Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys

Michel Abdalla, Céline Chevalier, Mark Manulis, David Pointcheval

This is the full version of the extended abstract which appears in Africacrypt '10 (3 – 6 may 2010, Stellenbosch, South Africa) D. Bernstein and T. Lange Eds., Springer-Verlag, LNCS 6055, pages 351–368.

Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys

Michel Abdalla[1], Céline Chevalier[2], Mark Manulis[3], and David Pointcheval[1]

[1] École Normale Supérieure, CNRS-INRIA, Paris, France
[2] Telecom ParisTech, Paris, France
[3] Cryptographic Protocols Group
Department of Computer Science, TU Darmstadt & CASED, Germany
mark@manulis.eu

Abstract. Modern multi-user communication systems, including popular instant messaging tools, social network platforms, and cooperative-work applications, offer flexible forms of communication and exchange of data. At any time point concurrent communication sessions involving different subsets of users can be invoked. The traditional tool for achieving security in a multi-party communication environment are group key exchange (GKE) protocols that provide participants with a secure group key for their subsequent communication. Yet, in communication scenarios where various user subsets may be involved in different sessions the deployment of classical GKE protocols has clear performance and scalability limitations as each new session should be preceded by a separate execution of the protocol. The motivation of this work is to study the possibility of designing more flexible GKE protocols allowing not only the computation of a group key for some initial set of users but also efficient derivation of independent secret keys for all potential subsets. In particular we improve and generalize the recently introduced GKE protocols enabling on-demand derivation of peer-to-peer keys (so called GKE+P protocols). We show how a group of users can agree on a secret group key while obtaining some additional information that they can use on-demand to efficiently compute independent secret keys for *any* possible subgroup. Our security analysis relies on the Gap Diffie-Hellman assumption and uses random oracles.

1 Introduction

Despite more than 20 years of research (see surveys in [5, 28]), group key exchange (GKE) protocols are still far from being widely used in practice, especially if one compares to two-party key (2KE) exchange protocols which found their deployment in various standards and daily applications. Among the main reasons for this limited spectrum of applications is the fact that GKE protocols are rather complex, costly to implement, and not versatile enough. Modern communication platforms, including diverse instant-messaging tools and collaborative applications, allow their users to communicate and exchange data within almost any subset of participants. Had classical GKE protocols been deployed in this multi-user environment, then every new communication attempt would require a different execution of the GKE protocol. This constraint is anchored in the design rationale behind existing GKE protocols. A possible improvement would be to design more flexible GKE protocols, which would not only provide their participants with the group key but would also leave space for the efficient computation of secret keys for use within any subset of the original group.

Recently, Manulis [27] suggested flexible GKE protocols allowing for a secure combination of two communication forms: secure communication within a group and secure communication amongst any two parties from the group. In particular, he designed GKE protocols enabling on-demand derivation of peer-to-peer (p2p) keys, denoted GKE+P, which provide any pair of participants with an independent secure p2p key in addition to the common group key. The main ingredient of GKE+P constructions

in [27] is the parallel execution of the classical Diffie-Hellman key exchange (PDHKE) protocol [17] and, in particular, the user's ability to re-use the same value g^x in the computation of group and p2p keys, where g is a generator and x is the private user's exponent.

Building on this, we investigate the even more generalized and flexible approach — the extension of GKE protocols with the ability to compute an independent session key for any possible *subgroup* of the initial GKE participants. Similar to [27], we are interested in solutions which would allow the derivation of the subgroup keys in a more efficient way than simply running an independent session of a GKE protocol for each subgroup. GKE protocols enriched in this way, which we denote GKE+S, would allow the combination of different forms of secure communication. For example, a single file deposited in a file sharing network or broadcasted to the group may contain documents encrypted for the whole group and different attachments encrypted for different subgroups.

The required independence between different key types imposes further security challenges on GKE+S protocols: The classical GKE security requirement concerning the secrecy of the group key with respect to the external parties (typically called AKE-security [8, 22]) should now be preserved even if some subgroup keys leak, and the independence of any subgroup key implicitly requires us to handle (collusion) attacks from parties that are external to that subgroup but internal to the preliminary computation process of the common group key or to a different subgroup with membership overlap. As a result, the specification of the adequate security requirements for GKE+S protocols with respect to these threats appears to be an interesting task from a formal point of view.

The protocols being considered in this paper are all based on the well-studied Burmester-Desmedt (BD) group key exchange protocol [13]. Interestingly, Manulis [27] had shown that, if users try to use the same exponents in the computation of group and p2p keys in the original BD protocol, then the AKE-security of p2p keys could no longer be guaranteed. In this work, we illustrate how a small modification of the original BD protocol suffices to obtain a secure GKE+P protocol. Interestingly, this modification of the original BD protocol only applies to the computation steps and leaves the communication complexity unchanged. In particular, our GKE+P protocol has better overall complexity than those proposed in [27]. After presenting our new GKE+P protocol, we show how to extend it into a secure GKE+S protocol. Our GKE+S protocol requires only one additional communication round for setting up any subgroup key as opposed to the two-round communication that one would need to compute a subgroup key via an independent protocol execution.

1.1 Related Work

Security of key exchange protocols is usually defined with the requirement of Authenticated Key Exchange (AKE) security; see [3, 14, 15, 26] for the 2KE case and [8, 10, 18, 21, 22] for the GKE case. AKE-security models the indistinguishability of the established session group key with respect to an active adversary treated as an external entity from the perspective of the attacked session. The signature-based compilation technique by Katz and Yung [22] (see also the work by Bresson, Manulis, and Schwenk [12]) can be used to achieve AKE-security for GKE protocols that already provide such indistinguishability but with regard to the passive attacks only. Besides AKE-security some security models for GKE protocols (e.g. [10, 11, 18, 21]) define optional security against insider attacks.

The notion of GKE+P protocols has been put forth by Manulis [27]. He showed how to compile the so-called family of *Group Diffie-Hellman* protocols, i.e. protocols such as [13,16,19,24,25,30–32] which extend the classical Diffie-Hellman method to the group setting, in such a way, that at the end of the protocol any pair of users can derive their own p2p key on-demand and without subsequent communication. The main building block of his GKE+P compiler is the parallel Diffie-Hellman key exchange (PDHKE) in which each user broadcasts a value of the form g^x and uses x for the derivation of different p2p keys. For the two efficient two-round unauthenticated GKE protocols by Burmester and Desmedt (BD) [13] and by Kim, Perrig, and Tsudik (KPT) [24], in which users need to broadcast g^x anyway, Manulis analyzed optimizations based on the re-use of the exponent x for the computation of both group and p2p keys showing that KPT remains secure whereas BD not.

We also notice that PDHKE has been used by Jeong and Lee [20] for the simultaneous computation of multiple two-party keys amongst a set of users, yet without considering collusion attacks against the secrecy of keys computed by non-colluding users and without considering group keys. In another work, Biswas [4] proposed a slight modification to the original Diffie-Hellman protocol allowing its participants to obtain up to 15 different shared two-party keys.

While GKE+P protocols take the top-down approach in the sense that the computation of p2p keys for any pair of users is seen as a feature of the GKE protocol there have been several suggestions, e.g. [1,29,33], that construct GKE protocols from 2KE protocols used as a building block. For example, Abdalla et al. [1], as well as Wu and Zhu [33] order protocol participants into a cycle (in a BD fashion) and a 2KE protocol is executed only between the neighbors. However, these constructions do not explicitly meet the requirements of GKE+P protocols as an independent p2p key may not be available for every pair of users.

1.2 Contributions and Organization

In Section 2, we recall the classical Burmester-Desmedt (BD) protocol [13] and explain problems behind the re-use of its exponents for PDHKE following the analysis by Manulis [27]. In Section 3, we propose a slight modification to the computations steps of the BD protocol such that the overall communication complexity remains unchanged, yet secure, non-interactive derivation of p2p keys through the re-use of users' exponents becomes possible. Our GKE+P protocol obtained in this way and denoted mBD+P is more efficient than other solutions proposed in [27].

We then continue in Section 4 with our second contribution — the description of a GKE+S protocol in which users compute the common group key first and then any subgroup of these users can agree on an independent subgroup key. Our GKE+S protocol, denoted mBD+S, requires two-rounds for the computation of the group key and only one round for the subsequent on-demand computation of any subgroup key. In comparison to the naive approach of computing a subgroup key through an independent execution of the full GKE protocol our solution is more scalable as it halves the number of communication rounds and required messages.

In Section 5, we formally model the security of GKE+S protocols. We address AKE-security of both group and subgroup keys whereby security of the latter is modeled under consideration of possible collusion attacks representing the main challenge is such protocols. Our model generalizes the security model by Manulis [27], which considered only derivation of p2p keys.

Finally, we apply the model in Section 6 to prove that mBD+P provides AKE-security for group and p2p keys and that mBD+S additionally provides AKE-security for the derived subgroup keys. We conclude our work in Section 7 by comparing the communication and computation complexity of our protocols with the previous solutions showing the expected efficiency gain.

2 Preliminaries and Background

2.1 Notations and Assumptions

Throughout the paper, unless otherwise specified, by $\mathbb{G} := \langle g \rangle$ we denote a cyclic group of prime order Q generated by g. By $\mathtt{H_g}, \mathtt{H_p}, \mathtt{H_s} : \{0,1\}^* \to \{0,1\}^\kappa$ we denote three cryptographic hash functions, which will be used to derive group, p2p, and subgroup keys, respectively, and by $\mathtt{H} : \mathbb{G} \mapsto \{0,1\}^\kappa$ an auxiliary hash function which will be used to slightly modify the computations of the original Burmester-Desmedt protocol. The symbol "|" will be used for the concatenation of bit-strings. Since we are interested in authenticated protocols we will further use a digital signature scheme $\Sigma = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$, which we assume to be existentially unforgeable under chosen message attacks (EUF-CMA). Additionally, we recall the following well-known cryptographic assumption (see e.g. [6]):

Definition 1. Let $\mathbb{G} := \langle g \rangle$ as above and $a, b, c \in_R \mathbb{Z}_Q$. The *Gap Diffie-Hellman (GDH)* problem in $\mathbb{G}$ is hard if the following success probability is negligible:

$$\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) := \max_{\mathcal{A}'} \Big| \Pr_{a,b} \big[\mathcal{A}'^{\mathcal{D}(\cdot)}(g, g^a, g^b) = g^{ab} \big] \Big|,$$

where $\mathcal{D}(\cdot)$ denotes the *Decision Diffie-Hellman (DDH)* oracle, i.e. $\mathcal{D}(g, g^a, g^b, g^c)$ outputs 1 if $c = ab$ and 0 otherwise.

Note that $\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa)$ is computed over all PPT adversaries $\mathcal{A}'$ running within time κ. In other words, the *Computation Diffie-Hellman (CDH)* problem in $\mathbb{G}$ is assumed to be hard even if the DDH problem in $\mathbb{G}$ is easy.

2.2 Burmester-Desmedt Group Key Exchange

The Burmester-Desmedt (BD) protocol from [13] is one of the best known unauthenticated GKE protocols. Its technique has influenced many GKE protocols, including [2, 23]. The BD protocol arranges participants $U_1, \ldots, U_n$ into a cycle, and requires two communication rounds:

Round 1. Each U_i broadcasts $y_i := g^{x_i}$ for some random $x_i \in_R \mathbb{Z}_Q$.
Round 2. Each U_i broadcasts $z_i := (y_{i+1}/y_{i-1})^{x_i}$ (the indices i form a cycle, i.e. $0 = n$ and $n + 1 = 1$).
Group Key Computation. Each U_i computes the secret group element

$$k'_i := (y_{i-1})^{nx_i} \cdot z_i^{n-1} \cdot z_{i+1}^{n-2} \cdots z_{i+n-2} = g^{x_1x_2 + x_2x_3 + \ldots + x_nx_1},$$

which is then used to derive the group key.

As shown by Katz and Yung [22] the group element k'_i remains indistinguishable from a randomly chosen element in $\mathbb{G}$ under the DDH assumption with regard to a passive adversary; using the general authentication technique from [22] this indistinguishability can be extended to resist active attacks.

2.3 Parallel Diffie-Hellman Key Exchange

The following one-round parallelized Diffie-Hellman protocol (PDHKE) with the additional key derivation step based on a hash function $\mathtt{H_p}$ (modeled as a random oracle) has been deployed by Manulis [27] for the computation of independent p2p keys between any two users U_i and U_j:

Round 1. Each U_i chooses a random $x_i \in_R \mathbb{Z}_Q$ and broadcasts $y_i := g^{x_i}$.
P2P Key Computation. Each U_i proceeds as follows:
- for the input identity U_j compute $k'_{i,j} := y_j^{x_i} = g^{x_i x_j}$,
- derive $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i|y_i, U_j|y_j)$.

(W.l.o.g. we assume that both users U_i and U_j use the same order for the inputs to $\mathtt{H_p}$.)

As motivated in [27] the input $(U_i|y_i, U_j|y_j)$ to $\mathtt{H_p}$ in addition to $k'_{i,j} = g^{x_i x_j}$ is necessary in order to ensure the independence of p2p keys in the presence of collusion attacks. This independence follows from the uniqueness of user identities and the negligible probability that an honest user U_i chooses the same exponent x_i in two different sessions. It has been shown in [27] that PDHKE provides security of p2p keys in the presence of passive attacks and that the authentication technique from [22] is also sufficient to preserve this property in the presence of active attacks.

2.4 Insecure Merge of BD and PDHKE

Given BD and PDHKE one of the questions investigated by Manulis [27] was whether a user U_i can safely *re-use* own exponent x_i for the computation of the group key and any p2p key. In other words whether a two-round merged version of original BD and PDHKE would result in a secure GKE+P protocol in which the keys are computed as follows: the group key $k_i := \mathtt{H_g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and any p2p key $k_{i,j} := \mathtt{H_s}(k'_{i,j}, U_i|y_i, U_j|y_j)$.

The analysis given in [27] shows that this merge is insecure. More precisely, for any group size $n \geq 3$ an attack (possibly by colluding participants) was presented that would break the AKE-security of any p2p key $k_{i,j}$.

In the next section we revise this result. We show how to slightly modify the computation of z_i in the original BD protocol in order to allow secure computation of $k_i := \mathtt{H_g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and $k_{i,j} := \mathtt{H_s}(k'_{i,j}, U_i|y_i, U_j|y_j)$. We stress that our changes do not increase the original communication complexity of the BD protocol which is the actual goal for considering its merge with PDHKE. Then, based on our new construction we show how to obtain a secure GKE+S protocol where the communication effort for the derivation of subgroup keys requires only one round; this in contrast to two rounds that would be necessary to establish a subgroup key using the original BD protocol from the scratch.

3 GKE+P Protocol from Modified BD and PDHKE

In order to obtain a secure merge of BD and PDHKE we make use of the following trick in the computation of z_i: Instead of computing $z_i := (y_{i+1}/y_{i-1})^{x_i}$ we let each U_i first compute $k'_{i,i+1} := y_{i+1}^{x_i}$ and $k'_{i-1,i} := y_{i-1}^{x_i}$ and then compute z_i as an XOR sum $\mathtt{H}(k'_{i-1,i}) \oplus \mathtt{H}(k'_{i,i+1})$. This does not introduce new communication costs to the BD protocol but has impact on the computation of the group key. We observe that similar trick has been applied for a different purpose by Kim, Lee, and Lee [23], who

considered possible extensions of BD-like protocols to handle dynamic membership events such as join and leave or to speed up the computation process, whereas here we use the trick exclusively to achieve independence between the group and p2p keys.

The actual description of the protocol which we denote mBD+P follows. Since we are interested in AKE-secure constructions we describe the necessary authentication steps as well. For this we assume that each U_i is in possession of a long-lived key pair $(sk_i, pk_i) \leftarrow \mathtt{KeyGen}(1^\kappa)$ for the EUF-CMA secure digital signature scheme $\Sigma = (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$. The authentication procedure is similar to the general authentication technique from [22], except that we construct the session id using elements from $\mathbb{G}$ as nonces; thus saving the communication costs. The protocol proceeds in two stages: the group stage involves all members of the group and results in the computation of the common group key, whereas the p2p stage is executed on-demand by any two group members wishing to compute an independent p2p key.

Group Stage. Let the group be defined by $\mathtt{pid} = (U_1, \ldots, U_n)$. In the following description we assume that user indices form a cycle such that $U_0 = U_n$ and $U_{n+1} = U_1$.

Round 1. Each U_i computes $y_i := g^{x_i}$ for some random $x_i \in_R \mathbb{Z}_Q$ and broadcasts (U_i, y_i).

Round 2. Each U_i proceeds as follows:

- compute $\mathtt{sid}_i := (U_1|y_1, \ldots, U_n|y_n)$,
- $k'_{i-1,i} := y_{i-1}^{x_i}$ and $k'_{i,i+1} := y_{i+1}^{x_i}$,
- $z'_{i-1,i} := \mathtt{H}(k'_{i-1,i}, \mathtt{sid}_i)$ and $z'_{i,i+1} := \mathtt{H}(k'_{i,i+1}, \mathtt{sid}_i)$,
- $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$,
- $\sigma_i := \mathtt{Sign}(sk_i, (U_i, z_i, \mathtt{sid}_i))$,
- broadcast (U_i, z_i, σ_i).

Group Key Computation. Each U_i checks whether $z_1 \oplus \ldots \oplus z_n = 0$ and whether all received signatures σ_j are valid and aborts if any of these checks fails. Otherwise, U_i proceeds as follows:

- iteratively for each $j = i, \ldots, i+n-1$: $z'_{j,j+1} := z'_{j-1,j} \oplus z_j$,
- accept $k_i := \mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$.

P2P Stage. On input any user identity $U_j \in \mathtt{pid}_i$ the corresponding user U_i proceeds as follows:

- compute $k'_{i,j} := y_j^{x_i} = g^{x_i x_j}$,
- derive $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i|y_i, U_j|y_j)$.

(W.l.o.g. we assume that both users U_i and U_j use the same order for the inputs to $\mathtt{H_p}$.)

Here we observe that the computation of p2p keys proceeds without any interaction.

4 GKE+S Protocol from Modified BD and PDHKE

In this section we extend our previous protocol to a secure GKE+S solution. We call it mBD+S. The design rationale is as follows: Users run the group stage to compute the group key and then any subgroup can on-demand repeat the second round of the protocol re-using the exponents from the group stage. Observe that each subgroup identified by some $\mathtt{spid} \subset \mathtt{pid}$ uniquely determines its own cycle. However, there can be many subgroups involving the same pair of users U_i and U_j in which they are

located at neighboring positions. In order to avoid the use of the same value $z'_{i,j}$ for the derivation of different subgroup keys we include a subgroup session id $\mathtt{ssid}_i$ given as the concatenation of all (U_i, y_i) with $U_i \in \mathtt{spid}$ as additional input to $\mathtt{H}$.

Since the group stage of the current protocol does not change with respect to the protocol in the previous section, we only present below the subgroup stage.

Subgroup Stage. On input any subgroup $\mathtt{spid} \subset \mathtt{pid}$ the corresponding users perform the following steps. For the ease of presentation we assume that $\mathtt{spid} = (U_1, \ldots, U_m)$ with $m < n$ and that $U_0 = U_m$ and $U_{m+1} = U_1$.

Round 1. Each $U_i \in \mathtt{spid}$ proceeds as follows:
- extract $\mathtt{ssid}_i := (U_1|y_1, \ldots, U_m|y_m)$ from $\mathtt{sid}_i$;
- $k'_{i-1,i} := y_{i-1}^{x_i}$ and $k'_{i,i+1} := y_{i+1}^{x_i}$,
- $z'_{i-1,i} := \mathtt{H}(k'_{i-1,i}, \mathtt{ssid}_i)$ and $z'_{i,i+1} := \mathtt{H}(k'_{i,i+1}, \mathtt{ssid}_i)$,
- $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$,
- $\sigma_i := \mathtt{Sign}(sk_i, (U_i, z_i, \mathtt{ssid}_i))$,
- broadcast (U_i, z_i, σ_i).

Subgroup Key Computation. Each U_i checks whether $z_1 \oplus \ldots \oplus z_m = 0$ and whether all received signatures σ_j are valid and aborts if any of these checks fails. Otherwise, U_i proceeds as follows:
- iteratively for each $j = i, \ldots, i+m-1$: $z'_{j,j+1} := z'_{j-1,j} \oplus z_j$,
- accept $k_{i,J} := \mathtt{H_s}(z'_{1,2}, \ldots, z'_{m,1}, \mathtt{ssid}_i)$.

(W.l.o.g. we assume that all users in $\mathtt{spid}$ use the same order for the inputs to $\mathtt{H_s}$.)

Note that for subgroups of size two, i.e. containing only U_i and U_j, both users can still derive their p2p key without executing the subgroup stage simply as $k_{i,j} := \mathtt{H_s}(z'_{i,j}, (U_i|y_i, U_j|y_j))$.

5 Generalized Security Model

Here we generalize the GKE+P model by Manulis [27] towards consideration of subgroup keys computed by participants of a GKE protocol.

5.1 Participants, Sessions, and Correctness of GKE+P Protocols

Let $\mathcal{U}$ denote a set of at most N users (more precisely, their identities which are assumed to be unique) in the universe. We assume that any subset of n users ($2 \leq n \leq N$) can be invoked for a single session of a GKE+S protocol $\mathcal{P}$. Each $U_i \in \mathcal{U}$ holds a (secret) long-lived key LL_i. The participation of U_i in distinct, possibly concurrent protocol sessions is modeled via an unlimited number of *instances* Π_i^s, $s \in \mathbb{N}$. An instance Π_i^s can be invoked for one GKE+S session with some partner id $\mathtt{pid}_i^s \subseteq \mathcal{U}$ encompassing the identities of all the intended participants (including U_i). An execution of a GKE+S protocol is then split in two stages, denoted as *group stage* and *subgroup stage*, described in the following.

The group stage results in Π_i^s holding a *session id* $\mathtt{sid}_i^s$ which uniquely identifies the current protocol session. Any two instances Π_i^s and Π_j^t are considered as being *partnered* if $\mathtt{sid}_i^s = \mathtt{sid}_j^t$ and $\mathtt{pid}_i^s = \mathtt{pid}_j^t$. The success of the group stage for some instance Π_i^s is modeled through its *acceptance* with some *session group key* k_i^s.

Each instance Π_i^s that has accepted in the group stage can later be invoked for the subgroup stage on input some *subgroup partner id* $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$ (which includes

U_i). This invocation can be performed several times for different subgroups of $\mathtt{pid}_i^s$. The success of the subgroup stage for some Π_i^s is modeled through its *acceptance* with some *session subgroup key* $k_{i,J}^s$, whereby J denotes the set of indices of users in $\mathtt{spid}_i^s$, i.e. it includes all j with $U_j \in \mathtt{spid}_i^s$.

Definition 2 (GKE+S/GKE+P Protocols). $\mathcal{P}$ is a *group key exchange protocol enabling on-demand derivation of subgroup keys (GKE+S)* if $\mathcal{P}$ consists of the following two protocols/algorithms:

$\mathcal{P}$.GKE$(U_1, \ldots, U_n)$**:** For each U_i, a new instance Π_i^s with $\mathtt{pid}_i^s = (U_1, \ldots, U_n)$ is created and a probabilistic interactive protocol between these instances is executed such that at the end every instance Π_i^s accepts holding the session group key k_i^s. This protocol defines the *group stage*.

$\mathcal{P}$.SKE$(\Pi_i^s, \mathtt{spid}_i^s)$**:** On input an accepted instance Π_i^s and a subgroup partner id $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$ this deterministic (possibly interactive) algorithm outputs the session subgroup key $k_{i,J}^s$. This algorithm defines the *subgroup stage*. (We assume that SKE is given only for groups of size $n \geq 3$ since for $n = 2$ the group key is sufficient.)

A GKE+S protocol $\mathcal{P}$ is *correct* if (when no adversary is present) all instances invoked for the group stage $\mathcal{P}$.GKE accept with identical group keys and for all instances Π_j^t partnered with Π_i^s the subgroup stage results in $\mathcal{P}$.SKE$(\Pi_j^t, \mathtt{spid}_j^t) = \mathcal{P}$.SKE$(\Pi_i^s, \mathtt{spid}_i^s)$ if $\mathtt{spid}_j^t = \mathtt{spid}_i^s$.

$\mathcal{P}$ is a *group key exchange protocol enabling on-demand derivation of p2p keys (GKE+P)* if it is a GKE+S protocol in which the only admissible input to SKE is of the form $\mathtt{spid}_i^s = (U_i, U_j)$. The execution of SKE in this case defines the *p2p stage*.

5.2 Adversarial Model and Security Goals

Security of GKE+S protocols must ensure independence of the session group key k_i^s and any subgroup key $k_{i,J}^s$. In particular the secrecy of k_i^s must hold even if any subgroup key $k_{i,J}^s$ is leaked to the adversary. Similarly, the leakage of the group key k_i^s must guarantee the secrecy of any subgroup key $k_{i,J}^s$. Since the computation of subgroup keys is triggered on-demand we must provide the adversary with the ability to schedule the execution of $\mathcal{P}$.SKE on the subgroups of its choice.

Additionally, GKE+S protocols must ensure independence amongst different subgroup keys. That is the knowledge of some $k_{i,J}^s$ should not reveal information about any other subgroup key computed by Π_i^s or any partner Π_j^t. This requirement implicitly assumes that some participants U_j with $j \notin J$ may collude during the protocol execution.

Finally, the described independence amongst the group and subgroup keys must hold across different sessions of GKE+S.

Adversarial Model. The adversary $\mathcal{A}$, modeled as a PPT machine, can schedule the protocol execution and mount attacks through a set of queries:

- *Execute*$(U_1, \ldots, U_n)$: This query executes the group stage protocol between new instances of $U_1, \ldots, U_n \in \mathcal{U}$ and provides $\mathcal{A}$ with the execution transcript.
- *Send*(Π_i^s, m) : With this query $\mathcal{A}$ can deliver a message m to Π_i^s whereby U denotes the identity of its sender. $\mathcal{A}$ is then given the protocol message generated by Π_i^s in response to m (the output may also be empty if m is unexpected or if

Π_i^s accepts). A special invocation query of the form $Send(U_i, ('start', U_1, \ldots, U_n))$ creates a new instance Π_i^s with $\mathtt{pid}_i^s := (U_1, \ldots, U_n)$ and provides $\mathcal{A}$ with the first protocol message. This query can be used by $\mathcal{A}$ also during the subgroup stage if $\mathcal{P}$.SKE requires interaction.

- $SKE(\Pi_i^s, \mathtt{spid}_i^s)$: This query allows $\mathcal{A}$ to schedule the on-demand computation of subgroup keys. If $\mathcal{P}$.SKE requires interaction then Π_i^s returns $\mathcal{A}$ its first message for the subgroup stage; otherwise Π_i^s computes the subgroup key $k_{i,J}^s$. This query is processed only if Π_i^s has accepted and $\mathtt{spid}_i^s \subset \mathtt{pid}_i^s$. Additionally, this query can be asked only once per input $(\Pi_i^s, \mathtt{spid}_i^s)$.
- $RevealGK(\Pi_i^s)$: This query models the leakage of group keys and provides $\mathcal{A}$ with k_i^s. It is answered only if Π_i^s has accepted in the group stage.
- $RevealSK(\Pi_i^s, \mathtt{spid}_i^s)$: This query models the leakage of subgroup keys and provides $\mathcal{A}$ with the corresponding $k_{i,J}^s$; the query is answered only if the algorithm $SKE(\Pi_i^s, \mathtt{spid}_i^s)$ has already been invoked and the subgroup key computed.
- $Corrupt(U_i)$: This query provides $\mathcal{A}$ with LL_i. Note that in this case $\mathcal{A}$ does not gain control over the user's behavior, but might be able to communicate on behalf of the user.
- $TestGK(\Pi_i^s)$: This query models indistinguishability of session group keys. Depending on a given (privately flipped) bit b $\mathcal{A}$ is given, if $b = 0$ a random session group key, and if $b = 1$ the real k_i^s. This query can be asked only once and is answered only if Π_i^s has accepted in the group stage.
- $TestSK(\Pi_i^s, \mathtt{spid}_i^s)$: This query models indistinguishability of session subgroup keys. Depending on a given (privately flipped) bit b $\mathcal{A}$ is given, if $b = 0$ a random session p2p key, and if $b = 1$ the real $k_{i,J}^s$. This query is answered only if the algorithm $SKE(\Pi_i^s, \mathtt{spid}_i^s)$ has already been invoked and the subgroup key computed.

Terminology. We say that U is *honest* if no $Corrupt(U)$ has been asked by $\mathcal{A}$; otherwise, U is *corrupted* (or *malicious*). This also refers to the instances of U.

Two Notions of Freshness. The classical notion of freshness imposes several conditions in order to prevent any trivial break of the AKE-security. Obviously, we need two definitions of freshness to capture such conditions for the both key types.

First, we define the notion of *instance freshness* which will be used in the definition of AKE-security of group keys. Our definition is essentially the one given in [22].

Definition 3 (Instance Freshness). An instance Π_i^s is *fresh* if Π_i^s has accepted in the group stage and none of the following is true, whereby Π_j^t denotes an instance partnered with Π_i^s: (1) $RevealGK(\Pi_i^s)$ or $RevealGK(\Pi_j^t)$ has been asked, or (2) $Corrupt(U')$ for some $U' \in \mathtt{pid}_i^s$ was asked before any $Send(\Pi_i^s, \cdot)$.

Note that in the context of GKE+S the above definition restricts $\mathcal{A}$ from active participation on behalf of any user during the attacked session, but implicitly allows for the leakage of (all) subgroup keys.

Additionally, we define the notion of *instance-subgroup freshness* which will be used to specify the AKE-security of subgroup keys.

Definition 4 (Instance-Subgroup Freshness). An instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$ is *fresh* if Π_i^s has accepted in the group stage and none of the following is true, whereby Π_j^t is assumed to be partnered with Π_i^s and $\mathtt{spid}_j^t = \mathtt{spid}_i^s$: (1) $RevealSK(\Pi_i^s, \mathtt{spid}_i^s)$

or $\mathit{RevealSK}(\Pi_j^t, \mathtt{spid}_j^t)$ has been asked, or (2) $\mathit{Corrupt}(U_i)$ or $\mathit{Corrupt}(U_j)$ was asked before any $\mathit{Send}(\Pi_i^s, \cdot)$ or $\mathit{Send}(\Pi_j^s, \cdot)$.

Here $\mathcal{A}$ is explicitly allowed to actively participate in the attacked session on behalf of any user except for U_i and any $U_j \in \mathtt{spid}_i^s$. Also $\mathcal{A}$ may learn the group key k_i and all subgroup keys except for $k_{i,J}$ returned by $\mathcal{P}.\mathrm{SKE}(\Pi_i^s, \mathtt{spid}_i^s)$. This models possible collusion of participants from $\mathtt{pid}_i^s \setminus \mathtt{spid}_i^s$ during the execution of the protocol aiming to break the secrecy of the subgroup key $k_{i,J}$.

AKE-Security of Group and Subgroup Keys. For the AKE-security of group keys we follow the definition from [22].

Definition 5 (AKE-Security of Group Keys). Let $\mathcal{P}$ be a correct GKE+P protocol and b a uniformly chosen bit. By $\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g},b}(\kappa)$ we define the following adversarial game, which involves a PPT adversary $\mathcal{A}$ that is given access to all queries:

- $\mathcal{A}$ interacts via queries;
- at some point $\mathcal{A}$ asks a $\mathit{TestGK}(\Pi_i^s)$ query for some instance Π_i^s which is (and remains) fresh;
- $\mathcal{A}$ continues interacting via queries;
- when $\mathcal{A}$ terminates, it outputs a bit, which is set as the output of the game.

We define

$$\mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g}}(\kappa) := \left| 2\Pr[\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}g},b}(\kappa) = b] - 1 \right|$$

and denote with $\mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake\text{-}g}}(\kappa)$ the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that $\mathcal{P}$ provides *AKE-security of group keys* if this advantage is negligible.

Finally, we define AKE-security of subgroup keys while considering possible collusion attacks, as above but with a $\mathit{TestSK}(\Pi_i^s, \mathtt{spid}_i^s)$ query.

Definition 6 (AKE-security of Subgroup Keys). Let $\mathcal{P}$ be a correct GKE+S protocol and b a uniformly chosen bit. By $\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}s},b}(\kappa)$ we define the following adversarial game, which involves a PPT adversary $\mathcal{A}$ that is given access to all queries:

- $\mathcal{A}$ interacts via queries;
- at some point $\mathcal{A}$ asks a $\mathit{TestSK}(\Pi_i^s, \mathtt{spid}_i^s)$ query for some instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$ which is (and remains) fresh;
- $\mathcal{A}$ continues interacting via queries;
- when $\mathcal{A}$ terminates, it outputs a bit, which is set as the output of the game.

We define

$$\mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}s}}(\kappa) := \left| 2\Pr[\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{ake\text{-}s},b}(\kappa) = b] - 1 \right|$$

and denote with $\mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake\text{-}s}}(\kappa)$ the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that $\mathcal{P}$ provides *AKE-security of subgroup keys* if this advantage is negligible.

6 Security of Our GKE+P and GKE+S Protocols

In this section we analyze security of our mBD+P and mBD+S protocols using our generalized security model. The corresponding proofs of our theorems can be found in appendix. The security results hold in the random oracle model. The following two theorems show that mBD+P is a secure GKE+P protocol.

Theorem 7. *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+P *provides AKE-security of group keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+P}}(\kappa) \leq & \frac{2N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H}_g}+\mathtt{q}_{\mathtt{H}_p})^2}{2^{\kappa-1}} \\ & + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathtt{q}_{\mathtt{Se}}\left(N\mathtt{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathtt{q}_{\mathtt{H}_g}}{2^{\kappa}}\right)\end{aligned}$$

with at most $(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries and at most* $\mathtt{q}_{\mathtt{H}_g}$, $\mathtt{q}_{\mathtt{H}_p}$, *and* $\mathtt{q}_{\mathtt{H}}$ *random oracle queries being asked.*

Theorem 8. *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+P *provides AKE-security of p2p keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}p}}_{\mathrm{mBD+P}}(\kappa) \leq & \frac{2N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H}_g}+\mathtt{q}_{\mathtt{H}_p})^2}{2^{\kappa-1}} \\ & + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathtt{q}_{\mathtt{Se}}\left((N\mathtt{q}_{\mathtt{H}}+\mathtt{q}_{\mathtt{SKE}}\mathtt{q}_{\mathtt{H}_p})\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa)\right)\end{aligned}$$

with at most $(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries, at most* $\mathtt{q}_{\mathtt{SKE}}$ *p2p keys being computed via SKE queries, and at most* $\mathtt{q}_{\mathtt{H}_g}$, $\mathtt{q}_{\mathtt{H}_p}$, *and* $\mathtt{q}_{\mathtt{H}}$ *random oracle queries being asked.*

Our next two theorems prove that mBD+S is a secure GKE+S protocol. The main difference in the security analysis is the consideration of the additional communication round for the subgroup stage (Theorem 10). Since the group stage of mBD+S does not differ from that of mBD+P, the proof of Theorem 9 follows from that of Theorem 7.

Theorem 9. *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+S *provides AKE-security of group keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+S}}(\kappa) \leq & \frac{2N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H}_g}+\mathtt{q}_{\mathtt{H}_s})^2}{2^{\kappa-1}} \\ & + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) + 2\mathtt{q}_{\mathtt{Se}}\left(N\mathtt{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathtt{q}_{\mathtt{H}_g}}{2^{\kappa}}\right)\end{aligned}$$

with at most $(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries and at most* $\mathtt{q}_{\mathtt{H}_g}$, $\mathtt{q}_{\mathtt{H}_s}$, *and* $\mathtt{q}_{\mathtt{H}}$ *random oracle queries being asked.*

Theorem 10. *If the GDH problem is hard in* $\mathbb{G}$ *then our protocol* mBD+S *provides AKE-security of subgroup keys and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}s}}_{\mathrm{mBD+S}}(\kappa) \leq & \frac{2N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H}_g}+\mathtt{q}_{\mathtt{H}_s})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) \\ & + 2\mathtt{q}_{\mathtt{Se}}\left(\big(N+(N-1)\mathtt{q}_{\mathtt{SKE}}\big)\mathtt{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathtt{q}_{\mathtt{SKE}}\mathtt{q}_{\mathtt{H}_s}}{2^{\kappa}}\right)\end{aligned}$$

with at most $(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})$ *sessions being invoked via Execute and Send queries, at most* $\mathtt{q}_{\mathtt{SKE}}$ *subgroup stages being invoked via SKE queries, and at most* $\mathtt{q}_{\mathtt{H}_g}$, $\mathtt{q}_{\mathtt{H}_s}$, *and* $\mathtt{q}_{\mathtt{H}}$ *random oracle queries being asked.*

7 Performance Comparison

In Table 1 we compare the complexity of our protocols. We measure the communication costs as a total number of transmitted elements in $\mathbb{G}$, and computation costs as a number of modular exponentiations *per* U_i (in the case of BD we count only exponentiations with x_i assuming that $|x_i| \gg n$). We omit signature generation and verification costs, which are equal for all considered protocols.

Table 1. Performance Comparison of GKE+P and GKE+S Protocols

GKE+P/S Protocols	Rounds	Communication (in $\log Q$ bits)	Computation (in mod. exp. per U_i)
GKE+P BD [27]	2	$3n$	3
GKE+P KPT [27]	2	$2n-2$	$n+2-i$ ($2n-2$ for U_1)
our mBD+P	2	$2n$	3
GKE+S BD	2	$2m$	2
our mBD+S	1	m	≤ 2

The first part of the table is devoted to GKE+P protocols. By GKE+P BD we denote the protocol from [27] in which the original Burmester-Desmedt protocol is executed in parallel with PDHKE, i.e. each user U_i uses two independent exponents x_i and $\bar{x}_i$ for the computation of the group key and any p2p key, respectively. By GKE+P KPT we denote the tree-based Kim-Perrig-Tsudik protocol from [24] in which each user U_i holds only one exponent x_i and uses it to compute both types of keys, which has been proven secure in [27]. Since all GKE+P protocols apply the PDHKE technique for the derivation of p2p keys we also exclude the computation costs needed to compute a Diffie-Hellman secret $k'_{i,j}$ that requires constantly one exponentiation per each U_j. Then, we observe that in comparison to GKE+P BD our mBD+P protocol has better communication complexity since it requires each U_i to hold only one exponent x_i. Note also that the PDHKE-KPT protocol has asymmetric costs, depending on the position of U_i in the sequence $U_1, \ldots, U_n$. Although this asymmetry may have benefits in groups with heterogeneous devices we remark that in general this protocol has much worse computation complexity.

The second part of the table compares the effort needed to derive a subgroup key for any subgroup of size $m < n$. By GKE+S BD we consider the trivial solution in which the original BD protocol from [13] is executed for each new subgroup. We compare it to the subgroup stage of our mBD+S protocol, which requires only one communication round per subgroup. Observe that the group stage of mBD+S is executed only once and its complexity is identical to the complete execution of the BD protocol. By ≤ 2 in the computation costs of mBD+S we point out that users can re-use the intermediate values $k'_{i-1,i}$ and $k'_{i,i+1}$ possibly computed during the group stage or during the subgroup stage for another subgroup, provided the relative position of U_{i-1} and U_i or of U_i and U_{i+1} in the cyclic order of the new subgroup remains the same. In this way our mBD+S protocol also offers a trade-off between space and computation complexity depending on whether users wish to cache the intermediate values that re-occur in different protocol stages.

8 Conclusion

The increasing popularity of multi-user communication systems offering various forms of communication can be secured using flexible GKE protocols that provide more than just a secret group key for the initial set of their participants. This paper addressed the extension of this basic functionality of GKE protocols towards the computation of different types of keys allowing a secure mix of group, subgroup, and peer-to-peer communication. While our mBD+P protocol with improved complexity compared to the protocols from [27] allows a secure mix of group and peer-to-peer communication our mBD+S protocol extends this approach to obtain the additional secure communication within any possible subgroup.

Acknowledgments

This work was supported in part by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II, by the French ANR-07-SESU-008-01 PAMPA Project, and through the Center for Advanced Security Research Darmstadt (`www.cased.de`).

References

1. M. Abdalla, J.-M. Bohli, M. I. G. Vasco, and R. Steinwandt. (Password) Authenticated Key Establishment: From 2-Party to Group. In *TCC 2007*, LNCS 4392, pp. 499–514. 2007.
2. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-Based Group Key Exchange in a Constant Number of Rounds. In *PKC 2006*, LNCS 3958, pp. 427–442. 2006.
3. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO 1993*, LNCS 773, pp. 232–249. Springer, 1994.
4. G. P. Biswas. Diffie-Hellman Technique: Extended to Multiple Two-Party Keys and One Multi-Party Key. *IET Inf. Sec.*, 2(1):12–18, 2008.
5. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment.* Springer, 2003.
6. A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003*, LNCS 2567, pp. 31–46. Springer, 2003.
7. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *EUROCRYPT 2002*, LNCS 2332, pp. 321–336. Springer, 2002.
8. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *ACM CCS'01*, pp. 255–264. ACM, 2001.
9. E. Bresson and M. Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *ATC 2007*, LNCS 4610, pp. 395–409. 2007.
10. E. Bresson and M. Manulis. Contributory Group Key Exchange in the Presence of Malicious Participants. *IET Inf. Sec.*, 2(3):85–93, 2008.
11. E. Bresson and M. Manulis. Securing Group Key Exchange against Strong Corruptions. In *ACM ASIACCS'08*, pp. 249–260. ACM Press, 2008.
12. E. Bresson, M. Manulis, and J. Schwenk. On Security Models and Compilers for Group Key Exchange Protocols. In *IWSEC 2007*, LNCS 4752, pp. 292–307. Springer, 2007.
13. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT 1994*, LNCS 950, pp. 275–286. Springer, 1994.
14. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT 2001*, LNCS 2045, pp. 453–474. Springer, 2001.
15. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In *ASIACRYPT 2005*, LNCS 3788, pp. 585–604. Springer, 2005.
16. Y. Desmedt and T. Lange. Revisiting Pairing Based Group Key Exchange. In *FC 2008*, LNCS 5143, pp. 53–68. Springer, 2008.
17. W. Diffie and M. E. Hellman. New Directions in Cryptography *IEEE Tran. on Inf. Th.*, 22(6):644–654, 1976.
18. M. C. Gorantla, C. Boyd, and J. M. González Nieto Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In *PKC 2009*, LNCS 5443, pp. 105–123. Springer, 2009.

19. I. Ingemarsson, D. T. Tang, and C. K. Wong. A Conference Key Distribution System. *IEEE Tran. on Inf. Th.*, 28(5):714–719, 1982.
20. I. R. Jeong and D. H. Lee. Parallel Key Exchange. *J. of Univ. Comp. Sci.*, 14(3):377–396, 2008.
21. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *ACM CCS'05*, pp. 180–189. ACM Press, 2005.
22. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO 2003*, LNCS 2729, pp. 110–125. Springer, 2003.
23. H.-J. Kim, S.-M. Lee, and D. H. Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In *ASIACRYPT 2004*, LNCS 3329, pp. 245–259, Springer, 2004.
24. Y. Kim, A. Perrig, and G. Tsudik. Group Key Agreement Efficient in Communication. *IEEE Tran. on Comp.*, 53(7):905–921, 2004.
25. Y. Kim, A. Perrig, and G. Tsudik. Tree-Based Group Key Agreement. *ACM Trans. on Inf. and Syst. Sec.*, 7(1):60–96, 2004.
26. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007*, LNCS 4784, pp. 1–16. Springer, 2007.
27. M. Manulis. Group Key Exchange Enabling On-Demand Derivation of Peer-to-Peer Keys. In *ACNS 2009*, LNCS 5536, pp. 1–19. Springer, 2009.
28. M. Manulis. Security-Focused Survey on Group Key Exchange Protocols. *Cryptology ePrint Archive*, Report 2006/395, 2006.
29. A. Mayer and M. Yung. Secure Protocol Transformation via "Expansion": From Two-Party to Groups. In *ACM CCS '99*, pp. 83–92. ACM Press, 1999.
30. J. Nam, J. Paik, U.-M. Kim, and D. Won. Constant-Round Authenticated Group Key Exchange with Logarithmic Computation Complexity. In *ACNS 2007*, LNCS 4521, pp. 158–176. Springer, 2007.
31. D. G. Steer, L. Strawczynski, W. Diffie, and M. J. Wiener. A Secure Audio Teleconference System. In *CRYPTO 1988*, LNCS 403, pp. 520–528. Springer, 1990.
32. M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *ACM CCS'96*, pp. 31–37. ACM Press, 1996.
33. S. Wu and Y. Zhu. Constant-Round Password-Based Authenticated Key Exchange Protocol for Dynamic Groups. In *FC 2008*, LNCS 5143, pp. 69–82. Springer, 2008.

A Proof of Theorem 1

In the following we show that the advantage of $\mathcal{A}$ in distinguishing k_i from some random element from $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_4$ and denote by $\mathsf{Win}_i^{\mathsf{ake\text{-}g}}$ the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in the i-th game. Recall that in each game $\mathit{TestGK}(\Pi_i^s)$ is answered only if the instance Π_i^s is fresh.

Game $\mathbf{G}_0$. This is the real execution of mBD+P in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances as defined in $\mathsf{Game}^{\mathsf{ake\text{-}g},b}_{\mathcal{A},\mathrm{mBD+P}}(\kappa)$.

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}$, $\mathtt{H_g}$, and $\mathtt{H_p}$, which are modeled as random oracles in the classical way, i.e., by returning new random values for new queries and replaying answers if the queries were previously made.

Game $\mathbf{G}_1$. In this game we exclude for every honest user U_i the collisions of the transcripts (U_i, y_i) and group keys k_i computed in different sessions. We also exclude any collisions between k_i and any p2p key $k_{i,j}$. Regarding the transcripts we observe that if U_i is honest then its session value y_i is randomly distributed in $\mathbb{G}$ (as a result of $y_i := g^{x_i}$ for $x_i \in_R \mathbb{Z}_Q$). Thus, according to the birthday paradox the collision on transcripts occurs with the probability of at most $N(\mathtt{q_{Ex}} + \mathtt{q_{Se}})^2/Q$ over all possible users (recall that sessions can be invoked via *Execute* and *Send* queries). The uniqueness of transcripts also implies the uniqueness of inputs to $\mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$. By construction inputs to $\mathtt{H_g}$ remain always different from the inputs to $\mathtt{H_p}$. Since $\mathtt{H_g}$ and $\mathtt{H_p}$ are modeled as random functions we can also apply the birthday paradox and

upper-bound the probability of collisions for k_i and collisions between k_i and any $k_{i,j}$ by $(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2/2^\kappa$. Thus,

$$|\Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}g}}]| \leq \frac{N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^\kappa}.$$

Game $\mathbf{G}_2$. In this game we assume that Δ fails and bit b' is set at random if $\mathcal{A}$ queries *Send* containing (U_i, z_i, σ_i) with σ_i being a valid signature that has not been previously output by an uncorrupted oracle Π_i^s. In other words the simulation aborts if $\mathcal{A}$ outputs a successful forgery. Following the classical reductionist argument (see for instance [8]) we can build a forger against the signature scheme and upper-bound the probability difference

$$|\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}g}}]| \leq N\mathsf{Succ}_\Sigma^{\mathsf{euf\text{-}cma}}(\kappa).$$

Game $\mathbf{G}_3$. In this game we let Δ guess a value $q^* \in [1, \mathsf{q}_{\mathtt{Se}}]$ and abort if the *TestGK* query is not asked for the session invoked by the q^*-th query. Let Q be the event that this guess is correct and $\Pr[\mathsf{Q}] = 1/\mathsf{q}_{\mathtt{Se}}$. Thus,

$$\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}] = \Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}]\frac{1}{\mathsf{q}_{\mathtt{Se}}} + \frac{1}{2}\left(1 - \frac{1}{\mathsf{q}_{\mathtt{Se}}}\right).$$

This implies,

$$\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}g}}] = \mathsf{q}_{\mathtt{Se}}\left(\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

Game $\mathbf{G}_4$. In this game we assume that Δ is given access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and computes in the simulation of the q^*-th session for each pair of consecutive users (U_i, U_{i+1}) in the cycle $z'_{i,i+1} = \mathtt{H}'(i, i+1)$. Clearly both games remain indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1})$ for any $i \in [1, n]$. However, this query can be used to break the GDH problem in $\mathbb{G}$, i.e. Δ embeds g^a and g^b from the challenge of the GDH problem into the corresponding values g^{x_i} and $g^{x_{i+1}}$, and uses the access to the DDH oracle $\mathcal{D}$ in order to identify $k'_{i,i+1} = g^{x_i x_{i+1}}$ provided by $\mathcal{A}$ as input to $\mathtt{H}$. Therefore,

$$|\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}g}}] - \Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}g}}]| \leq N\mathsf{q}_{\mathtt{H}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa).$$

As a result of this game the group key k_i computed in the q^*-th session is the output of $\mathtt{H}_\mathtt{g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ where all input values $z'_{1,2}, \ldots, z'_{n,1}$ are uniform in $\{0,1\}^\kappa$. Since $\mathtt{H}_\mathtt{g}$ is modeled as a random oracle and collisions on group keys computed in two different sessions have been excluded in Game $\mathbf{G}_1$ the probability that $\mathcal{A}$ wins without querying $\mathtt{H}_\mathtt{g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ is $1/2$; on the other hand, the probability that $\mathcal{A}$ asks such a query is given by $\mathsf{q}_{\mathtt{H}_\mathtt{g}}/2^{n\kappa} < \mathsf{q}_{\mathtt{H}_\mathtt{g}}/2^\kappa$ (for the guess of $z'_{1,2}, \ldots, z'_{n,1}$). Hence, $\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}g}}] \leq 1/2 + \mathsf{q}_{\mathtt{H}_\mathtt{g}}/2^\kappa$.

Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}_{\mathrm{mBD+P}}^{\mathsf{ake\text{-}g}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}g}}] - 1| \\ &\leq \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}_\Sigma^{\mathsf{euf\text{-}cma}}(\kappa) \\ &\quad + 2\mathsf{q}_{\mathtt{Se}}\left(N\mathsf{q}_{\mathtt{H}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa) + \frac{\mathsf{q}_{\mathtt{H}_\mathtt{g}}}{2^\kappa}\right).\end{aligned}$$

□

B Proof of Theorem 2

In the following we show that the advantage of $\mathcal{A}$ in distinguishing any $k_{i,j}$ computed by uncorrupted users U_i and U_j from some random element in $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_6$ and denote by $\mathsf{Win}_i^{\mathsf{ake\text{-}g}}$ the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in the i-th game. Recall that in each game $\mathit{TestSK}(\Pi_i^s, \mathtt{spid}_i^s)$ is answered only if the instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$ is fresh.

Game $\mathbf{G}_0$. This is the real execution of mBD+P in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances as defined in $\mathsf{Game}^{\mathsf{ake\text{-}p},b}_{\mathcal{A},\mathrm{mBD+P}}(\kappa)$.

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}$, $\mathtt{H}_\mathtt{g}$, and $\mathtt{H}_\mathtt{p}$, which are modeled as random oracles.

Game $\mathbf{G}_1$. In this game we exclude for every honest user U_i the collisions of the transcripts (U_i, y_i) and group keys k_i. We also exclude collisions between k_i and any p2p key $k_{i,j}$. Additionally, we exclude collisions amongst p2p keys $k_{i,j}$ computed by U_i for different (possibly malicious) U_j. Since sessions can be invoked via *Execute* and *Send* queries we obtain for the collision on transcripts (according to the birthday paradox) the probability of at most $N(\mathtt{q}_\mathtt{Ex} + \mathtt{q}_\mathtt{Se})^2/Q$ over all possible users. Collisions amongst k_i and collisions between k_i and any $k_{i,j}$ can be upper-bounded by $(\mathtt{q}_{\mathtt{H}_\mathtt{g}} + \mathtt{q}_{\mathtt{H}_\mathtt{p}})^2/2^\kappa$ using the same argument as in Game $\mathbf{G}_1$ from the proof of Theorem 7. This upper-bound also includes collisions amongst $k_{i,j}$ computed by the same honest U_i for different U_j. To see this observe that even if U_j malicious and chooses own y_j in some rogue way (e.g. in relation to other values) the input to $\mathtt{H}_\mathtt{p}(k'_{i,j}, U_i|y_i, U_j|y_j)$ still remains unique due to the uniqueness of the honest user's transcript (U_i, y_i) and identity U_j. Hence,

$$|\Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}p}}] - \Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}p}}]| \leq \frac{N(\mathtt{q}_\mathtt{Ex} + \mathtt{q}_\mathtt{Se})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H}_\mathtt{g}} + \mathtt{q}_{\mathtt{H}_\mathtt{p}})^2}{2^\kappa}.$$

Game $\mathbf{G}_2$. In this game we abort simulation (setting bit b' at random) if $\mathcal{A}$ queries *Send* containing (U_i, z_i, σ_i) with σ_i being a valid signature that has not been previously output by an oracle of uncorrupted U_i. In other words the simulation fails if $\mathcal{A}$ outputs a successful forgery such that

$$|\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}p}}] - \Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}p}}]| \leq N\mathsf{Succ}_\Sigma^{\mathsf{euf\text{-}cma}}(\kappa).$$

Game $\mathbf{G}_3$. In this game Δ randomly chooses a session $q^* \in_R [1, \mathtt{q}_\mathtt{Se}]$ and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ asks the *TestSK* query to some oracle Π_i^s that accepted in a group stage of another session, i.e. not in the q^*-th session. Using similar argument as in Game $\mathbf{G}_3$ from the proof of Theorem 7 we obtain

$$\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}p}}] = \mathtt{q}_\mathtt{Se}\left(\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}p}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

As a result of this game it is sufficient for the simulator to focus on the q^*-th session and to simulate instances of users participating in the group and p2p stages of that session.

Game $\mathbf{G}_4$. In this game Δ is given access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and in the simulation of the q^*-th session for each pair of consecutive users (U_i, U_{i+1}) in the cycle Δ computes $z'_{i,i+1} = \mathtt{H}'(i, i+1)$ if both users U_i and U_{i+1}

are uncorrupted. Note that the definition of AKE-security for p2p keys requires the *TestSK* query to be asked to a fresh instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$. In case of p2p keys $\mathtt{spid}_i^s$ would contain two users U_i and U_j that must be uncorrupted at the moment when their p2p key is computed. However, $\mathcal{A}$ is allowed to corrupt and act on behalf of other users during the protocol execution. This replacement remains indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1})$, in which case we can solve the challenge of the GDH problem (similar to the proof of the previous theorem). Thus,

$$|\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}p}}] - \Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}p}}]| \leq N\mathsf{q}_{\mathtt{H}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa).$$

This game implies that in the q^*-th session the computation of the group key $k_i = \mathtt{H}_{\mathtt{g}}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ uses values $z'_{i,i+1}$ that are uniform in $\{0,1\}^\kappa$ if the respective users U_i and U_{i+1} were uncorrupted at the moment when $z'_{i,i+1}$ was computed.

Game $\mathbf{G}_5$. In this game Δ randomly chooses a value $q^{**} \in_R [1, \mathsf{q}_{\mathtt{SKE}}]$ and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ asks its q^{**}-th query $SKE(\Pi_i^s, \mathtt{spid}_i^s)$ without asking the $TestSK(\Pi_i^s, \mathtt{spid}_i^s)$ query later. I.e. Δ tries to guess which invocation in the p2p stage of the q^*-th session will lead to the computation of the target p2p key. Then,

$$\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}p}}] = \mathsf{q}_{\mathtt{SKE}}\left(\Pr[\mathsf{Win}_5^{\mathsf{ake\text{-}p}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

Note that if the guess is positive then $\mathtt{spid}_i^s$ contains two uncorrupted users U_i and U_j.

Game $\mathbf{G}_6$. In this game we assume that Δ is given access to the private random oracle $\mathtt{H}'_{\mathtt{p}} : \{0,1\}^* \to \{0,1\}^\kappa$ and in response to the q^{**}-th $SKE(\Pi_i^s, \mathtt{spid}_i^s)$ query guessed above Δ computes the corresponding p2p key $k_{i,j}$ as $\mathtt{H}'_{\mathtt{p}}(U_i|y_i, U_j|y_j)$ that is independent of $k'_{i,j}$. Again, this replacement remains indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}_{\mathtt{p}}(k'_{i,i+1}, U_i|y_i, U_j|y_j)$, in which case reduction to the GDH problem becomes possible, i.e.

$$|\Pr[\mathsf{Win}_6^{\mathsf{ake\text{-}p}}] - \Pr[\mathsf{Win}_5^{\mathsf{ake\text{-}p}}]| \leq \mathsf{q}_{\mathtt{H}_{\mathtt{p}}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa).$$

As a consequence the p2p key $k_{i,j}$ is uniform in $\{0,1\}^\kappa$ and, thus $\Pr[\mathsf{Win}_6^{\mathsf{ake\text{-}p}}] \leq 1/2$. Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}_{\mathrm{mBD+P}}^{\mathsf{ake\text{-}p}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}p}}] - 1| \\ &\leq \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_{\mathtt{g}}} + \mathsf{q}_{\mathtt{H}_{\mathtt{p}}})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\kappa) \\ &\quad + 2\mathsf{q}_{\mathtt{Se}}\left((N\mathsf{q}_{\mathtt{H}} + \mathsf{q}_{\mathtt{SKE}}\mathsf{q}_{\mathtt{H}_{\mathtt{p}}})\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa)\right).\end{aligned}$$

□

C Proof of Theorem 3

In the following we show that the advantage of $\mathcal{A}$ in distinguishing k_i from some random element from $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_4$ and denote by $\mathsf{Win}_i^{\mathsf{ake\text{-}g}}$ the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in the i-th game. Recall that in each game $TestGK(\Pi_i^s)$ is answered only if the instance Π_i^s is fresh.

Game $\mathbf{G}_0$. This is the real execution of mBD+S in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances as defined in $\mathsf{Game}^{\mathsf{ake\text{-}g},b}_{\mathcal{A},\mathrm{mBD+S}}(\kappa)$.

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}$, $\mathtt{H}_\mathtt{g}$, and $\mathtt{H}_\mathtt{s}$, which are modeled as random oracles in the classical way, i.e., by returning new random values for new queries and replaying answers if the queries were previously made.

Game $\mathbf{G}_1$. In this game we exclude for every honest user U_i the collisions of the transcripts (U_i, y_i) and group keys k_i computed in different sessions. We also exclude any collisions between k_i and any subgroup key $k_{i,J}$. Regarding the transcripts we observe that if U_i is honest then its session value y_i is randomly distributed in $\mathbb{G}$ (as a result of $y_i := g^{x_i}$ for $x_i \in_R \mathbb{Z}_Q$). Thus, according to the birthday paradox the collision on transcripts occurs with the probability of at most $N(\mathsf{q}_\mathtt{Ex} + \mathsf{q}_\mathtt{Se})^2/Q$ over all possible users (recall that sessions can be invoked via *Execute* and *Send* queries). The uniqueness of transcripts also implies the uniqueness of inputs to $\mathtt{H}_\mathtt{g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$. Since for each possible subgroup the corresponding $\mathtt{ssid}_i \subset \mathtt{sid}_i$ used by an instance of U_i for the computation of the subgroup key does not repeat and since $\mathtt{H}_\mathtt{g}$ and $\mathtt{H}_\mathtt{s}$ are modeled as random functions we can also apply the birthday paradox and upper-bound the probability of collisions for k_i and collisions between k_i and any $k_{i,J}$ by $(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{s}})^2/2^\kappa$. Thus,

$$|\Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_1] - \Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_0]| \leq \frac{N(\mathsf{q}_\mathtt{Ex} + \mathsf{q}_\mathtt{Se})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{s}})^2}{2^\kappa}.$$

Game $\mathbf{G}_2$. In this game we assume that Δ fails and bit b' is set at random if $\mathcal{A}$ forges some signature, i.e. if $\mathcal{A}$ queries *Send* containing (U_i, z_i, σ_i) with σ_i being a valid signature that has not been previously output by an uncorrupted instance of U_i. Hence,

$$|\Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_2] - \Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_1]| \leq N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_\Sigma(\kappa).$$

Game $\mathbf{G}_3$. In this game we let Δ guess a value $q^* \in [1, \mathsf{q}_\mathtt{Se}]$ and abort if the *TestGK* query is not asked for the session invoked by the q^*-th query. Similar to Game $\mathbf{G}_3$ from the proof of Theorem 7 we obtain

$$\Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_2] = \mathsf{q}_\mathtt{Se}\left(\Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_3] - \frac{1}{2}\right) + \frac{1}{2}.$$

Game $\mathbf{G}_4$. In this game we assume that Δ is given access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and computes in the group stage of the q^*-th session for each pair of consecutive users (U_i, U_{i+1}) in the cycle $z'_{i,i+1} = \mathtt{H}'(i, i+1)$, that is independent of $k'_{i,i+1}$, $i \in [1, n]$. Clearly both games remain indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1})$ for any $i \in [1, n]$. However, this query can be used to break the GDH problem in $\mathbb{G}$, i.e. Δ embeds g^a and g^b from the challenge of the GDH problem into the corresponding values g^{x_i} and $g^{x_{i+1}}$ used on behalf U_i and U_{i+1} in the group stage, and uses the access to the DDH oracle $\mathcal{D}$ in order to identify $k'_{i,i+1} = g^{x_i x_{i+1}}$ provided by $\mathcal{A}$ as input to $\mathtt{H}$. Therefore,

$$|\Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_4] - \Pr[\mathsf{Win}^{\mathsf{ake\text{-}g}}_3]| \leq N\mathsf{q}_\mathtt{H}\mathsf{Succ}^{\mathtt{GDH}}_\mathbb{G}(\kappa).$$

As a result of this game the group key k_i computed in the q^*-th group stage session is the output of $\mathtt{H}_\mathtt{g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ where all input values $z'_{1,2}, \ldots, z'_{n,1}$ are

uniform in $\{0,1\}^\kappa$. Since $\mathtt{H_g}$ is modeled as a random oracle and collisions on group keys computed in two different sessions have been excluded in Game $\mathbf{G}_1$ the probability that $\mathcal{A}$ wins without querying $\mathtt{H_g}(z'_{1,2},\ldots,z'_{n,1},\mathtt{sid}_i)$ is $1/2$; on the other hand, the probability that $\mathcal{A}$ asks such a query is given by $\mathtt{q}_{\mathtt{H_g}}/2^{n\kappa} < \mathtt{q}_{\mathtt{H_g}}/2^\kappa$ (for the guess of $z'_{1,2},\ldots,z'_{n,1}$). Hence, $\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}g}}] \leq 1/2 + \mathtt{q}_{\mathtt{H_g}}/2^\kappa$.

Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake\text{-}g}}_{\mathrm{mBD+S}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}g}}] - 1| \\ &\leq \frac{2N(\mathtt{q_{Ex}}+\mathtt{q_{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H_g}}+\mathtt{q}_{\mathtt{H_s}})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) \\ &\quad + 2\mathtt{q_{Se}}\left(N\mathtt{q_H}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathtt{q}_{\mathtt{H_g}}}{2^\kappa}\right).\end{aligned}$$

□

D Proof of Theorem 4

In the following we show that the advantage of $\mathcal{A}$ in distinguishing any $k_{i,J}$ computed by an instance Π_i^s of some uncorrupted U_i in the subgroup stage with regard to the subgroup $\mathtt{spid}_i^s = \{U_j\}_{j\in J}$ from some random element in $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0,\ldots,\mathbf{G}_8$ and denote by $\mathsf{Win}_i^{\mathsf{ake\text{-}s}}$ the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in the i-th game. Recall that in each game $\mathit{TestSK}(\Pi_i^s,\mathtt{spid}_i^s)$ is answered only if the instance-subgroup pair $(\Pi_i^s,\mathtt{spid}_i^s)$ is fresh.

Game $\mathbf{G}_0$. This is the real execution of mBD+S in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances as defined in $\mathsf{Game}^{\mathsf{ake\text{-}s},b}_{\mathcal{A},\mathrm{mBD+S}}(\kappa)$.

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}$, $\mathtt{H_g}$, and $\mathtt{H_s}$, which are modeled as random oracles.

Game $\mathbf{G}_1$. In this game we exclude for every honest user U_i the collisions of the transcripts (U_i,y_i) and group keys k_i. We also exclude collisions between k_i and any subgroup key $k_{i,J}$. Additionally, we exclude collisions amongst subgroup keys $k_{i,J}$ computed by U_i for different subgroups $\mathtt{spid}_i$ consisting of (possibly malicious) U_j, $j\in J$. Since sessions can be invoked via *Execute* and *Send* queries we obtain for the collision on transcripts (according to the birthday paradox) the probability of at most $N(\mathtt{q_{Ex}}+\mathtt{q_{Se}})^2/Q$ over all possible users. Collisions amongst k_i and collisions between k_i and any $k_{i,J}$ can be upper-bounded by $(\mathtt{q}_{\mathtt{H_g}}+\mathtt{q}_{\mathtt{H_s}})^2/2^\kappa$ using the same argument as in Game $\mathbf{G}_1$ from the proof of Theorem 9. This upper-bound also includes collisions amongst $k_{i,J}$ computed by an honest U_i for different subgroups involving U_j, $j\in J$. This holds since even if U_j, $j\in J$ is corrupted and chooses own y_j in some rogue way (e.g. in relation to other values) the input to $\mathtt{H_s}(z'_{1,2},\ldots,z'_{m,1},\mathtt{ssid}_i)$ still remains unique due to the uniqueness of the honest user's transcript (U_i,y_i) and identity U_j. (Note that for any subgroup $\mathtt{spid}_i \subset \mathtt{pid}_i$ the subgroup stage can be invoked only once.) Hence,

$$|\Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}s}}] - \Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}s}}]| \leq \frac{N(\mathtt{q_{Ex}}+\mathtt{q_{Se}})^2}{Q} + \frac{(\mathtt{q}_{\mathtt{H_g}}+\mathtt{q}_{\mathtt{H_s}})^2}{2^\kappa}.$$

Game $\mathbf{G}_2$. In this game we abort simulation (setting bit b' at random) if $\mathcal{A}$ queries *Send* containing (U_i,z_i,σ_i) with σ_i being a valid signature that has not been previously

output by an oracle of uncorrupted U_i. In other words the simulation fails if $\mathcal{A}$ outputs a successful forgery such that

$$|\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}s}}] - \Pr[\mathsf{Win}_1^{\mathsf{ake\text{-}s}}]| \leq N\mathsf{Succ}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\kappa).$$

Game $\mathbf{G}_3$. In this game Δ randomly chooses a session $q^* \in_R [1, \mathtt{q_{Se}}]$ and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ asks the *TestSK* query to some oracle Π_i^s that accepted in a group stage of another session, i.e. not in the q^*-th session. Using similar argument as in Game $\mathbf{G}_3$ from the proof of Theorem 7 we obtain

$$\Pr[\mathsf{Win}_2^{\mathsf{ake\text{-}s}}] = \mathtt{q_{Se}}\left(\Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}s}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

As a result of this game it is sufficient for the simulator to focus on the q^*-th session and to simulate instances of users participating in the group and subgroup stages of that session.

Game $\mathbf{G}_4$. In this game Δ is given access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and in the simulation of the q^*-th group stage session Δ computes for each pair of consecutive users (U_i, U_{i+1}) in the cycle $z'_{i,i+1} = \mathtt{H}'(i, i+1)$ if both users U_i and U_{i+1} are uncorrupted. Note that the definition of AKE-security for subgroup keys requires the *TestSK* query to be asked to a fresh instance-subgroup pair $(\Pi_i^s, \mathtt{spid}_i^s)$, that is users in the subgroup $\mathtt{spid}_i^s$ remain uncorrupted until the computation of the subgroup key. Our replacement remains indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1})$, in which case we can solve the GDH problem. Thus,

$$|\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}s}}] - \Pr[\mathsf{Win}_3^{\mathsf{ake\text{-}s}}]| \leq N\mathtt{q_H}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa).$$

Note also that the simulation remains perfect if $\mathcal{A}$ corrupts U_i or U_{i+1} later since the corrupt query does not reveal their secret exponents.

This game also implies that in the q^*-th session values $z'_{i,i+1}$ computed for pairs (U_i, U_{i+1}) of uncorrupted users and used in the computation of $k_i = \mathtt{H_g}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ are uniform in $\{0,1\}^\kappa$.

Game $\mathbf{G}_5$. In this game Δ tries to guess the invocation of the subgroup stage which will be selected by $\mathcal{A}$ as a target for the attack. That is Δ chooses a value $q^{**} \in_R [1, \mathtt{q_{SKE}}]$ at random and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ does not ask its *TestSK* query to an instance that participates in the q^{**}-th subgroup stage. The probability that Δ correctly guesses q^{**} is $1/\mathtt{q_{SKE}}$, and

$$\Pr[\mathsf{Win}_4^{\mathsf{ake\text{-}s}}] = \mathtt{q_{SKE}}\left(\Pr[\mathsf{Win}_5^{\mathsf{ake\text{-}s}}] - \frac{1}{2}\right) + \frac{1}{2}.$$

If the guess is successful then users in the target subgroup remain uncorrupted until the subgroup key is computed. The target subgroup defines its own cycle for the order of users in which two neighboring users U_i and U_{i+1} may have not been neighbors in the cycle defined by the group stage. Therefore, we need to apply the same changes as in the previous game with respect to the subgroup's cycle in order to achieve uniformity of values $z'_{i,i+1}$ that will be used to compute the subgroup key.

Game $\mathbf{G}_6$. In this game Δ has access to the private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$. In the simulation of the guessed q^{**}-th subgroup stage Δ computes for each pair of consecutive users (U_i, U_{i+1}) in the cycle defined by $\mathtt{spid}_i^s = (U_1, \ldots, U_m)$ the intermediate value $z'_{i,i+1} = \mathtt{H}'(i, i+1, \mathtt{ssid}_i)$, that is independent of $k'_{i,i+1}$. This

replacement remains indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}(k'_{i,i+1}, \mathtt{ssid}_i)$ in which case we again can solve the GDH problem. Since m (the size of the subgroup) is strictly smaller than the size of the group $n \leq N$ we have

$$|\Pr[\mathsf{Win}_6^{\mathsf{ake\text{-}s}}] - \Pr[\mathsf{Win}_5^{\mathsf{ake\text{-}s}}]| \leq (N-1)\mathsf{q}_{\mathtt{H}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{GDH}}(\kappa).$$

This game implies that in the q^{**}-th subgroup stage the computation of the subgroup key $k_{i,J} = \mathtt{H}_{\mathtt{g}}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{sid}_i)$ uses values $z'_{i,i+1}$ that are uniform in $\{0,1\}^\kappa$. Hence, the probability that $\mathcal{A}$ wins without querying $\mathtt{H}_{\mathtt{s}}(z'_{1,2}, \ldots, z'_{n,1}, \mathtt{ssid}_i)$ is $1/2$; on the other hand, the probability that $\mathcal{A}$ asks such a query is given by $\mathsf{q}_{\mathtt{H}_{\mathtt{s}}}/2^{m\kappa} < \mathsf{q}_{\mathtt{H}_{\mathtt{s}}}/2^\kappa$ (for the guess of $z'_{1,2}, \ldots, z'_{n,1}$). Hence, $\Pr[\mathsf{Win}_6^{\mathsf{ake\text{-}s}}] \leq 1/2 + \mathsf{q}_{\mathtt{H}_{\mathtt{s}}}/2^\kappa$.

Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{ake\text{-}s}}_{\mathrm{mBD}+\mathrm{S}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ake\text{-}s}}] - 1| \\
&\leq \frac{2N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_{\mathtt{g}}} + \mathsf{q}_{\mathtt{H}_{\mathtt{s}}})^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf\text{-}cma}}_{\Sigma}(\kappa) \\
&\quad + 2\mathsf{q}_{\mathtt{Se}}\left(\big(N + (N-1)\mathsf{q}_{\mathtt{SKE}}\big)\mathsf{q}_{\mathtt{H}}\mathsf{Succ}^{\mathtt{GDH}}_{\mathbb{G}}(\kappa) + \frac{\mathsf{q}_{\mathtt{SKE}}\mathsf{q}_{\mathtt{H}_{\mathtt{s}}}}{2^\kappa}\right).
\end{aligned}$$

□