Scalable Authenticated Tree Based Group Key Exchange for Ad-Hoc Groups

Yvo Desmedt[1]*, Tanja Lange[2], and Mike Burmester[3]

[1] Information Security, Department of Computer Science,
University College London, UK
`y.desmedt@cs.ucl.ac.uk`

[2] Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
`tanja@hyperelliptic.org`

[3] Department of Computer Science
Florida State University, USA
`burmester@cs.fsu.edu`

Abstract. Task-specific groups are often formed in an ad-hoc manner within big structures, like companies. Take the following typical scenario: A high rank manager decides that a task force group for some project needs to be built. This order is passed down the hierarchy where it finally reaches a manager who calls some employees to form a group. The members should communicate in a secure way and for efficiency reasons symmetric systems are the common choice. To establish joint secret keys for groups, group key exchange (GKE) protocols were developed. If the users are part of e.g. a Public Key Infrastructure (PKI), which is usually the case within a company or a small network, it is possible to achieve authenticated GKE by modifying the protocol and particularly by including signatures.

In this paper we recall a GKE due to Burmester and Desmedt which needs only $O(\log n)$ communication and computation complexity per user, rather than $O(n)$ as in the more well-known Burmester-Desmedt protocol, and runs in a constant number of rounds. To achieve authenticated GKE one can apply compilers, however, the existing ones would need $O(n)$ computation and communication thereby mitigating the advantages of the faster protocol. Our contribution is to extend an existing compiler so that it preserves the computation and communication complexity of the non-authenticated protocol. This is particularly important for tree based protocols.

Keywords. Key Distribution, Group Key Exchange, Tree based GKE, Ad-Hoc Groups, Forward Security, Authentication, Anonymity.

1 Introduction

Today several banks have branches worldwide. Moreover, stock owners of these banks often are in different countries. In general, globalization implies that decision makers are in different locations. So efficient and secure communication within a group is very important. Clearly symmetric systems offer the higher throughput and so all group members must hold a common *secret key*. Given the distributed nature of the group new keys can only be established through an insecure channel. Furthermore, the parties need to be assured that this key is shared with the correct group members, so the users must be authenticated. This shows that in financial cryptography group key exchange is an important primitive.

* Part of this research was done while visiting the ITSC Bochum 2004.

Another scenario is the need to set up ad-hoc groups. Organizations, such as financial ones, are usually organized in a hierarchical way. Often outside consultants are needed in virtual group meetings. The president of the organization does not know who these experts are. We find ourselves in a situation where members lower in the hierarchy decide who the outside-consultants are that should join the group. Again secure group communication is needed.

Once this symmetric key is established it can be used for basically all communication needs within the group. It not only works as the key in a symmetric system to ensure that data cannot be decrypted but also allows members within the group to communicate anonymously and the key can also be used in MACs to authenticate messages.

A simple solution for authenticated group key exchange is to have one user (the chair) choose the key and exchange it with the next one, who will exchange it with the next, etc., each by using the Diffie-Hellman KE protocol. The cost of this solution is that it requires $O(n)$ rounds and each user has constant communication and computation. We refer to this scenario as the *naive approach*.

So far, several group KE protocols have been proposed, most of which are extensions of the two-party Diffie-Hellman protocol [7, 8, 15, 17, 9]. Katz and Yung [19] designed a compiler that transforms any secure group KE protocol into an authenticated KE protocol. The model used is a refinement of models proposed by Bresson et al. in [6]. The additional requirements are: a PKI for digital signatures, and each message issued should be signed and checked by all participants. [4] To avoid replay attacks and loss of intermediate messages without noticing, each group member needs to maintain a counter for messages send during one KE and a signed string must include *all* group participants as well as a random nonce for *each* participant. As an example they consider the Burmester-Desmedt scheme [7, 9] which requires a constant number of rounds but has communication complexity[5] $O(n)$ and computational complexity of $O(n)$. Since, apparently, there is another Burmester-Desmedt group key exchange protocol [8], we refer to the one in [7, 9] as Burmester-Desmedt I (BD-I).

Although BD-I is considered algebraically nice, it is rarely used in real life application. As has been pointed out by many authors, e.g. [21, p.86] "One shortcoming of BD is the high communication overhead." It is for this reason that tree based group key exchange protocols are considered superior, as we describe further. Tree based group key exchange protocols have become quite popular due to the work of Wong-Gouda-Lam [23] (see also [24]). They actually considered a generalization of trees, i.e., directed acyclic graphs. One should observe that many schemes can be described as trees. Indeed the Boyd and Nieto [5] scheme corresponds to a star which is a special tree. Certain trees provided better efficiency. We now discuss such a scheme.

[4] We like to remark that the PKI could also be replaced by an Identity-based Key Infrastructure (IKI) [10].

[5] Although each sender only sends a constant number of strings (two), the number of received strings is $O(n)$. See Section 2.3 for more details.

The Burmester-Desmedt scheme II (BD-II) [8] is one of the schemes which uses a (binary) tree for group KE. Using the properties of trees, it can achieve *logarithmic complexity* while keeping the same *constant number of rounds.* (Note that the work in BD-II predates the one by Wong-Gouda-Lam.) Now if one were to use the Katz-Yung compiler [19] it would mitigate the advantage of BD-II as checking of $O(n)$ signatures would be required. In this contribution we give an *authenticated version of the BD-II group KE* protocol with an overall communication and computation complexity of $O(\log n)$ while running in a constant number of rounds. *These ideas extend to other group key exchange protocols, in particular tree based ones with logarithmic complexity.* For this reason we modify the Katz-Yung compiler.

After introducing the BD-II schemes and stating the security model we give a security proof for a passive adversary of one of the Burmester-Desmedt II schemes (although the paper contains several security claims, no proofs are given in [8]). We then extend the scheme to an authenticated KE by modifying the Katz-Yung compiler. The main problem with a direct application of the Katz-Yung compiler is that each party needs to verify $O(n)$ signatures which turns the overall complexity to $O(n)$ even though the underlying protocol allows for $O(\log n)$. The main observation which enables us to keep the logarithmic complexity is to notice that the joint key is computed by any user U entirely from information provided by its at most $\log n$ ancestors along the tree. Hence, only logarithmically many signatures need to be verified. We present the compiler in the most general setting applicable to any group KE protocol.

The BD-II group KE protocol is non-contributory in the sense that the key does not depend on the contribution of all members. In fact, no protocol with a computation or communication complexity lower than $O(n)$ can be fully contributory if it runs in a constant number of rounds and without delay. However, the Katz-Yung model allows for this since it does *not* deal with *active insiders.* We show that for a passive adversary the security of the protocols has a tight reduction to the DDH problem. Active external adversaries cannot insert messages in the name of a group member as the protocol is authenticated. There is no attack advantage of being a member of system, i.e. being registered to the system's PKI, if one is not part of the group which is performing the group key exchange. When we speak of insiders we mean actual members of the group which are agreeing on a key. For the BD-I protocol it has been shown in [17, 22, 12] that as few as 2 malicious insider are enough to force the resulting group key to be any element they want, e.g. one known to an outsider, without needing any extra communication during the execution of the GKE. Our presented scheme is no more secure against active insiders. More specifically, we shall use the same model as Katz-Yung [19] which does not deal with active malicious insiders that attempt to prevent an honest party from obtaining the common group key. The advantage of an active adversary is defined to be the advantage of obtaining the common group key and like Katz and Yung we assume the *"best-case" scenario* of *an adversary who delivers all messages intact to the appropriate recipient(s) as soon as they are sent.* ([19, Section 1.3]). Recent work by Katz and Sun Shin [18] formalizes insider attacks and defines the security of authenticated GKE protocols against malicious insiders in the framework of universal composablility. However, [12] demonstrate that dealing with active insiders is far from trivial. They criticize the model in [18] and present a different one which they consider to be more realistic. We do not want to enter into this controversy and exclude malicious insiders. Note that a full contributory variant of BD-II is easy to design, maintaining, order-wise, the same complexity but introducing delays. A detailed description will be given in the full paper.

We recently noticed an independent result on authenticated tree based GKEs [20]. That paper is also based on the Burmester-Desmedt II GKE even though this is not stated in the paper. One shortcoming of that paper is that they require each party to check all signatures on all messages; this implies that they have $O(n)$ computation costs rather than the desired $O(\log n)$. It is because of that difference that the Katz-Yung compiler [19] needs to be adjusted. Furthermore, their group identifiers have length $O(n)$.

The remainder of this paper is organized as follows. We start by stating the security model and surveying one of the BD-II schemes. We then prove its security against passive adversaries and provide an authenticated variant based on the DDH assumption. A comparison with other schemes shows the advantages of our proposal.

2 Models

Unless particularly mentioned we follow the same lines as Katz and Yung [19] who used the security model for group KE due to Bresson et al. [6]. We first introduce the notations and then briefly mention the oracles the adversary can query depending on the protocol. For full details we refer to [19].

2.1 Participants and initialization

There is a polynomial-size set $\mathcal{P}$ of potential participants in the group key exchange, any subset of $\mathcal{P}$ may decide to establish a session key. We assume that during an initialization phase each participant in $\mathcal{P}$ runs an algorithm $\mathcal{G}(1^k)$ to generate a pair of public and private keys (PK, SK). The secret key is stored by the user and (certified) public keys are accessible to all participants.

2.2 Adversarial model

We denote the instance i of user U as Π_U^i, each instance may be used only once. Each instance has associated with it the variables state_U^i, term_U^i, acc_U^i, used_U^i, pid_U^i and the session key sk_U^i which we now explain. In our model, the partner ID pid_U^i contains a group identifier gid which identifies all partners involved in the current execution of the GKE. The other definitions are as in [6], so state_U^i represents the current (internal) state of instance Π_U^i, term_U^i acc_U^i and used_U^i take boolean values indicating whether the instance has been terminated or accepted or used, respectively. Most of these variables appear only implicitly except for pid_U^i, sid_U^i, and sk_U^i.

The adversary is assumed to have full control over all communication in the network. His interaction is modeled by the following oracles:

– $\mathsf{Send}(U, i, M)$ – to send the message M to instance Π_U^i and output the reply generated by this instance. This oracle may also be used to initiate a key exchange among a group $\{U_1, \ldots, U_n\}$ of users identified by some group identifier gid. The length of gid must correspond to the security parameter. This means that the first round of the KE protocol is executed upon receipt of this message.
– $\mathsf{Execute}(\mathsf{gid})$ – to execute the protocol between unused instances of players $U_1, \ldots, U_n \in \mathcal{P}$ determined by the group identifier gid and to output the transaction of the execution. The adversary has control over the number of players and their identities.
– $\mathsf{Reveal}(U, i)$ – to reveal the session key sk_U^i of player U belonging to instance i.
– $\mathsf{Corrupt}(U)$ – to output the long-term secret key SK_U of player U.
– $\mathsf{Test}(U, i)$ – to be issued the final test. Once the adversary decides that he has enough data he queries the Test oracle for a challenge. A random bit b is generated; if $b = 1$ then the adversary is given sk_U^i, otherwise he receives a random session key.

A *passive adversary* is given access to the $\mathsf{Execute}$, Reveal, $\mathsf{Corrupt}$ and Test oracles, while an *active adversary* is additionally given access to the Send oracle. Both types of adversaries are allowed to make adaptive queries before and after the Test oracle is queried.

Partnering The session ID sid_U^i equals the concatenation of all messages sent and received by Π_U^i during the course of its execution. For the partner ID we deviate from the suggestion in [19] and *generalize their setting to the situation that not each party communicates with each of the others* during an execution of the protocol. Although this may seem as a slight adaptation of [19] (see also [6]), it allows us to dramatically improve on the efficiency of the schemes.

We assume that the group of users can be identified uniquely by a group identifier gid. This assumption holds true in all network protocols and fits well for hierarchical situations we encounter in the financial world. Then pid_U^i consists of the group identifier and the identities of the players in the group *with which Π_U^i interacts* during the KE protocol, i.e., *to which he sends messages or from which he receives messages.* Since the underlying unauthenticated protocol is assumed to provide each user with the same key this implies that the union of all pid_U^i covers each user involved in the key exchange. Furthermore, the graph displaying the communication must be connected. This is due to the fact that all users obtain the same key. In our compiler this ensures that each signature is checked by a group member which is connected via checked paths.

Correctness We require that for all U, U' and i, i' involved in the same key exchange and such that $\mathsf{acc}_U^i = \mathsf{acc}_{U'}^{i'} =$ TRUE, the same valid session key is established $\mathsf{sk}_U^i = \mathsf{sk}_{U'}^{i'} \neq$ NULL.

Security and freshness An instance Π_U^i is *fresh* unless one of the following is true: (1) at some point, the adversary queried $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', i')$ for any $\Pi_{U'}^{i'}$ in the same group (denoted by gid) as Π_U^i or (2) a query $\mathsf{Corrupt}(V)$ was asked before a query of the form $\mathsf{Send}(U, i, *)$ or $\mathsf{Send}(U', i', *)$, where V is in pid_U^i. (Note that our definition of pid_U^i only includes those users that U is directly interacting with, i.e., those providing input to the key computation of U).

The event Succ occurs if the attacker is successful, i.e., if he queries the Test oracle on a fresh instance Π_U^i for which acc_U^i = TRUE and guesses the bit b correctly. The advantage of attacker $\mathcal{A}$ against protocol P is defined as $\mathsf{Adv}_{\mathcal{A},P(k)} \overset{\text{def}}{=} |2 \cdot \mathsf{Pr}[\mathsf{Succ}] - 1|$.

Protocol P is a secure group KE protocol if it is secure against a passive adversary, i.e., for any PPT passive adversary $\mathcal{A}$ the advantage $\mathsf{Adv}_{\mathcal{A},P}(k)$ is negligible. Protocol P is a secure authenticated group KE (AKE) if it is secure against an active adversary.

We use $\mathsf{Adv}_P^{\mathsf{KE}}(t, q_{\text{ex}})$ to denote the maximum advantage of any passive adversary attacking P, running in time t and making q_{ex} calls to the $\mathsf{Execute}$ oracle. For the authenticated group KE we use $\mathsf{Adv}_P^{\mathsf{AKE-fs}}(t, q_{\text{ex}}, q_{\text{s}})$, where q_{s} refers to the number of Send queries.

The security of the example protocols will be based on the decisional Diffie-Hellman problem (DDHP), let $\mathsf{Adv}_G^{\mathsf{ddh}}(t)$ be the advantage of a PPT adversary against the DDHP in a group G of order ℓ. The DDHP is the problem of distinguishing the distributions of $\{(g^a, g^b, g^{ab}) : a, b \in_R \mathbb{Z}/\ell\mathbb{Z}\}$ from $\{(g^a, g^b, g^c) : a, b, c \in_R \mathbb{Z}/\ell\mathbb{Z}\}$ given g. For a single triple this amounts to deciding whether the triple $(g_1, g_2, g_3) \in G^3$ is a valid Diffie-Hellman triple, i.e., if $g_3 = g_2^{\log_g g_1}$.

2.3 Complexity

Group KE protocols are often carried out in dynamic sets of players. One important feature of good protocols is their scalability. To take this into account we always consider the maximal complexity occurring for any user in the system. Furthermore, we consider the number of users n as the important parameter and ignore costs depending on the cryptographic primitive like the size of the underlying group $\langle g \rangle$.

The communication complexity is the maximal amount of information *sent and received* per user. We assume that broadcasting a message does not depend on the number of receivers, however, receiving l *different* messages means cost of l, even if this occurs in one round. Katz and Yung [19] mention *only sent messages* but use it in the same way. We mention the received messages explicitly to take into account the costs for being online, and for receiving and storing messages. At the same time we also allow users to ignore communication not intended for them.

The computation complexity is the maximal amount of computation during one call of the protocol.

In both cases we are interested in the dependence on n, all other variables like group size or security parameter are considered as constant in the big-O estimates.

3 The Burmester-Desmedt Scheme II

We now describe in detail one scheme with logarithmic complexity. This example serves two purposes: on the one hand it gives a good example of the advantage of our compiler over that of Katz and Yung and on the other hand it closes the gap that [8] was published without security proofs.

The BD-II scheme [8] is a compiler that transforms a two-party KE protocol into a multiparty key exchange protocol. There are two variants. The first is sequential with delays. The second is a multicast version with minimal delay.

To ease exposition we focus on the Diffie-Hellman version of the BD-II scheme [8, p. 127] (one can use other two party KE protocols as primitives) and use a cyclic group $G = \langle g \rangle$ of prime order (although the proof remains similar if the last condition is not satisfied). The advantage of BD-II over other group KE protocols is that the communication complexity, the computation complexity and memory reduce from $O(n)$ per party to $O(\log n)$ per party in the multicast version, and to constant in the sequential version which has $O(\log n)$ delay. The number of rounds each party is involved in (per instance) is constant and equal to three for both variants.

In this section we briefly recall the BD-II protocol [8] in the multicast version and give a proof of security of the unauthenticated scheme; a sequential version is given in the Appendix.

Let $U_1, \ldots, U_n$ be the users who want to make a group KE. We now show how their index automatically determines their place in the (almost) binary tree as in Figure 1. This ordered tree has the property that user U_i is at level $\lfloor \log_2(i+1) \rfloor$

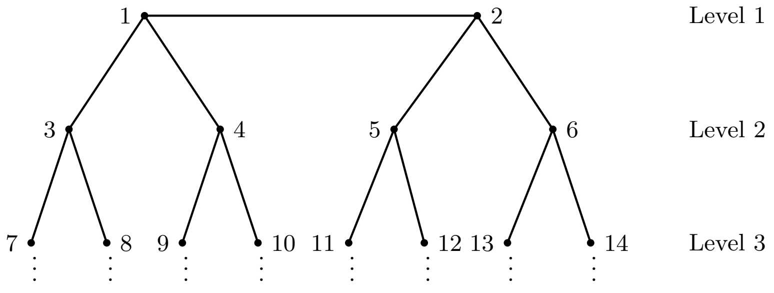

Fig. 1. The binary BD-tree in the BD-II scheme.

(a rooted tree version would need one more multiplication round and is therefore avoided). The set-up can be easily done as the position of user U_i corresponds to the binary expansion of $i + 1$, *i.e.*, if $i + 1 = (i_{l-1}i_{l-2} \ldots i_1 i_0)_2$ is the binary expansion with i_j the leftmost non-zero bit then U_i is on level j and his parent has expansion $\text{parent}(i) + 1 = (0 \ldots 0 i_j i_{j-1} i_1)_2$ (one more leading zero). Likewise the children can easily be determined as their expressions are shifted to the right and they have the same initial binary representation with different concatenated tails.

The vertices 1 and 2 consider their respective opposite as parent. So all but the leaves of the tree each have one parent and two children. To ease notation let ancestors(i) be the set of indices of all ancestors of U_i, including i *but* having removed 1 and 2. Let parent(i), left child(i) and right child(i) be the indices of the parent of U_i and its left child or right child respectively.

Remark 1. One advantage of this scheme is that users only need to know users higher up in the hierarchy. This fits well to common applications of GKE in groups invoked by some manager higher up in the hierarchy who decides the members of the group and their respective function. Once the GKE protocol is started the users have already received some information publicly and their public keys are available in a public database.

While the BD-I scheme requires that *each* user needs to know the position of *every* other user relative to him, our scheme limits this to the knowledge of the positions of at most $O(\log n)$ users.

Note that all this is usually not a problem. The user name in the protocol should contain information about the user's position (the numbers $1, 2, \ldots, n$ can be encoded with $\log n$ bits) and this number is enough to determine the relative positions for both BD-I and BD-II.

We only stress this observation since it is a often claimed as a disadvantage of BD-II that the position needs to be known even though this is true even more for BD-I.

Protocol 1 (BD-II group KE)
Let $U_1, \ldots, U_n$ be the set of all users who want to generate a common conference key. Assume that they are arranged in the binary tree as in Figure 1. The key exchange is performed in a group $\langle g \rangle$ of order ℓ with generator g.

Step 1 *Each U_i, $i = 1, \ldots, n$, selects $k_i \in_R \mathbb{Z}/\ell\mathbb{Z}$, computes and sends $z_i = g^{k_i}$ to his parent and children.*

Step 2 *Each U_i, $i = 1, \ldots, n$, computes and multicasts to its descendants:*

$$X_{\text{left child}(i)} = (z_{\text{parent}(i)}/z_{\text{left child}(i)})^{k_i},$$

$$X_{\text{right child}(i)} = (z_{\text{parent}(i)}/z_{\text{right child}(i)})^{k_i}.$$

Step 3 *Each U_i, $i = 1, \ldots, n$, computes the conference key,*

$$K_i = z_{\text{parent}(i)}^{k_i} \prod_{j \in \text{ancestors}(i)} X_j.$$

Remark 2. Honest users compute the same key,

$$K = g^{k_1 k_2}.$$

We prove this claim by induction. First observe that $K_1 = K_2 = g^{k_1 k_2}$. Next let $K_{\text{parent}(i)} = K$. Then since $K_{\text{parent}(i)} = z_{\text{parent}(\text{parent}(i))}^{k_{\text{parent}(i)}} \prod_{j \in \text{ancestors}(\text{parent}(i))} X_j$ it is obvious that we have:

$$K_i = z_{\text{parent}(i)}^{k_i} * (z_{\text{parent}(\text{parent}(i))}^{k_{\text{parent}(i)}})^{-1} * X_i * K_{\text{parent}(i)}.$$

Since

$$X_i = (z_{\text{parent}(\text{parent}(i))}/z_i)^{k_{\text{parent}(i)}} \tag{1}$$

and $z_{\text{parent}(i)}^{k_i} = z_i^{k_{\text{parent}(i)}}$ we obtain $K_i = K$.

Remark 3. Obviously this scheme is vulnerable to denial of service attacks. This is always an issue in group KEs and more severely in non-contributory schemes as users on lower levels or on different branches cannot notice that some members might be excluded. However, as stated before, like Katz-Yung, we only deal with attacks from outsiders.

4 Proof of Security against Passive Attacker for BD-II

We first show that an attacker against Protocol 1 can be used to solve the decisional Diffie-Hellman problem The proof is an adaption of Burmester-Desmedt's proof for BD-I in [9].

Theorem 1. *Protocol P is a secure group KE protocol. Namely*

$$\mathsf{Adv}_P^{\mathsf{KE}}(t, q_{\text{ex}}) \leq \mathsf{Adv}_G^{\mathsf{ddh}}(t'),$$

where $t' = t + O(|\mathcal{P}|q_{\text{ex}}t_{\text{exp}})$, q_{ex} is the number of Execute *queries, and t_{exp} is the time required to perform exponentiations in G.*

Note that the time for the execution of P is made explicit by the use of t_{exp}.

Proof. Given an algorithm $\mathcal{A}$ against P running in time t we show how to build a distinguisher $\mathcal{D}$ against the DDHP. First consider the case that $\mathcal{A}$ makes a single Execute query.

Let $\mathcal{D}$ be given a triple $(g_1, g_2, g_3) \in \langle g\rangle^3$. Now $\mathcal{D}$ can generate a valid transcript for $\mathcal{A}$ as shown below. Then $\mathcal{D}$ runs $\mathcal{A}$ on this transcript and outputs 1, i.e., the claim that (g_1, g_2, g_3) is a valid Diffie-Hellman triple, if $\mathcal{A}$ outputs 1 and outputs 0 otherwise.

Put $z'_1 = g_1$ and $z'_2 = g_2$. Randomly choose $c_3, \ldots, c_n \in_R \mathbb{Z}/\ell\mathbb{Z}$ and put $z'_i := z'_{\text{parent}(\text{parent}(i))}g^{-c_i}$ for $i \geq 3$. So, $z'_3 := g_2 \cdot g^{-c_3}$, $z'_5 := g_1 \cdot g^{-c_5}$, etc.

Consistent X'_i's are obtained as $X'_i := (z'_{\text{parent}(i)})^{c_i}$ for $i \geq 3$ is easy to verify. As the c_i $(i \geq 3)$ are distributed uniformly at random, the distribution of z'_i and X'_i is identical to that in P.

The transcript consists of $\mathsf{T} = (z'_1, \ldots, z'_n, X'_3, \ldots, X'_n)$. Upon the Test request, $\mathcal{D}$ issues $\mathsf{sk}' = g_3$.

Indeed, if sk' is the valid group key then $g_3 = \mathsf{sk}' = {z'_1}^{\log_g z'_2} = g_1^{\log_g g_2}$, i.e., (g_1, g_2, g_3) is a valid Diffie-Hellman triple. So $\mathcal{D}$ succeeds with the same advantage as $\mathcal{A}$ and needs $(2n-4)t_{\text{exp}}$ additional time for the exponentiations to generate the transcript.

If more than one execution of the protocol should be allowed, one can easily generate further triples of the same type as (g_1, g_2, g_3) by L $=$

$\{(g_1, g_2, g_3), (g_1^{r_2}, g_2^{r_2}, g_3^{r_2}), \ldots, (g_1^{r_{q_{\mathrm{ex}}}}, g_2^{r_{q_{\mathrm{ex}}}}, g_3^{r_{q_{\mathrm{ex}}}})\}$ for random exponents $r_j \in_R \mathbb{Z}/\ell\mathbb{Z}$.

Bounding the number n by the total number of participants $|\mathcal{P}|$ the claim follows. □

Note that the computational complexity of computing the group key under a passive attack for an outsider corresponds to the Computational Diffie-Hellman problem, as is easy to verify (similar as in [9]).

This protocol does not involve any longterm secrets, so $\mathsf{Corrupt}$ queries need not be taken into account and the protocol automatically achieves forward security.

5 Authenticated Group Key Exchange

A direct application of the compiler of Katz and Yung [19] would transform Protocol 1 into an authenticated KE protocol with running time and communication complexity $O(n)$. We now provide our adjusted compiler which allows us to stay in $O(\log n)$. To simplify the presentation we assume that the reader is familiar with [19] and will only mention the differences.

To avoid replay attacks, Katz-Yung [19] introduce fresh randomness r_i per user U_i for each execution of the protocol, add a message number for each user, and make the signature contain information on the group of players in the AKE. We follow the same road and so we modify the protocol to always send $U|j|m$ (the user name, the message number and the message) and not only m or $U|m$.

We observe that in BD-II user U_i computes the $X_{\mathrm{child}(i)}$ depending only on information from its parent and two children. To compute the group key $\mathsf{sk}_{U_i} = K_i$, user U_i ($i \neq 1, 2$) uses information coming only from the same branch of the binary tree from nodes on levels above U_i while U_1 and U_2 use information by their respective parents. In general, in each group KE protocol there is a clearly defined ordered set of messages used by a specific user U_i and our compiler requires to check only signatures on these messages. In most tree based systems the set of users any specific user communicates with is much smaller than the total set of users; even to the extent that the number of used messages is logarithmic in the total number of users. In BD-I, however, each user processes input from every user.

We formulate the compiler in a more general setting which contains Katz-Yung's as a special case, we use the term "multicast" to stress that not all users need to receive the message. To link the messages to the structure in which the users are arranged we let the set $\mathsf{rel}_U = \{V_1, V_2, \ldots, V_{t_U}\}$ be the set of users whose input is processed by user U at some point in the protocol and U itself. E.g. in the BD-I protocol rel_U is the whole group for any user U while in BD-II rel_U is the set of all ancestors and both children of U; it contains at most $O(\log n)$.

1. During the initialization phase, each party $U \in \mathcal{P}$ generates the verification/signing keys (PK'_U, SK'_U) by running $\mathcal{G}(1^k)$. This is in addition to any keys (PK_U, SK_U) needed as part of the initialization phase of P.

2. Let $U_1, \ldots, U_n$ be the identities of users wishing to establish a joint group key and let gid be their group identifier[6]. Each user U_i chooses some random nonce $r_i \in \{0,1\}^k$ and broadcasts $U_i|0|r_i$. So for each execution of the protocol fresh randomness is used and so replay is not possible.
   Let $\mathsf{rel}_U = \{V_1, V_2, \ldots, V_{t_U}\}$ be as above. Each instance Π_U^j stores the identities and their per-round randomness together with the group ID in $\mathsf{direct}_U^j = (\mathsf{gid}|V_1|r_1|\ldots|V_{t_U}|r_{t_U})$ and stores this as part of the state information.
3. The members of the group now execute the protocol P with the following changes:
   – Whenever instance Π_U^i is supposed to multicast $U|j|m$ as part of protocol P, the instance computes $\sigma = \mathsf{Sign}_{SK'_U}(j|m|\mathsf{direct}_U^i)$ and multicasts $U|j|m|\sigma$.
   – Before using message $V|j|m|\sigma$ the instance Π_U^i checks that (1) $V \in \mathsf{pid}_U^i$[7], (2) j is the next expected sequence number for messages from V, and, finally, (3) that $\mathsf{Vrfy}_{PK'_V}(j|m|\mathsf{direct}_V^i, \sigma) = 1$. If any of these are untrue, Π_U^i aborts the protocols and sets acc_U^i = FALSE and sk_U^i = NULL. Otherwise, Π_U^i continues as it would in P and uses the message m.

Remark 4. We like to stress that the authentication does not prevent attacks by malicious insiders. Like Katz and Yung state in [19, Section 2.1] these definitions cannot achieve "agreement" in the sense of [14], e. g. since the attacker could stop all communications by denial of service attacks.

Remark 5. The overhead introduced by the compiler does not change the complexity classes for communication and computation. The number of signatures a user makes is equal to the number of messages he sends, the length of the message is extended by adding the message number and direct_U^i. The latter has length equal to the number of partners U directly communicates with and thus is reflected by the computation costs. Reading through the list as part of the signing process does not change the complexity. The number of signature verifications equals the number of processed messages. If user U has to verify that $\mathsf{Vrfy}_{PK'_V}(j|m|\mathsf{direct}_V^i, \sigma) = 1$ then V must be in rel_U and thus U knows V's position and thus also direct_V.

By sticking to the very same security definition as [19] we show that our compiler works just as well, and in particular allows having a better complexity if the underlying protocol does.

Theorem 2. *If P is a secure group KE protocol achieving forward secrecy, then P' given by the above compiler is a secure group AKE protocol achieving forward*

[6] Katz and Yung suggest to use $\mathsf{gid} = U_1|\ldots|U_n$ which automatically forces them to deal with inputs of length $O(n)$. This does not pose a problem for them because BD-I has complexity $O(n)$ anyway. In practical situations, however, the group is formed by e. g. a manager in a bank or a network administrator at the same time as deciding membership and the number of actually used groups is on a much smaller scale than the total number of all possible groups. Quite commonly there there exists a compact description, e.g. the name of the task force.

[7] Our definitions of pid and direct ensure that all players V sending necessary input are actually present together with their initial randomness.

secrecy. Namely, for q_{s} the number of Send *queries and q_{ex} the number of* Execute *queries we obtain*

$$\mathsf{Adv}_{P'}^{\mathsf{AKE-fs}}(t, q_{\mathrm{ex}}, q_{\mathrm{s}}) \leq \frac{q_{\mathrm{s}}}{2} \cdot \mathsf{Adv}_{P}^{\mathsf{KE}}(t', 1) + \mathsf{Adv}_{P}^{\mathsf{KE}}(t', q_{\mathrm{ex}}) \\ + |\mathcal{P}| \cdot \mathsf{Succ}_{\Sigma}(t') + \frac{q_{\mathrm{s}}^2 + q_{\mathrm{ex}} q_{\mathrm{s}}}{2^k},$$

where $t' = t + (|\mathcal{P}| q_{\mathrm{ex}} + q_{\mathrm{s}}) \cdot t_{P'}$ and $t_{P'}$ is the time required for an execution of P' by any party and Succ_{Σ} *is the success probability against the used signature scheme Σ.*

The compiler maintains the number of rounds and the complexity class of the protocol P.

Proof. The proof follows the same lines at the corresponding one in [19]. We use an active adversary $\mathcal{A}'$ against P' to construct an adversary against P. There are three ways in which $\mathcal{A}'$ can succeed, namely by forging a signature if he has not queried the Corrupt oracle before, by repeating the information direct and thus reusing a signature obtained in a previous execution, or by distinguishing the key from random. It is only the latter that leads to an attack against P.

The contribution of the event Forge is as in [19], namely

$$\Pr[\mathsf{Forge}] \leq |\mathcal{P}| \cdot \mathsf{Succ}_{\Sigma}(t').$$

Due to the different definition of direct compared to nonces in [19] we need to reconsider the probability of Repeat. The probability that the nonce used by any user in response to an initial Send query was previously used by that user (in either another execution of Send or in an Execute query) is still bounded by

$$\Pr[\mathsf{Repeat}] \leq \frac{q_{\mathrm{s}}(q_{\mathrm{s}} + q_{\mathrm{ex}})}{2^k}.$$

[8]

The remainder of the considerations works as in [19]. Let Ex be the event that $\mathcal{A}'$ queries the Test oracle to an instance Π_U^i such that $\mathcal{A}'$ never made a query of the form $\mathsf{Send}(U, i, *)$, i.e., in the simulation $\mathcal{A}$ has the full transcript without patches from its own execution of P. Defining $\mathsf{Se} = \overline{\mathsf{Ex}}$ and considering the probabilities $\Pr_{\mathcal{A}',P'}[\mathsf{Succ} \wedge \mathsf{Ex}]$ and $\Pr_{\mathcal{A}',P'}[\mathsf{Succ} \wedge \mathsf{Se}]$ separately by constructing appropriate adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ one obtains the stated result. □

Corollary 1. *The authenticated group key exchange protocol obtained from Protocol 1 by applying this compiler is secure against active attacks and has communication and computation complexity $O(\log n)$.*

[8] We like to stress that this is the point where an *inside attacker* has much larger chances against our protocol than against the authenticated BD-I protocol as some messages might be concatenated with very short strings direct meaning that a collusion of only few users is necessary which do not change their random strings and thus enable replay attacks. Likewise, the assumption (which is also made in [19]) that messages are delivered and the protocol is consistent is necessary here to maintain the same level of security.

Remark 6. The original paper [8] describes the system in more generality allowing more than two children per vertex. For k children this means that each vertex has to compute k values for X_i whereas the final computation of the key reduces to $\log_k n$ computations only. While the overall number of operations is smallest for $k = 2$; larger k might be an interesting alternative if the storage is restricted and only $\log_k n$ elements can be stored from the multicast to compute K. In the sequential version (see Appendix) a larger k reduces the delay while making each step more expensive. Again the maximal overall efficiency is obtained for $k = 2$.

In an authenticated version, however, at most $k + \log_k n$ signatures need to be verified and $k + 1$ messages must be signed. Additionally $k + 2$ exponentiations are needed. In the ElGamal signature scheme, two exponentiations are needed per signature generation and a multi-exponentiation is used for verification. Accordingly a larger k reduces the computational complexity.

6 Comparison

To show the advantages of our group AKE we give an overview of the costs for the naive version (one party transmits the key to the next participant, etc.), the BD-I version considered in [19] and the two authenticated BD-II versions proposed in this paper, in the table below – see Table 6.

We also take into account the scheme by Boyd and Nieto [5] and the scheme by Bresson et al. [6]. In the Boyd-Nieto scheme one user U_1 chooses the key and encrypts this key for all $n - 1$ other group members. This long message is signed and broadcast to all participants who then check the signature and decrypt their part to obtain the joint key. If a public key system is used for the encryption this scheme runs in 1 round. Otherwise a further round is necessary in which each party U_i (including U_1) sends g^{k_i}. Then the Diffie-Hellman key $g^{k_1 k_i}$ is used to transmit the group key. This scheme requires the party U_1 to perform $O(n)$ computations and the message has length $O(n)$. We denote the first version by BN PK and the second one by BN DH.

In [6] the authors use $g^{k_1 k_2 k_3 \dots k_n}$ as the joint secret key. To make this possible, the users sequentially add their contribution k_i to the key which accounts for requiring n rounds and the last user needs to send $g^{(k_1 k_2 k_3 \dots k_n)/k_i}$ to user U_i for each i, so he has communication and computation $O(n)$.

Except for [5] and [6] group KE schemes are balanced in workload in the sense that all users have about the same amount of work. The table below states the *maximal* values per user. Katz and Yung use the same measures because maximal effort per user captures scalability.

To describe the message and communication complexity, we use **p** to denote point-to-point communication and **b** to denote broadcast. As the set of users is finite we do not distinguish between multicast and broadcast, otherwise the BD-II protocols use only multicast while BD-I needs broadcast. Furthermore, we list the maximal length of the messages.

For the computation S means signatures, V means verifying, E stands for full exponentiation and M for multiplication. For Boyd-Nieto PK we assume ElGamal encryption; the costs for retrieving the public keys are not mentioned. We neglect the number of inversions as there are at most 2 of them. For the number of rounds we consider the maximal delay of the protocol. Except for the second version of BD-II and [6] this is the maximum number of rounds per user. In the last protocol there are 3 rounds per user.

	Rounds	messages	communication	length	computation
naive	$n-1$	$1\mathbf{b}, 2\mathbf{p}$	$1\mathbf{b}, 2\mathbf{p}$	$O(1)$	$2S, 3V, 3E, 2M$
[6]	n	$2\mathbf{b}$	$(n-1)\mathbf{p}, 2\mathbf{b}$	$O(1)$	nS, nV, nE, nM
BN PK [5]	1	$1\mathbf{b}$	$1\mathbf{b}$	$O(n)$	$1S, nV, 2(n-1)E, (n-1)M$
BN DH [5]	2	$1\mathbf{p}, 1\mathbf{b}$	$(n-1)\mathbf{p}, 1\mathbf{b}$	$O(n)$	$2S, nV, nE, (n-1)M$
BD-I	3	$2\mathbf{p}, 1\mathbf{b}$	$4\mathbf{p}, n\mathbf{b}$	$O(1)$	$2S, nV, 3E, (2n-1)M$
BD-II	3	$3\mathbf{p}, 1\mathbf{b}$	$6\mathbf{p}, (\log_2 n)\mathbf{b}$	$O(1)$	$2S, (\log_2 n)V, 4E, (\log_2 n)M$
BD-II seq.	$(\log_2 n)$	$5\mathbf{p}$	$6\mathbf{p}$	$O(1)$	$3S, 4V, 4E, 2M$

Table 1. A comparison of the overhead costs of six AKE schemes.

Accordingly, our authenticated BD-II version achieves an overall complexity of $O(\log n)$ while Katz-Yung [19] based on BD I need $O(n)$ in communication and in computation. Additionally, the number of messages to be stored from the broadcast is $O(\log n)$, as only the messages of the maximal $\log n$ ancestors are needed. The sequential version has a delay of $\log n$ but reduces the requirements considerably, *e.g.,*, no broadcast is assumed and far fewer operations. This might be interesting in restricted networks which need to minimize the computational and technical requirements trading it off for a longer overall execution of the protocol.

7 Conclusions

We have presented a compiler that adds authenticity to any group KE protocols while preserving computation and communication complexities of the original group KE protocol. In particular we have detailed a secure group AKE protocol based on the BD-II group KE which has a constant number of rounds and requires only $O(\log n)$ computation and communication. This is, essentially, an exponential saving compared to previously proposed AKE protocols.

References

1. R. Barua, R. Dutta, and P. Sarkar. Extending Joux's protocol to multi party key agreement. In *Progress in Cryptology - INDOCRYPT 2003* volume 2904 of *Lect. Notes Comput. Sci.*, pages 205–217. Springer 2003.
2. R. Barua, R. Dutta, and P. Sarkar. Provably secure authenticated tree based group key agreement protocol using pairing. In *ICICS 2004*, volume 3269 of *Lect. Notes Comput. Sci.*, pages 92–104. Springer, 2004. see also: ePrint archive, 2004/090, 2004.
3. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advances in cryptology – Crypto 2001*, volume 2139 of *Lect. Notes Comput. Sci.*, pages 213–229. Springer, 2001.
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3), pages 586–615, 2003.

5. C. Boyd and J.M.G. Nieto. Round-optimal contributory conference key agreement. In *Public Key Cryptography 2003*, volume 2567 of *Lect. Notes Comput. Sci.*, pages 161–174. Springer, 2003.
6. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *Proc. 8th Annual ACM Conference on Computer and Communications Security*, pages 255–264, 2001.
7. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology – Eurocrypt '94*, volume 950 of *Lect. Notes Comput. Sci.*, pages 275–286, Springer, 1995.
8. M. Burmester and Y. Desmedt. Efficient and secure conference key distribution. In *Security Protocols*, volume 1189 of *Lect. Notes Comput. Sci.*, pages 119–130, Springer, 1997.
9. M. Burmester and Y. Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3), pages 137–143, 2005.
10. M. Burmester and Y. Desmedt. Identity-based Key Infrastructures (IKI). In *Security and Protection in Information Processing Systems – SEC 2004*, pages 167–176, Kluwer, 2004.
11. K.Y. Choi, J.Y. Hwang, and D.H. Lee. Efficient ID-based group key agreement with bilinear maps. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lect. Notes Comput. Sci.*, pages 130–144, Springer, 2004.
12. Y. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. A Non-Malleable Group Key Exchange Protocol Robust Against Active Insiders. In *Information Security Conference – ISC 2006*, volume 4176 of *Lect. Notes Comput. Sci.*, pages 459–475, Springer, 2006.
13. X. Du, Y. Wang, J. Ge, and Y. Wang. An improved ID-based authenticated group key agreement scheme. ePrint archive, 2003/260, 2003.
14. M.J. Fischer, N.A. Lynch, and M.S. Patterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2), pages 374–382, 1985.
15. I. Ingemarsson, D.T. Tang, and C.W. Wong. A conference key distribution system. *IEEE Trans. Inform. Theory*, 28, pages 714–720, 1982.
16. A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory, ANTS-IV*, volume 1838 of *Lect. Notes Comput. Sci.*, pages 385–394, Springer, 2000.
17. M. Just and S. Vaudenay. Authenticated multi-party key agreement. In *Advances in Cryptology – Asiacrypt 1996*, volume 1163 of *Lect. Notes Comput. Sci.*, pages 36–49, Springer, 1996.
18. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. *ePrint archive, 163/2005.*
19. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology – Crypto 2003*, volume 2729 of *Lect. Notes Comput. Sci.*, pages 110–125. Springer, 2003. full version available at `http://www.cs.umd.edu/~jkatz/research.html`.
20. J. Nam, Y. Lee, and D. Won. Constant Round Group Key Exchange with Logarithmic Computational Complexity. *ePrint archive, 284/2006.*
21. Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Secur.*, 7(1), pages 60–96, 2004.
22. J. Pieprzyk and H. Wang. Malleability attacks on multi-party key agreement protocols. In *Coding, Cryptography and Combinatorics*, volume 23 of *Progress in Computer Science and Applied Logic*, pages 277–288. Birkhäuser, 2004.
23. C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. In *SIGCOMM*, pages 68–79, 1998.
24. C. K. Wong, M. G. Gouda, and S. S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1), pages 16–30, 2000.

Appendix – A Peer-to-peer version of BD-II

The following gives a sequential version with only peer to peer communication and constant memory requirements having a $O(\log n)$ delay by changing slightly the protocol. Note that the number of rounds in which a party is actively involved remains unchanged but a user corresponding to a leaf has $\log n$ delay.

Protocol 2 (Peer-to-peer version of BD-II group KE)
Let $U_1, \ldots, U_n$ be a (dynamic) subset of all users who want to generate a common conference key. Assume that they are arranged in the binary tree as in Figure 1. The key exchange is performed in a group of order ℓ with generator g.

Step 1 *Each U_i, $i = 1, \ldots, n$, selects $k_i \in_R \mathbb{Z}/\ell\mathbb{Z}$, computes and sends $z_i = g^{k_i}$ to his parent and children.*
Step 2 *Each $U_i, i = 1, \ldots, n$ computes $X_{p,i} = z^{k_i}_{\text{parent}(i)}, X_{l,i} = z^{k_i}_{\text{left child}(i)}$ and $X_{r,i} = z^{k_i}_{\text{right child}(i)}$.*
Step 3 *U_1 and U_2 have already obtained the joint key $K_i = X_{p,i}$ and send $Y_{\text{left child}(i)} = K_i \cdot X_{l,i}$ and $Y_{\text{right child}(i)} = K_i \cdot X_{r,i}$ to their respective children.*
Step 4 *For $j = 2, \ldots, m$ do:*
each user U_i on level j computes upon receipt of Y_i the joint key as

$$K_i = y_i / X_{p,i}.$$

Then he sends $Y_{\text{left child}(i)} = K_i \cdot X_{l,i}$ and $Y_{\text{right child}(i)} = K_i \cdot X_{r,i}$ to his respective children.

Remark 7. For this scheme it is even more obvious why honest users obtain the same key

$$K = g^{k_1 k_2}.$$

First note that this holds for K_1 and K_2. Assume that the parent of U_i obtained $K_{\text{parent}(i)} = K$. Then U_i computes $K_i = X_{p,i} Y_i = X_{p,i}(X_{p,\text{child}(\text{parent}(i))})^{-1} K_{\text{parent}(i)} = K$, where child takes into account the correct value of left or right.