distribution: GDH.2

In order to reduce the number of rounds in GDH.1 we modify the protocol as shown in Figure 2. The upflow stage is still used to collect contributions from all group members; the only change is that each M_i now has to compose i intermediate values (each with $(i-1)$ exponents.) and one *cardinal* value containing i exponents. For example, M_4 receives a set: $\{\alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2}, \alpha^{N_1 N_3}, \alpha^{N_3 N_2}\}$ and outputs a set: $\{\alpha^{N_1 N_2 N_3 N_4}, \alpha^{N_1 N_2 N_3}, \alpha^{N_1 N_2 N_4}, \alpha^{N_1 N_3 N_4}, \alpha^{N_3 N_2 N_4}\}$

The *cardinal value* in this example is $\alpha^{N_1 N_2 N_3 N_4}$. By the time the upflow reaches M_n, the cardinal value becomes $\alpha^{N_1 \cdots N_{n-1}}$. M_n is thus the first group member to compute the key K_n. Also, as the final part of the upflow stage, M_n computes the last batch of intermediate values.

In the second stage M_n broadcasts the intermediate values to all group members.

GDH.2 has the following characteristics:[1]

rounds	n
messages	n
combined message size	$(n-1)(n/2+2)-1$
exponentiations per M_i	$(i+1)$ for $i<n$, n for M_n
total exponentiations	$\frac{(n+3)n}{2}-1$

In GDH.2, more so than in GDH.1, the highest-indexed group member M_n plays a special role by having to broadcast the last round of intermediate values. The main advantage of GDH.2 is due to its low number of protocol rounds; n as opposed to almost twice as many in GDH.1.

3.3 Practical Considerations

To summarize our discussion this far, GDH.1 and GDH.2 offer the following advantages:

1. No *a priori* ordering of group members
   Sequencing and numbering of M_i-s can take place in real time, as the protocol executes. Of course, the starting participant automatically becomes M_1.

2. No synchronization
   The protocol assumes asynchronous operation; no clock or round synchronization is necessary.

[1] Assuming atomic, one-message broadcast.

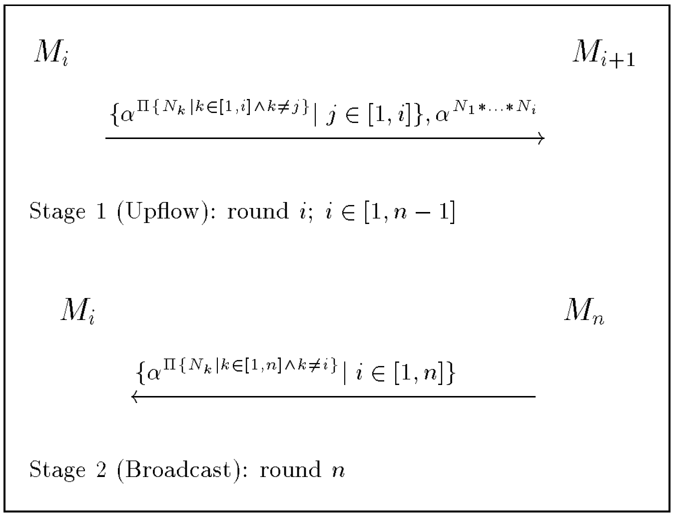

Figure 2: Group Key Distribution: GDH.2

3. Small number of exponentiations
   Number of exponentiations depends on the participant's index i. On the average, each M_i will perform $n/2$ exponentiations.

4. Minimal total number of messages (GDH.2)
   It is easy to see that at least n messages are required in any group key agreement protocol, i.e., each M_i has to contribute its own share of the key.

5. Minimal number of rounds for asynchronous operation (GDH.2)
   In order to construct a true DH key – $K_n = \alpha^{\Pi\{N_i | i\in[1..n]\}}$ – each participant needs to contribute its own exponent. Assuming that the protocol starts asynchronously (first round initiated by M_1), only one M_i can add its own exponent in a given round. Otherwise, either i) exponents have to be revealed, or ii) there has to be a way to construct K_n from $\alpha^{\Pi(\mathcal{S})}$ and $\alpha^{\Pi(\mathcal{T})}$ where $\mathcal{S}\cup\mathcal{T} = \{N_1,\ldots,N_n\}$. Both (i) and (ii) violate our basic premises.

6. Minimal number of messages sent/received by each participant (GDH.2)
   Proof outline: Assume that there exists a protocol that constructs a DH key and requires each M_i (other than M_1 or M_n) to send one and receive one message.

   One possibility is that M_i receives a message before sending one. In that case, from the message received, M_i must be able to construct K_n – since no further messages will be received. The message received must contain $\alpha^{\Pi(\{N_1,\ldots,N_n\}-N_i)}$, since there is no other way for M_i to construct K_n. This means that every M_j $(j\neq i)$ has already contributed its exponent N_j, and, hence, already received (except M_1) and subsequently sent, a message. Therefore, $M_i = M_n$ and $i=n$ since this can can only take place in $(n-1)$-st round.

   If we assume that M_i sends a message before receiving one then $M_i = Mi_1$ because the protocol runs asynchronously, i.e., only M_1 can start the protocol.

7. Security equivalent to 2-party Diffie-Hellman
   As shown in Section 2.3 above.

8. Implementation simplicity
   Just like 2-party Diffie-Hellman, GDH.1 and GDH.2 require only the modular exponentiation operation and the random number generator for the protocol execution. This means that, given a *black-box* realization of 2-party Diffie-Hellman, GDH.1/2 can be implemented thereupon without any additional arithmetic operations.

3.4 Group Key Distribution: GDH.3

In certain environments, it is desirable to minimize the amount of computation performed by each group member. This is particularly the case in very large groups. Since GDH.1/2 both require $(i+1)$ exponentiations from every M_i, the computational burden increases as the group size grows. The same, of course, is true for message sizes.

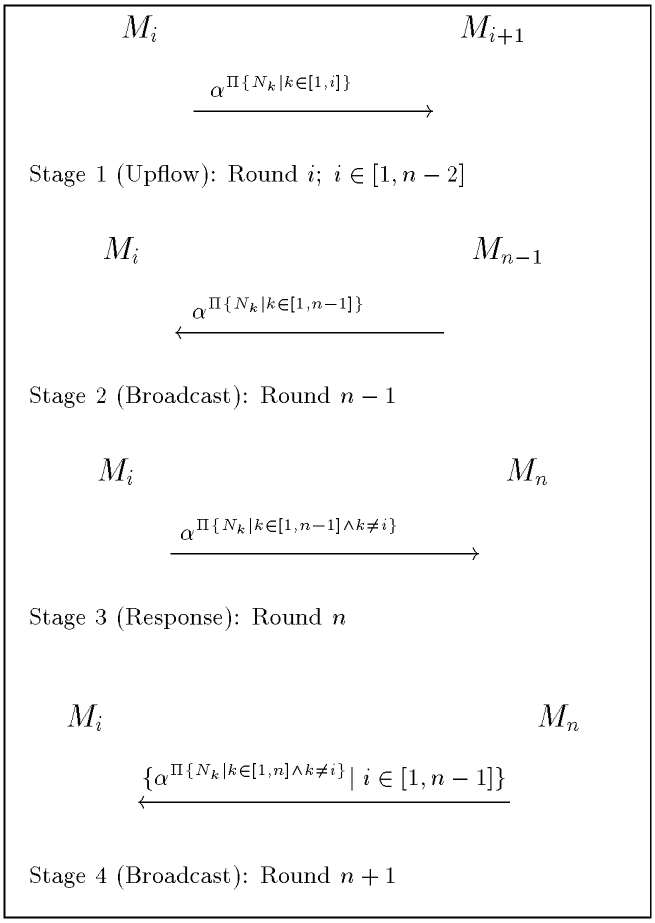

Figure 3: Group Key Distribution: GDH.3

In order to address these concerns we construct a protocol that is quite different from GDH.1/2 (see Figure 3.4.) The protocol consists of four stages. In the first stage we collect contributions from all group members similar to the upflow stage in GDH.1. After processing the upflow message M_{n-1} obtains $\alpha^{\Pi\{N_k|k\in[1,n-1]\}}$ and broadcasts this value in the second stage to all other participants. At this time, every M_i $(i \neq n)$ factors out its own exponent and forwards the result to M_n. (Note that factoring out N_i requires computing its inverse – N_i^{-1}. This is always possible if we choose the group q as a group of prime order). In the final stage, M_n collects all inputs from the previous stage, raises every one of them to the power of N_n and broadcasts the resulting $n-1$ values to the rest of the group. Every M_i now has a value of the form $\alpha^{\Pi\{N_k|k\in[1,n]\wedge k\neq i\}}$ and can easily generate the intended group key K_n.

This protocol – GDH.3 – has two appealing features:

- Constant message sizes
- Constant (and small) number of exponentiations for each M_i
  (except for M_n with n exponentiations required)

Its properties are summarized in the following table:

rounds	$n+1$
messages	$2n-1$
combined message size	$3(n-1)$
exponentiations per M_i	4 for $i < (n-1)$, 2 for M_{n-1}, n for M_n
total exponentiations	$5n-6$

3.5 Alteration of Group Membership

Thus far, we have assumed that the exact group membership is determined prior to the execution of our protocols. However, it is oftentimes necessary to either add a new, or delete an existing, group member after the initial group creation. Naturally, it is desirable to do so without having to re-run the entire protocol anew. To this end, we briefly sketch out below the member addition and member deletion protocols for GDH.2 and GDH.3. (GDH.1 does not lend itself to efficient construction of such protocols.) For a more general solution to the secure group membership see, e.g. [16].

3.5.1 Member Addition

The main security requirement of member addition is the secrecy of the previous group keys with respect to both outsiders and new group members.

In GDH.2 this can be achieved as follows:

1. We assume that M_n saves the contents of the Upflow message (Stage 1, round $n-1$ in Figure 3.4.)

2. M_n generates a new exponent $\hat{N}_n$ and computes a new upflow message (using $\hat{N}_n$, not N_n):
   $\{\alpha^{\Pi\{N_k|k\in[1,i]\wedge k\neq j\}} | \ j \in [1, n]\}, \alpha^{N_1 * \ldots * N_{n-1} * \hat{N}_n}$
   and sends it to the new member, M_{n+1}.

3. M_{n+1} generates its own exponent and computes the new key $K_{n+1} = \alpha^{N_1 * \ldots * \hat{N}_n * N_{n+1}}$

4. Finally, as in the normal protocol run, M_{n+1} computes n sub-keys of the form:
   $\{\alpha^{\Pi\{N_k|k\in[1,i]\wedge k\neq j\}} |; \jmath \in \ [1, n]\}$
   and broadcasts to the other group members.

Member addition in GDH.3 is almost identical to that in GDH.2. M_n has to save the contents of the original Broadcast and Response messages (Stage 2 and 3 in Figure 3.) M_n generates a new exponent and, with it, computes a new set of sub-keys which it forwards to the new member M_{n+1}. M_{n+1}

computes the new key K_{n+1} and adds its own exponent to each of the n sub-keys it received. Finally, M_{n+1} broadcasts the sub-keys as in Stage 4 of GDH.3 and all members compute K_{n+1}.

The extensions to GDH.2/3 are quite straight-forward and require only two additional rounds per each new member. The new key, K_{n+1} is easily computable by all parties and retains the same secrecy properties as K_n. However, while all other group members compute K_{n+1} with a single exponentiation, M_n is required to perform n exponentiations in addition to generating a new exponent. This extra burden on M_n may be undesirable.

3.5.2 Member Deletion

The main security requirement of member deletion is the secrecy of the subsequent (future) group keys with respect to both outsiders and former group members.

Protocol extensions for member deletion in both GDH.2 and GDH.3 are very similar to those for member addition.

Let M_p be the member slated for removal from the group. We assume, for the moment, that $p \in [1, n-1]$, i.e., $p \neq n$. M_n, once again, plays a special role by generating a new exponent $\hat{N}_n$. This time, however, M_n computes a new set of $n-2$ sub-keys: $\{\alpha^{\Pi\{N_k | k \in [1,i] \wedge k \neq j\}} | \ j \in [1, n-1] \wedge k \neq p\}$ and broadcasts them to all group members. Note that, since $\alpha^{N_1 *, \ldots, N_{p-1}, N_{p+1}, \ldots, N_{n-1}, \hat{N}_n}$ is missing from the set of broadcasted sub-keys, the newly excluded M_p is unable to compute the new group key.

In the event that M_n is to be removed from the group, M_{n-1} assumes the special role as described above.

4 Related Work

In this section we briefly review some notable previous work in DH-like protocols. A detailed and up-to-date discussion of this subject can be found in [7].

4.1 Ingemarsson et al.

The protocol depicted in Figure 4 is one of the family of protocols proposed by Ingemarsson et al. in [3]. (See also [13].) This protocol – hereafter referred to as ING – requires a synchronous startup and completes in $(n-1)$ rounds. The participants must be arranged in a logical ring. In a given round, every participant raises the previously-received intermediate key value to the power of its own exponent and forwards the result to the next participant. After $(n-1)$ rounds everyone computes the same key K_n.

$$M_i \qquad\qquad M_{(i+1) mod\ n}$$

$$\xrightarrow{\alpha^{(\Pi\{N_j |\ j \in [(i-k) mod\ n, i]\})}}$$

Figure 4: ING Protocol: Round k; $k \in [1, n-1]$

The ING protocol has the following characteristics:

rounds	$(n-1)$
messages	$n(n-1)$
combined message size	$n(n-1)$
exponentiations per M_i	n
total exponentiations	n^2

We note that, since ING falls into the class of "natural" extensions of Diffie-Hellman 2-party protocol, the proof of security in Section 2.3 applies to it as well.

4.2 Burmester/Desmedt Protocol

Burmester and Desmedt present in [9] a much more efficient protocol. Their protocol is executed in only three rounds:

1. Each user M_i generates its random exponent N_i and broadcasts $z_i = \alpha^{N_i}$.
2. Every M_i computes and broadcasts $X_i = (z_{i+1}/z_{i-1})^{N_i}$
3. M_i can now compute[2] the key $K_n = z_{i-1}^{nN_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \bmod p$

The key defined by this scheme is different from the previous protocols, namely $K_n = \alpha^{N_1 N_2 + N_2 N_3 + \cdots + N_n N_1}$. Nevertheless the protocol is proven secure provided the Diffie-Hellman problem is intractable.

In summary, the BD protocol has the following characteristics:

rounds	2
messages	$2n$
combined message size	$2n$
exponentiations per M_i	$n+1$
total exponentiations	$(n+1)n$
divisions per M_i	1

An important advantage of the BD protocol is its "cheap" exponentiations. While the number of exponentiations per M_i is still $(n+1)$, in all but one the exponent is at most $(n-1)$. This makes for big savings in computation.

5 Comparison and Summary

All group key distribution protocols discussed above are summarized and compared in Figure 5.

As indicated in the previous section, BD (and BD*) is markedly superior to the others with respect to exponentiation operations since almost all operations involve relatively small exponents.

From Table 5 it is clear that, with respect to time (i.e., number of rounds), the BD protocol is well ahead of the rest. It requires only two rounds of *simultaneous* broadcasts as opposed to linear (in terms of n) number of rounds in the other protocols. However, the ability to perform n simultaneous broadcasts is not a feature available in most network environments. Even in a broadcast LAN environment, only one party can broadcast at any given time. Therefore, it may be worthwhile to compare the other protocols with BD* – a version of BD without the simultaneous broadcast feature. Since BD* would require $2n-1$ rounds, it does not compare with the rest as favorably as plain BD. (On the other hand, it has been noted[3] that extra rounds in BD* are due to nodes waiting for a chance to This is in contrast to GDH.1-3 where rounds are mostly triggered by message arrival. Thus,

[2] All indexes are modulo n.

[3] By one of the referees.

a broadcast round in BD* is *shorter* than a round in GDH.1-3.)

In the same vein, GDH.3 (in Stage 3, Figure 3) requires one round of $(n-1)$ simultaneous unicasts to M_n. We note that a more realistic GDH.3* would require $2n$ rounds. On the other hand, $(n-1)$ simultaneous unicasts in GDH.3 result in significantly less load as compared with n simultaneous broadcasts in BD.

In terms of communication bandwidth overhead, GDH.2 leads with only n messages. On the other hand, if we measure total bandwidth overhead (by tallying all message sizes), BD* comes out a clear winner with the least total information exchanged.

Another important measure of protocol efficiency is **the number of messages received and sent by each participant**. It is well-known that sending or receiving a message involves going through the entire protocol stack – a non-negligible task in terms of both time and resource consumption. Moreover, it is impossible in most (non-specialized) network architectures for a node to receive multiple messages simultaneously. This consideration is especially applicable to both BD and BD* protocols, i.e., regardless of whether all nodes can broadcast simultaneously, a given node cannot receive $(n-1)$ incoming messages all at once. Table 5 clearly illustrates that GDH.2 involves the least overhead with respect to the communication infrastructure: as part of the protocol each node sends a single message and receives only two (except M_1 and M_n which receive one message.)

Finally, we consider the issue of protocol symmetry. Both BD/BD* and ING offer symmetric operation.[5] This is partly due to their synchronous nature. (An asynchronous protocol can not be symmetric; someone has to initiate it.) All three GDH protocols are, to certain extent, asymmetric. GDH.1/2 are both *communication-asymmetric*. GDH.1 requires M_1 to initiate the upflow, and M_n – the downflow, stage. GDH.2 is similar in that it requires M_n to perform the final broadcast.

GDH.3 is not only communication- but also *computation-asymmetric*. The former is because M_1 and M_{n-1} are required to initiate stages 1 and 2, respectively. Computational asymmetry is due to the special role of M_n who has to perform computations different from those of other participants. (Note that M_n performs $n-1$ exponentiations in stage 4; however, it does not compute an inverse of N_n.)

6 Conclusions and Future Work

In conclusion, we have defined a class of "natural" extensions of Diffie-Hellman key exchange to the n-party setting and have shown that the security of a generic n-party protocol of this class is equivalent to the security of the original 2-party protocol. Armed with this general result, we introduced three concrete group key distribution protocols. We have shown that – in a realistic communication environment – our protocols are more efficient in some respects than previous results (or sometimes even optimal.)

There remain some items for future work. Our protocols do not provide authentication of the participants. It should be possible to augment them to provide authentication in a manner similar to that described in [9] or [14]. Another issue to address is protocol extensions for handling periodic re-keying. Finally, more formal (and convincing) arguments need to be developed to support optimality/minimality claims in Section 3.3.

[4] the version of Burmester/Desmedt protocol without simultaneous broadcast.

[5] In other words, all participants do the same thing.

Acknowledgements

The authors are grateful to the anonymous referees for their many helpful comments.

References

[1] Whit Diffie and Martin Hellman. New Directions In Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[2] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A Secure Audio Teleconference System. In S. Goldwasser, editor, *Advances in Cryptology – CRYPTO '88*, number 403 in Lecture Notes in Computer Science, pages 520–528, Santa Barbara, CA, USA, August 1990. Springer-Verlag, Berlin Germany.

[3] I. Ingemarsson, D. Tang, and C. Wong. A Conference Key Distribution System. *IEEE Transactions on Information Theory*, 28(5):714–720, September 1982.

[4] Hugh Harney, Carl Muckenhirn, and Thomas Rivers. Group Key Management Protocol (GKMP) Architecture. INTERNET-DRAFT, September 1994.

[5] Yi Mu, Yuliang Zheng, and Yan-Xia Lin. Quantum Conference Key Distribution Systems. Technical Report 94-6, University of Wollongong, NSW, Australia, 1994.

[6] Chin Chen Chang, Tzong Chen Wu, and C.P. Chen. The Design Of A Conference Key Distribution System. In *Advances in Cryptology – AUSCRYPT '92*, Lecture Notes in Computer Science, pages 467–474. Springer-Verlag, Berlin Germany, December 1992.

[7] Michael K.Just. Methods Of Multi-party Cryptographic Key Establishment. Master's thesis, Ottwa Carleton Institute for Computer Science, Caleton University, Ottawa, Ontario, August 1994.

[8] Tzonelih Hwang. Cryptosystem For Group-oriented Cryptography. In I.B. Damgard, editor, *Advances in Cryptology – EUROCRYPT '90*, number 473 in Lecture Notes in Computer Science, pages 352–360. Springer-Verlag, Berlin Germany, May 1991.

[9] M. Burmester and Y. Desmedt. A Secure And Efficient Conference Key Distribution System. In I.B. Damgard, editor, *Advances in Cryptology – EUROCRYPT '94*, Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 1994.

[10] C.P. Schnorr. Efficient Signature Generation By Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[11] The Digital Signature Standard Proposed By NIST. *CACM*, 35(7):36–40, July 1992.

[12] S. Brands. An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical Report CS-R9323, CWI, March 1993.

[13] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, Inc, 1994.

[14] M. Steiner, G. Tsudik, and M. Waidner. Refinement And Extension Of *Encrypted Key Exchange. ACM Operating Systems Review*, July 1995.

	ING	BD	BD*[4]	GDH.1	GDH.2	GDH.3
rounds	$n-1$	2	$2n-1$	$2(n-1)$	n	$n+1$
total messages	$n(n-1)$	$2n$	$2n-1$	$2(n-1)$	n	$2n-1$
combined msg size	$n(n-1)$	$2n$	$2n$	$n(n-1)$	$\frac{(n+3)n}{2}-3$	$3(n-1)$
messages sent per M_i	$n-1$	2	2	2 1 for M_1, M_n	1	2
messages received per M_i	$n-1$	$n+1$	$n+1$	2 1 for M_1, M_n	2 1 for M_1, M_n	3 n for M_n
exponentiations per M_i	n	$n+1$	$n+1$	$i+1$	$i+1$	4 2 for M_{n-1} $n-1$ for M_n
total exponentiations	n^2	$(n+1)n$	$(n+1)n$	$\frac{(n+3)n}{2}-1$	$\frac{(n+3)n}{2}-1$	$5n-6$
synchronization	Y	Y	Y	N	N	N
DH key	Y	N	N	Y	Y	Y
symmetry	Y	Y	Y	N	N	N

Figure 5: Protocol Comparison

[15] T. Matsumoto, Y. Takashima, H. Imai. A Method Of Generating Secret Data Common To All Members Of A Specified Group. *IECE Technical Report IT85-34*, September 1985.

[16] M. Reiter, A Secure Group Membership Protocol. *IEEE Symposium On Research in Security and Privacy*, May 1994.