This is the full version of the paper which appeared in:
Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009.
M. Abdalla et al. (Eds.) Springer-Verlag, LNCS vol. 5536 , pp. 1-19.

Group Key Exchange Enabling On-Demand Derivation of Peer-to-Peer Keys

(Full Version)

Mark Manulis

Cryptographic Protocols Group
Department of Computer Science
TU Darmstadt & CASED, Germany
mark@manulis.eu

Abstract. We enrich the classical notion of group key exchange (GKE) protocols by a new property that allows each pair of users to derive an independent *peer-to-peer (p2p) key* on-demand and without any subsequent communication; this, in addition to the classical *group key* shared amongst all the users. We show that GKE protocols enriched in this way impose new security challenges concerning the secrecy and independence of both key types. The special attention should be paid to possible collusion attacks aiming to break the secrecy of p2p keys possibly established between any two non-colluding users.
In our constructions we utilize the well-known parallel Diffie-Hellman key exchange (PDHKE) technique in which each party uses the same exponent for the computation of p2p keys with its peers. First, we consider PDHKE in GKE protocols where parties securely transport their secrets for the establishment of the group key. For this we use an efficient multi-recipient ElGamal encryption scheme. Further, based on PDHKE we design a generic compiler for GKE protocols that extend the classical Diffie-Hellman method. Finally, we investigate possible optimizations of these protocols allowing parties to *re-use* their exponents to compute both group and p2p keys, and show that not all such GKE protocols can be optimized.

Key words: group key exchange, peer-to-peer keys, on-demand derivation

1 Introduction

Traditional *group key exchange (GKE)* protocols allow users to agree on a secret *group key* and are fundamental for securing applications that require group communication. However, messages authenticated or encrypted with the group key attest only that the originator of the message is a valid member of the group. The goal of this paper is to investigate the enrichment of GKE protocols with the additional derivation of *peer-to-peer (p2p) keys* for any pair of users. A single run of a GKE protocol enriched in this way would suffice to set up a secure group channel providing possibly each pair of users with an independent secure peer-to-peer channel "for free", thus implicitly allowing for a secure combination of group and p2p communication. Note that messages authenticated or encrypted with a p2p key would attest not only the group membership but also allow for the identification of the sender. For example, in digital conferences or instant messaging systems each user can participate in a secure group discussion and if necessary switch for a while to a secure bilateral discussion with some other user; or a user can encrypt some file for all users using the group key and attach supplementary files encrypted with p2p keys for the selected subset of its peers.

Obviously, the simultaneous computation of group and p2p keys can be achieved through the execution of a GKE protocol in parallel with the execution of a two-party key exchange (2KE) protocol between every pair of users. The drawback of this approach is that it would require $(n^2 - n)/2$ parallel 2KE executions in order to provide each pair with the own key (where n is the number of users). The only way to avoid such parallel 2KE executions is to consider solutions where p2p keys are computed *on-demand*; we denote such GKE protocols by GKE+P.

A rather naïve construction of GKE+P protocols can be obtained from the execution of a GKE protocol followed by a separate execution of a 2KE protocol between some pair of users. The drawback of this solution is the additional interaction for the computation of p2p keys (in the worst case requiring up to $n - 1$ different

2KE protocol runs involving the same user) and the deployment of two different protocols (GKE and 2KE). Therefore, since GKE participants already interact to establish the group key it appears interesting to investigate whether GKE+P protocols can be constructed enabling the completely *non-interactive* derivation of p2p keys?

GKE+P protocols raise new security challenges concerning the independence of group and p2p keys. Traditional GKE protocols require that a session group key remains secret from any adversary that is an external entity to that session. In GKE+P protocols this requirement should hold even in case where p2p keys leak. By the same token GKE+P protocols should provide secrecy of the p2p keys computed in some session independent of whether the adversary learns the group key or not. However, the most significant challenge specific to GKE+P protocols results from the independence amongst different p2p keys computed in the same session and even by the same user (for different peers). In particular GKE+P protocols should provide secrecy of some session p2p key if other participants that are not intended to compute that key collude. Thus, when defining the secrecy of some session p2p key we should no longer assume that the adversary remains an external entity to that session but rather that it may act on behalf of colluding participants and thus deviate from the protocol specification.

Specification of the appropriate security requirements and efficient, provably secure solutions for GKE+P protocols represents the main focus of our work.

1.1 Related Work

The basic security goal of any key exchange protocol is called (Authenticated) Key Exchange security ((A)KE-security, for short) and deals with the secrecy or indistinguishability of the established session group key with respect to an (active) adversary which is usually modeled as an external entity from the perspective of the attacked session. This requirement became an inherent part of all security models for 2KE protocols, e.g. [3, 5–7, 17–19, 34, 38], and GKE protocols, e.g. [10, 11, 13, 15, 28, 29]. A general signature-based compilation technique proposed by Katz and Yung [29] can turn any KE-secure (group) key exchange protocol into an AKE-secure one, thus by adding the authentication and thwarting possible impersonation attacks. Additionally, we remark that some of the mentioned security models for GKE protocols (e.g. [12, 13, 28]) aim at defining optional security against insider attacks, and the corresponding compilers defined in these papers can turn any AKE-secure GKE protocol into a protocol that withstands such attacks. These compilers also provide the so-called requirement of mutual authentication (MA) [7, 11, 15], which ensures the bilateral authentication of all protocol participants and is usually combined with a key confirmation step.

From the variety of the existing GKE protocols (see [9, 35] for surveys) of special interest in the context of our GKE+P constructions are the (unauthenticated) extensions of the classical 2KE approach by Diffie and Hellman [21] to a group setting, e.g. [16, 20, 24, 31, 32, 37, 39, 40]. Let us denote all these protocols for simplicity as *Group Diffie-Hellman (GroupDH)* protocols since they derive the group key from some shared secret which in turn depends on the individual exponents chosen by the protocol participants during the execution. For the design of GKE+P protocols it appears promising to investigate to what extent the existing GroupDH protocols allow for the non-interactive, on-demand computation of p2p keys, in particular whether or not secret exponents used in these GroupDH protocols can be safely re-used for the computation of p2p keys.

GKE protocols proposed in [1, 36] are partially related since they consider a 2KE protocol as a building block in order to obtain a secure GKE protocol, yet without enabling on-demand computation of p2p keys amongst any pair of users. Also, the so-called *group secret handshakes* [25, 26] should be noticed since these can be seen as extensions of GKE protocols with another property called affiliation-hiding. We mention them here since the on-demand computation of p2p keys can be also considered in that scenarios (in particular in particular our results can be extended to deal with [25] that is based on the GKE protocol from [16]).

One of the main building blocks across all our GKE+P constructions is the *parallel* execution of the 2KE Diffie-Hellman protocol (PDHKE), in which each user broadcasts a value of the form g^x (for the appropriate generator g and private user's exponent x) and uses x for the computation of different p2p keys. In this context, Jeong and Lee [27] recently specified and analyzed a related mechanism where keys are derived in parallel from ephemeral and long-lived exponents. However, their security model does not consider collusion attacks against the secrecy of p2p keys computed by non-colluding users. Note also the recent work by Biswas [8] who revised the 2KE Diffie-Hellman protocol allowing its participants to choose two different exponents each and obtain 15 different shared keys.

1.2 Contributions and Organization

We start in Section 2 with the extension of the classical GKE security model from [29] in order to address the additional challenges of GKE+P protocols and define the corresponding requirements of (A)KE-security of group and p2p keys; the latter in the presence of collusion attacks. Our model is designed in a modular way and can be selectively applied to GKE+P and GKE protocols, and also to the protocols like PDHKE. In Section 3 we introduce general notations and recall some classical assumptions.

In Section 4 we present and analyze our first GKE+P protocol, denoted PDHKE-MRE. In this protocol we merge PDHKE with the multi-recipient ElGamal encryption (MRE) from [4, 33]. PDHKE-MRE optimizes the combination of PDHKE and MRE in that it utilizes user's exponent for both — generation of p2p keys and decryption of ElGamal ciphertexts. This optimization is tricky (compared to the simple "black-box" combination) since it requires an additional hardness assumption. Our security analysis of PDHKE-MRE also demonstrates that PDHKE can be used as a stand-alone protocol to obtain KE-secure p2p keys in the presence of collusion attacks.

In Section 5 we obtain more efficient GKE+P protocols from GroupDH protocols (see related work for examples). First, we describe a general compilation technique to obtain GKE+P solutions from any GroupDH protocol based on PDHKE, yet assuming that the exponents used for the derivation of p2p keys are independent from those used in the computation of the group key. Additionally, we investigate whether private exponents that are implicit to the GroupDH protocols can be re-used for the on-demand computation of p2p keys. The key observation here is that many GroupDH protocols require each user U_i to choose some exponent x_i and broadcast a public value g^{x_i}. The natural question is whether a value $g^{x_i x_j}$, if computed from the exponents x_i and x_j used in the GroupDH protocol, would be suitable for the derivation of a secure p2p key between U_i and U_j? In this light we analyze the well-known communication-efficient protocols by Burmester and Desmedt (BD) [16] and by Kim, Perrig, and Tsudik (KPT) [31] (the latter as a representative for the family of Tree Diffie-Hellman protocols). We show that in the BD protocol this technique will not guarantee the KE-security of p2p keys, whereas in the KPT protocol it will, though at the cost of an additional hardness assumption. The latter result is of special interest since we do not introduce any new communication costs to the KPT protocol.

In Section 6, we compare the performance of the introduced GKE+P protocols.

In Section 7 we show that the authentication compiler introduced in [29] for securing traditional KE-secure GKE protocols is also sufficient for adding the authentication to KE-secure GKE+P protocols.

2 Security Model for GKE+P Protocols

Our security model for GKE+P protocols extends the meanwhile standard GKE security model from [29] by capturing the additional requirements concerning the on-demand computation of p2p keys.

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

By $\mathcal{U}$ we denote a set of at most N users (more precisely, their identities which are assumed to be unique) in the universe. Any subset of n users ($2 \leq n \leq N$) can participate in a single session of a GKE+P protocol $\mathcal{P}$. Each $U_i \in \mathcal{U}$ holds a (secret) long-lived key LL_i.[1] The participation of U_i in distinct, possibly concurrent protocol sessions is modeled via an unlimited number of *instances* Π_i^s, $s \in \mathbb{N}$. Each instance Π_i^s can be invoked for one session with some partner id $\mathtt{pid}_i^s \subseteq \mathcal{U}$ encompassing the identities of the intended participants (including U_i). At the end of the interactive phase Π_i^s holds a *session id* $\mathtt{sid}_i^s$ which uniquely identifies the session. Two instances Π_i^s and Π_j^t are considered as *partnered* if $\mathtt{sid}_i^s = \mathtt{sid}_j^t$ and $\mathtt{pid}_i^s = \mathtt{pid}_j^t$. The success of the interactive phase by some instance Π_i^s is modeled through its *acceptance*, in which case the instance holds a *session group key* k_i^s. Each instance Π_i^s that has accepted can later decide to compute a *session p2p key* $k_{i,j}^s$ for some user $U_j \in \mathtt{pid}_i^s$. We are now ready to formally define what a GKE+P protocol is.

[1] Our GKE+P protocols are first analyzed in the authenticated links model where long-lived keys are assumed to be empty. The authentication in GKE+P protocols using the compiler technique from [29] that we discuss in Section 7 will assume that each LL_i corresponds to some digital signature key pair.

Definition 1 (GKE+P Protocol and Correctness). *$\mathcal{P}$ is a* group key exchange protocol enabling on-demand derivation of p2p keys (GKE+P) *if $\mathcal{P}$ consists of the group key exchange protocol* GKE *and a p2p key derivation algorithm* P2P *defined as follows:*

$\mathcal{P}$.GKE$(U_1, \dots, U_n)$*: For each input U_i a new instance Π_i^s 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.*

$\mathcal{P}$.P2P(Π_i^s, U_j)*: On input an accepted instance Π_i^s and some user identity $U_j \in \mathtt{pid}_i^s$ this deterministic algorithm outputs the session p2p key $k_{i,j}^s$. (We assume that* P2P *is given only for groups of size $n \geq 3$ since for $n = 2$ the group key is sufficient.)*

A GKE+P protocol $\mathcal{P}$ is correct *if (when no adversary is present) all instances participating in the protocol* $\mathcal{P}$.GKE *accept with identical group keys and* $\mathcal{P}$.P2P$(\Pi_i^s, U_j) = \mathcal{P}$.P2P$(\Pi_j^t, U_i)$ *holds for any pair of partnered instances Π_i^s and Π_j^t.*

2.2 Adversarial Model and Security Goals

Security model for GKE+P protocols must address the following two challenges that are new compared to the classical GKE setting: The first challenge is to model the secrecy of a session group key k_i^s by taking into account possible leakage of any p2p key that can be computed in that session (including all $k_{i,j}^s$). Since for the secrecy of the session group key the adversary is treated as an external entity and not as a legitimate participant of that session our model should provide the adversary with the ability to schedule the on-demand computation of p2p keys and to reveal them. The second, main challenge is to model the secrecy of a session p2p key $k_{i,j}^s$ by taking into account the leakage of the group key and also the leakage of other p2p keys computed in that session (with the obvious exclusion of $k_{j,i}^t$ when Π_i^s and Π_j^t are partnered). Note that the secrecy of p2p keys does not require the adversary to be an external entity (unlike the secrecy of the group key). Hence, we have to face possible collusion attacks aiming to break the secrecy of $k_{i,j}^s$ and allow for the active participation of the adversary in the attacked session.

ADVERSARIAL MODEL The adversary $\mathcal{A}$, modeled as a PPT machine, can schedule the protocol execution and mount own attacks via the following queries:

- *Execute*$(U_1, \dots, U_n)$: This query executes the protocol between new instances of $U_1, \dots, U_n \in \mathcal{U}$ 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*$(U_i, ('start', U_1, \dots, U_n))$ creates a new instance Π_i^s with $\mathtt{pid}_i^s := \{U_1, \dots, U_n\}$ and provides $\mathcal{A}$ with the first protocol message.
- *Peer*(Π_i^s, U_j): This query allows $\mathcal{A}$ to schedule the on-demand computation of p2p keys. In response, Π_i^s computes $k_{i,j}^s$; the query is processed only if Π_i^s has accepted and $U_j \in \mathtt{pid}_i^s$, and it can be asked only once per input (Π_i^s, U_j).
- *Reveal*(Π_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.
- *RevealPeer*(Π_i^s, U_j): This query models the leakage of p2p keys and provides $\mathcal{A}$ with $k_{i,j}^s$; the query is answered only if *Peer*(Π_i^s, U_j) has already been asked and processed.
- *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.
- *Test*(Π_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.
- *TestPeer*(Π_i^s, U_j): This query models indistinguishability of session p2p 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 $k_{i,j}^s$. It is answered only if *Peer*(Π_i^s, U_j) has been previously asked and processed.

TERMINOLOGY We say that U is *honest* if no $\mathit{Corrupt}(U)$ 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 (A)KE-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 (A)KE-security of group keys. Our definition is essentially the one given in [29].

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

Note that in the context of GKE+P 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) p2p keys.

Additionally, we define the new notion of *instance-user freshness* which will be used to specify the (A)KE-security of p2p keys.

Definition 3 (Instance-User Freshness). *An instance-user pair (Π_i^s, U_j) is* fresh *if Π_i^s has accepted and none of the following is true, whereby Π_j^t denotes an instance partnered with Π_i^s: (1) $\mathit{RevealPeer}(\Pi_i^s, U_j)$ or $\mathit{RevealPeer}(\Pi_j^t, U_i)$ 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 U_j. Also $\mathcal{A}$ may learn the group key k_i and all p2p keys except for $k_{i,j}$. This models possible collusion of participants during the execution of the protocol aiming to break the secrecy of the p2p key $k_{i,j}^s$.

(A)KE-SECURITY OF GROUP AND P2P KEYS For the (A)KE-security of group keys we follow the definition from [29]. Note that in case of KE-security $\mathcal{A}$ is restricted to pure eavesdropping attacks via the *Execute* query without being able to access the *Send* queries.

Definition 4 ((A)KE-Security of Group Keys). *Let $\mathcal{P}$ be a correct GKE+P protocol and b a uniformly chosen bit. By $\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{(a)ke\text{-}g},b}(\kappa)$ we define the following adversarial game, which involves a PPT adversary $\mathcal{A}$ that is given access to all queries (except for* Send *when dealing with KE-security):*

- *$\mathcal{A}$ interacts via queries;*
- *at some point $\mathcal{A}$ asks a $\mathsf{Test}(\Pi_i^s)$ 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.*

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

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

Finally, we define (A)KE-security of p2p keys where we must consider possible collusion attacks. For this it is essential to allow $\mathcal{A}$ access to *Send* queries, even in the case of KE-security. The difficulty is that given general access to *Send* queries $\mathcal{A}$ can trivially impersonate any protocol participant. Hence, when dealing with KE-security of p2p keys we must further restrict $\mathcal{A}$ to truly forward all messages sent by honest users. According to our definition of instance-user freshness of (Π_i^s, U_j) this restriction will imply an unbiased communication between the instances of U_i and U_j.

Definition 5 ((A)KE-security of P2P Keys). *Let $\mathcal{P}$ be a correct GKE+P protocol and b a uniformly chosen bit. By $\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\mathsf{(a)ke\text{-}p},b}(\kappa)$ we define the following adversarial game, which involves a PPT adversary $\mathcal{A}$ that is given access to all queries (with the restriction to truly forward all messages of honest users in case of KE-security):*

- *$\mathcal{A}$ interacts via queries;*
- *at some point $\mathcal{A}$ asks a* TestPeer(Π_i^s, U_j) *query for some instance-user pair (Π_i^s, U_j) 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.*

$$\text{We define:} \qquad \mathsf{Adv}_{\mathcal{A},\mathcal{P}}^{\text{(a)ke-p}}(\kappa) := \left|2\Pr[\mathsf{Game}_{\mathcal{A},\mathcal{P}}^{\text{(a)ke-p},b}(\kappa) = b] - 1\right|$$

and denote with $\mathsf{Adv}_{\mathcal{P}}^{\text{(a)ke-p}}(\kappa)$ the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that $\mathcal{P}$ provides (A)KE-security of p2p keys *if this advantage is negligible.*

3 General Notations and Preliminaries

Throughout the paper, unless otherwise specified, by $\mathbb{G} := \langle g \rangle$ we denote a cyclic subgroup in $\mathbb{Z}_P^*$ of prime order $Q|P-1$ generated by g, where P is also prime. By $\mathtt{H_g}, \mathtt{H_p} : \{0,1\}^* \to \{0,1\}^\kappa$ we denote two cryptographic hash functions, which will be used in our constructions for the purpose of derivation of group and p2p keys, respectively. Additionally, we recall the following three well-known cryptographic assumptions:

Definition 6 (Hardness Assumptions). *Let $\mathbb{G} := \langle g \rangle$ as above and $a, b, c \in_R \mathbb{Z}_Q$. We say that:*
The Discrete Logarithm (DL) problem is hard in $\mathbb{G}$ *if the following success probability is negligible:*

$$\mathsf{Succ}_{\mathbb{G}}^{\mathtt{DL}}(\kappa) := \max_{\mathcal{A}'} \left(\Pr_a \left[\mathcal{A}'(g, g^a) = a\right]\right);$$

The Decisional Diffie-Hellman (DDH) problem is hard in $\mathbb{G} = \langle g \rangle$ *if the following advantage is negligible:*

$$\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa) := \max_{\mathcal{A}'} \left| \Pr_{a,b}\left[\mathcal{A}'(g, g^a, g^b, g^{ab}) = 1\right] - \Pr_{a,b,c}\left[\mathcal{A}'(g, g^a, g^b, g^c) = 1\right]\right|;$$

The Square-Exponent Decisional Diffie-Hellman (SEDDH) problem is hard in $\mathbb{G}$[2] *if the following advantage is negligible:*

$$\mathsf{Adv}_{\mathbb{G}}^{\mathtt{SEDDH}}(\kappa) := \max_{\mathcal{A}'} \left| \Pr_{a}\left[\mathcal{A}'(g, g^a, g^{a^2}) = 1\right] - \Pr_{a,b}\left[\mathcal{A}'(g, g^a, g^b) = 1\right]\right|.$$

Note that $\mathsf{Succ}_{\mathbb{G}}^{\mathtt{DL}}(\kappa)$, $\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa)$, *and* $\mathsf{Adv}_{\mathbb{G}}^{\mathtt{SEDDH}}(\kappa)$ *are computed over all PPT adversaries $\mathcal{A}'$ running within time κ.*

4 Optimized PDHKE-MRE

Here we introduce our first GKE+P protocol, called PDHKE-MRE. The optimization concerns the utilization of each $x_i \in \mathbb{Z}_Q$ as a private decryption key for the multi-recipient ElGamal encryption [4, 33] *and* as a secret exponent for the computation of p2p keys via PDHKE. Note that PDHKE-MRE can be generalized by applying other multi-recipient public key encryption schemes [4]. However, in this case our optimization may no longer hold.

4.1 Parallel Diffie-Hellman Key Exchange (PDHKE)

Assuming that users interact over the authenticated channels we define PDHKE as follows (we describe all our protocols from the perspective of one session using the identities of users and not their instances):

Round 1 Each U_i chooses a random $x_i \in_R \mathbb{Z}_Q$ and broadcasts $y_i := g^{x_i}$.

[2] Wolf [41] showed that SEDDH is reducible to DDH and that the converse does not hold.

P2P key computation Each U_i for a given identity U_j computes $k'_{i,j} := g^{x_i x_j}$ and derives $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i|y_i, U_j|y_j)$. W.l.o.g. we assume that $i < j$ and that if U_j computes own p2p key for U_i it uses the same order for the inputs of $\mathtt{H_p}$ as U_i does.

A special attention in PDHKE should be paid to the key derivation step based on $\mathtt{H_p}$. Note that in the random oracle model this construction ensures the independence of different p2p keys (possibly computed by the same U_i for different U_j). The reason is that if U_i is honest then the hash input remains unique for each derived p2p key (due to the uniqueness of $U_i|y_i$ across different sessions and the uniqueness of each U_j within the same session). The uniqueness of hash inputs is of importance. Assume, that $k_{i,j}$ would be derived as $\mathtt{H_p}(k'_{i,j})$. In this case $\mathcal{A}$ may impose dependency between $k'_{i,j}$ and $k'_{i,a}$ for some user U_a that it may control, e.g. by using $y_a = y_j$. With this simple attack $\mathcal{A}$ cannot compute $k'_{i,a}$ due to the lack of $x_a = x_j$ but it can easily distinguish $k_{i,j}$ by obtaining $k_{i,a}$ (which would then be equal to $k_{i,j}$) via an appropriate *RevealPeer* query to an instance of honest U_i.

4.2 Multi-Recipient ElGamal Encryption (MRE)

In the classical ElGamal encryption [23] a message $m \in \mathbb{G}$ is encrypted under the recipient's public key $y = g^x$ through the computation of the ciphertext $(g^r, y^r m)$ using some random $r \in_R \mathbb{Z}_Q$. A multi-recipient ElGamal encryption (MRE) [4,33] re-uses the random exponent r for the construction of ciphertexts of several messages $m_1, \ldots, m_n$ under several public keys $y_1 = g^{x_1}, \ldots, y_n = g^{x_n}$, i.e., by computation of $(g^r, y_1^r m_1, \ldots, y_n^r m_n)$. However, in PDHKE-MRE we will be encrypting the same message $m = m_1 = \ldots = m_n$. For this case [33] defines a computation-efficient MRE version where the ciphertext has the form $(mg^r, y_1^r, \ldots, y_n^r)$. Obviously, this technique results in shorter ciphertexts should a single protocol message contain ciphertexts for multiple recipients. Informally, the IND-CPA security of MRE means that any encrypted plaintext remains indistinguishable, even if the adversary is in possession of the secret keys $\{x_j\}_{j \neq i}$. This has been proven in [33] (and also in [4] under a stronger setting) based on the DDH assumption.

4.3 Description of PDHKE-MRE

Our optimization in PDHKE-MRE is based on the idea to re-use the same exponent x_i for both — derivation of p2p keys from $k'_{i,j} = g^{x_i x_j}$ and decryption of $\{\bar{x}_j\}_j$. The protocol PDHKE-MRE.GKE amongst a set of n users $U_1, \ldots, U_n$ proceeds in two rounds:

Round 1 Each U_i chooses a random $x_i \in_R \mathbb{Z}_Q$ and broadcasts $y_i := g^{x_i}$.

Round 2 Each U_i chooses random $\bar{x}_i \in_R \mathbb{G}$, $r_i \in_R \mathbb{Z}_Q$, computes $z_i := \bar{x}_i g^{r_i}$ and $\{z_{i,j} := y_j^{r_i}\}_j$ and broadcasts $(z_i, \{z_{i,j}\}_j)$.

Group key computation Each U_i decrypts $\left\{\bar{x}_j := \frac{z_j}{z_{j,i}^{(1/x_i)}}\right\}_j$ and accepts with $k_i := \mathtt{H_g}(\bar{x}_1, \ldots, \bar{x}_n, \mathtt{sid}_i)$ where $\mathtt{sid}_i := (U_1|y_1, \ldots, U_n|y_n)$.

The algorithm PDHKE-MRE.P2P when executed by some user U_i for a peer U_j computes $k'_{i,j} := g^{x_i x_j}$ and outputs $k_{i,j} := \mathtt{H_p}(k'_{i,j}, U_i|y_i, U_j|y_j)$ whereby the inputs $U_i|y_i$ and $U_j|y_j$ are taken from $\mathtt{sid}_i$. W.l.o.g. we assume that $i < j$ and that U_j will use the same order for the inputs to $\mathtt{H_p}$ in the computation of $k_{j,i}$.

4.4 Security Analysis of PDHKE-MRE

Although the stand-alone security of MRE can be proven under the DDH assumption, its optimized merge with PDHKE requires the additional use of the SEDDH assumption for the proof of KE-security of group keys as motivated in the following.

The natural way to prove the IND-CPA security of MRE under the DDH assumption would be to simulate $y_j = g^{a\alpha_j}$, $z_i = \bar{x}_i g^{b\beta_i}$, and each $z_{i,j} = g^{ab\alpha_j\beta_i}$, where g^a and g^b belong to the DDH tuple and $\alpha_j, \beta_i \in_R \mathbb{Z}_Q$ (observe that the DDH problem is self-reducible). However, in PDHKE-MRE this simulation would also mean that $y_i = g^{a\alpha_i}$ for some $\alpha_i \in_R \mathbb{Z}_Q$ and possibly imply $g^{x_i x_j} = g^{a^2\alpha_i\alpha_j}$ upon the simulation of p2p keys, which in turn involves g^{a^2} from the SEDDH tuple.

Theorem 1. *If both problems DDH and SEDDH are hard in* $\mathbb{G}$ *then* PDHKE-MRE *provides KE-security of group keys and*

$$\mathsf{Adv}^{\mathsf{ke\text{-}g}}_{\mathrm{PDHKE\text{-}MRE}}(\kappa) \leq \frac{2(N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2+\mathtt{q}_{\mathtt{H_g}})}{Q}+\frac{(\mathtt{q}_{\mathtt{H_g}}+\mathtt{q}_{\mathtt{H_p}})^2}{2^{\kappa-1}}+2N\mathsf{Adv}^{\mathtt{SEDDH}}_{\mathbb{G}}(\kappa)+2N(N-1)\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa)$$

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

Proof. 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 $\mathbf{G}_0,\ldots,\mathbf{G}_5$ and denote by $\mathsf{Win}^{\mathsf{ke\text{-}g}}_i$ 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 $\mathit{Test}(\Pi^s_i)$ is answered only if the instance Π^s_i is fresh.

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

We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H_g}$ 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 $k_{i,j}$. 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 $N(\mathtt{q}_{\mathtt{Ex}}+\mathtt{q}_{\mathtt{Se}})^2/Q$ 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 $\mathtt{H_g}(\bar{x}_1,\ldots,\bar{x}_n,U_1|y_1,\ldots,U_n|y_n)$. 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 by $(\mathtt{q}_{\mathtt{H_g}}+\mathtt{q}_{\mathtt{H_p}})^2/2^\kappa$. Thus,

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

Game $\mathbf{G}_2$**.** In this game we assume that Δ chooses some random value $a \in_R \mathbb{Z}_Q$ and simulates the protocol execution in response to some $\mathit{Execute}(U_1,\ldots,U_n)$ query according to the real specification for the only difference that it sets each $y_i := g^{a\alpha_i}$ for some random $\alpha_i \in_R \mathbb{Z}_Q$ and computes any $k_{i,j}$ using $k'_{i,j} := g^{a^2\alpha_i\alpha_j}$. Clearly, the distributions do not change, thus $\Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_1] = \Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_2]$.

Game $\mathbf{G}_3$**.** In this game we assume that Δ chooses an additional random value $b \in_R \mathbb{Z}_Q$ and simulates the protocol execution in response to a $\mathit{Execute}(U_1,\ldots,U_n)$ query as in the previous game except that now any $k_{i,j}$ is computed using $k'_{i,j} := g^{b\alpha_i\alpha_j}$, thus independently from both $y_i = g^{a\alpha_i}$ and $y_j = g^{a\alpha_j}$. Obviously, both games remain indistinguishable in case that the SEDDH assumption holds in $\mathbb{G}$ (note that (g, g^a, g^{a^2}) and (g, g^a, g^b) embedded respectively in $\mathbf{G}_2$ and $\mathbf{G}_3$ correspond to a SEDDH tuple). Hence,

$$|\Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_3]-\Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_2]| \leq N\mathsf{Adv}^{\mathtt{SEDDH}}_{\mathbb{G}}(\kappa).$$

Observe, that in this game all $k'_{i,j}$ are uniformly distributed in $\mathbb{G}$ and independent, due to the use of α_i and α_j (in each session there are at most n linearly independent equations of the form $\log_g k'_{i,j} = b\alpha_i\alpha_j$). Note also that in this game Δ computes z_i and $z_{i,j}$ as specified in the protocol.

Game $\mathbf{G}_4$**.** Now we assume that Δ (holding $a, b \in_R \mathbb{Z}_Q$) modifies the simulation as follows: It no longer computes $k'_{i,j}$ using b but chooses it at random from $\mathbb{Z}_Q$. Note, this does not affect the distribution of $k'_{i,j}$ compared to the previous game. Δ computes now each $z_i := \bar{x}_i g^{b\beta_i}$ using some additional random β_i and, therefore, also each $z_{i,j} := g^{ab\alpha_j\beta_i}$ (recall that we still have $y_j = g^{a\alpha_j}$). Note that all distributions remain the same compared to the previous game, s.t. $\Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_3] = \Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_4]$.

Game $\mathbf{G}_5$**.** In this game we assume that Δ chooses an additional random value $c \in_R \mathbb{Z}_Q$ and simulates the protocol execution in response to a $\mathit{Execute}(U_1,\ldots,U_n)$ query as in the previous game except that it computes each $z_{i,j} := g^{c\beta_i\alpha_j}$, that is completely independent of $z_i = g^{b\beta_i}$ and $y_j = g^{a\alpha_j}$. Both games remain indistinguishable in case that the DDH assumption holds in $\mathbb{G}$ (note that (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) embedded respectively in $\mathbf{G}_4$ and $\mathbf{G}_5$ correspond to a DDH tuple). Hence,

$$|\Pr[\mathsf{Win}_5^{\text{ke-g}}] - \Pr[\mathsf{Win}_4^{\text{ke-g}}]| \leq N(N-1)\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa).$$

Observe, that in this game for any fixed index i all values $z_{i,j}$ are uniformly distributed in $\mathbb{G}$ and independent due to the use of β_i and different α_j (in each session for every index i there are exactly $n-1$ linearly independent equations of the form $\log_g z_{i,j} = c\beta_i\alpha_j$). This implies that each $\bar{x}_i$ (encrypted in all $z_{i,j}$) is uniformly distributed in $\mathbb{G}$.

Let us now investigate the success probability of $\mathcal{A}$ in this game. From Game $\mathbf{G}_1$ we know that the input to $\mathtt{H}_\mathtt{g}$ used by some honest U_i to compute the group key k_i is unique across multiple sessions. Since $\mathtt{H}_\mathtt{g}$ is modeled as a random function the group key k_i computed by some fresh instance of U_i remains independent of the group keys computed in other sessions and any p2p keys $k_{i,j}$. Hence, the probability that $\mathcal{A}$ wins without querying $\mathtt{H}_\mathtt{g}(\bar{x}_1, \ldots, \bar{x}_n, U_1|y_1, \ldots, U_n|y_n)$ is given by a random guess $1/2$, on the other hand the probability that $\mathcal{A}$ asks such a query is given by $\mathsf{q}_{\mathtt{H}_\mathtt{g}}/Q^n \leq \mathsf{q}_{\mathtt{H}_\mathtt{g}}/Q$ (for the guess of $\bar{x}_1, \ldots, \bar{x}_n$ which are uniformly distributed in $\mathbb{G}$). Hence, $\Pr[\mathsf{Win}_5^{\text{ke-g}}] \leq 1/2 + \mathsf{q}_{\mathtt{H}_\mathtt{g}}/Q$. Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}_{\text{PDHKE-KPT}}^{\text{ke-g}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\text{ke-g}}] - 1| \\ &\leq \frac{2(N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2 + \mathsf{q}_{\mathtt{H}_\mathtt{g}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^{\kappa-1}} + 2N\mathsf{Adv}_{\mathbb{G}}^{\mathtt{SEDDH}}(\kappa) + 2N(N-1)\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa).\end{aligned}$$

□

Since secret contributions $\bar{x}_i$ used in the computation of the group key are independent from the secret exponents x_i we can prove that PDHKE-MRE provides KE-security of p2p keys based on the DDH assumption.

Theorem 2. *If the DDH problem is hard in $\mathbb{G}$ then* PDHKE-MRE *provides KE-security of p2p keys and*

$$\mathsf{Adv}_{\text{PDHKE-MRE}}^{\text{ke-p}}(\kappa) \leq \frac{N(2(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2 + \mathsf{q}_{\mathtt{Se}}\mathsf{q}_{\mathtt{H}_\mathtt{p}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^{\kappa-1}} + N\mathsf{q}_{\mathtt{Se}}\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa)$$

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

Proof. In the following we show that the advantage of $\mathcal{A}$ in distinguishing some p2p key $k_{i,j}$ from some random value in $\{0,1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_4$ and denote by $\mathsf{Win}_i^{\text{ke-p}}$ 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 *TestPeer*(Π_i^s, U_j) is answered only for some instance-user pair (Π_i^s, U_j) which is fresh.

Game $\mathbf{G}_0$. This is the real execution of PDHKE-MRE in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances of users as specified in $\mathsf{Game}_{\mathcal{A},\text{PDHKE-MRE}}^{\text{ke-p},b}(\kappa)$. We assume that $\mathcal{A}$ has access to the hash queries for the hash functions $\mathtt{H}_\mathtt{g}$ and $\mathtt{H}_\mathtt{p}$ 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 $k_{i,j}$. Additionally, we exclude collisions between p2p keys $k_{i,j}$ for different (possibly malicious) U_j. Since sessions can be invoked via *Execute* and *Send* queries we obtain for the collisions of transcripts (according to the birthday paradox) the probability of at most $N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2/Q$ over all possible users. Collisions amongst k_i and collisions between k_i and any $k_{i,j}$ can be upper-bounded by $(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2/2$ using the same arguments as in Game $\mathbf{G}_1$ from the proof of Theorem 1. This upper-bound also includes collisions between different p2p keys $k_{i,j}$ computed by the same honest user U_i for different U_j. Indeed, even if U_j is malicious and defines own y_j is some rogue way (e.g. replaying the value of some other honest user) 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 user identities. That is even if U_j is malicious its identity which is among the inputs to $\mathtt{H}_\mathtt{p}$ is unique. Hence,

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

Game $\mathbf{G}_2$. In this game Δ randomly selects two user identities $U_{i^*}, U_{j^*} \in_R \mathcal{U}$ and a value $q^* \in_R [1, \mathsf{q}_{\mathsf{Se}}]$ and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ asks the *TestPeer* query on an input which is different from $(\Pi^s_{i^*}, U_{j^*})$ or $(\Pi^t_{j^*}, U_{i^*})$ in a session which is not the q^*-th session. Let E be an event that the guesses are correct. Since the guesses for the user identities (U_{i^*}, U_{j^*}) are independent from the guess of the q^*-th session we have $\Pr[\mathsf{E}] = 2/N\mathsf{q}_{\mathsf{Se}}$. Then,

$$\begin{aligned}\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2] &= \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2|\mathsf{E}]\Pr[\mathsf{E}] + \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2|\neg\mathsf{E}]\Pr[\neg\mathsf{E}]\\ &= \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_1]\frac{2}{N\mathsf{q}_{\mathsf{Se}}} + \frac{1}{2}\left(1 - \frac{2}{N\mathsf{q}_{\mathsf{Se}}}\right),\end{aligned}$$

and

$$\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_1] = \frac{N\mathsf{q}_{\mathsf{Se}}}{2}\left(\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2] - \frac{1}{2}\right) + \frac{1}{2}.$$

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 U_{i^*} and U_{j^*} that participate in that session. Note that a pair $(\Pi^s_{i^*}, U_{j^*})$ (or equivalently $(\Pi^t_{j^*}, U_{i^*})$) to which the *TestPeer* query is asked in this game must be fresh.

Game $\mathbf{G}_3$. In this game we assume that Δ chooses random $a, b \in_R \mathbb{Z}_Q$ and in the q^*-th session which is invoked via a query *Send*$(U_{i^*}, ('start', U_1, \dots, U_n))$ or *Send*$(U_{j^*}, ('start', U_1, \dots, U_n))$ s.t. $U_1, \dots, U_n$ includes both U_{i^*} and U_{j^*} proceeds as follows: In the first round Δ computes $y_{i^*} := g^{a\alpha_{i^*}}$ and $y_{j^*} := g^{b\alpha_{j^*}}$ for some random $\alpha_{i^*}, \alpha_{j^*} \in_R \mathbb{Z}_Q$. In the second round Δ follows the protocol specification treating $a\alpha_{i^*}$ and $b\alpha_{j^*}$ as the corresponding ephemeral exponents x_{i^*} and x_{j^*}. Thus, Δ will also compute $k'_{i^*,j^*} := g^{ab\alpha_{i^*}\alpha_{j^*}}$. Here it is important to mention that the freshness of $(\Pi^s_{U_{i^*}}, U_{j^*})$ or $(\Pi^t_{U_{j^*}}, U_{i^*})$ ensures that messages exchanged between the instances of U_{i^*} and U_{j^*}, in particular values y_{i^*} and y_{j^*}, are truly forwarded by $\mathcal{A}$. It is easy to see that the original distributions remain unchanged, so that $\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2] = \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_3]$.

Game $\mathbf{G}_4$. This game proceeds identical to the previous one, except that in the q^*-th session Δ chooses an additional random $c \in_R \mathbb{Z}_Q$ and the only computation which is modified is that of $k'_{i^*,j^*} := g^{c\alpha_{i^*}\alpha_{j^*}}$. The simulation in this game remains indistinguishable from the previous one incase that the DDH assumption holds in $\mathbb{G}$. Thus,

$$|\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_4] - \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_3]| \le \mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa).$$

As a consequence of Game $\mathbf{G}_1$ the inputs to $\mathtt{H}_\mathsf{p}$ used to compute p2p keys involving U_{i^*} or U_{j^*} remain unique. Since $\mathtt{H}_\mathsf{p}$ is modeled as a random function the p2p key k_{i^*,j^*} computed in this q^*-th session remains independent of any other p2p key (in the same or other sessions). Hence, the probability that $\mathcal{A}$ wins in this game without asking a query $\mathtt{H}_\mathsf{p}(k'_{i^*,j^*}, U_{i^*}|y_{i^*}, U_{j^*}|y_{j^*})$ is given by a random guess, i.e. $1/2$. On the other hand, $\mathcal{A}$ can ask such a query with probability at most $\mathsf{q}_{\mathtt{H}_\mathsf{p}}/Q$ (since k'_{i^*,j^*} is uniform in $\mathbb{G}$). Therefore, $\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_4] = 1/2 + \mathsf{q}_{\mathtt{H}_\mathsf{p}}/Q$. Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ke\text{-}p}}_{\text{PDHKE-MRE}}(\kappa) &= |2\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_0] - 1|\\ &\le \frac{N(2(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathsf{Se}})^2 + \mathsf{q}_{\mathsf{Se}}\mathsf{q}_{\mathtt{H}_\mathsf{p}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathsf{g}} + \mathsf{q}_{\mathtt{H}_\mathsf{p}})^2}{2^{\kappa-1}} + N\mathsf{q}_{\mathsf{Se}}\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa).\end{aligned}$$

□

4.5 On Security of PDHKE as a Stand-Alone Protocol

The result of Theorem 2 allows us to derive the following corollary, which is of independent interest since it addresses security of PDHKE as a stand-alone protocol.

Corollary 1. *If the DDH problem is hard in $\mathbb{G}$ then* PDHKE *as defined in Section 4.1 guarantees the KE-security of p2p keys in the random oracle model in the sense of Definition 5.*[3]

[3] Observe that our security model can be used to deal with PDHKE as a stand-alone protocol assuming that in the execution of PDHKE instances accept with empty group keys. In this case all parts of the model that explicitly deal with the computation and security of group keys become irrelevant.

4.6 Performance Limitations of PDHKE-MRE

The drawback of PDHKE-MRE despite of our optimizations is the quadratic *communication complexity*, i.e. the total number of bits communicated throughout the protocol and usually measured in the size of group (or public key) elements [29]. This complexity is due to the rather naïve secure transport of each $\bar{x}_i$ for the computation of the group key. Note that the linear communication complexity of PDHKE used to compute p2p keys is already optimal since each user has to broadcast at least one message in order to contribute to the on-demand computation of its p2p keys.

Therefore, we will try to replace the computation of the group key via MRE with an alternative process, while preserving the computation of p2p keys based on PDHKE. Since PDHKE derives p2p keys from Diffie-Hellman secrets it appears promising to search for alternative candidates amongst the family of GroupDH protocols, i.e. GKE protocols that extend the original Diffie-Hellman method.

5 GKE+P Protocols from Group Diffie-Hellman Protocols

We start by describing a generic solution that would convert any secure GroupDH protocol into a secure GKE+P protocol. Then, we address possible optimization issues.

5.1 GKE+P Compiler based on PDHKE

Let us first capture the similarities between different GroupDH protocols by providing a generalized definition of what a GroupDH protocol should mean (we define from the perspective of one session).

Definition 7 (GroupDH Protocols). *A* GroupDH protocol *is a GKE protocol amongst n users $U_1, \ldots, U_n$ such that during its execution each user U_i chooses own exponent $x_i \in_R \mathbb{Z}_Q$ and at the end computes a group element $k'_i \in \mathbb{G}$ which can be expressed as the output of $f(g, x_1, \ldots, x_n)$ for some function $f : \mathbb{G} \times \mathbb{Z}_Q^n \to \mathbb{G}$ which is specific to the protocol.*

We say that a GroupDH protocol is KE-secure *if it achieves KE-security of group keys in the sense of Definition 4 whereby considering k'_i instead of k_i and thus requiring its indistinguishability from some random element in $\mathbb{G}$ instead of some random string in $\{0,1\}^\kappa$.*[4]

The above definition of KE-secure GroupDH protocols already captures many protocols, including those from [16, 20, 24, 31, 32, 37, 39, 40].

The actual generic solution (GKE+P compiler) for obtaining a GKE+P protocol from such GroupDH protocols is to combine them with PDHKE, while ensuring independence between the exponents used in both protocols. More precisely, GKE+P compiler requires each user U_i to choose a random exponent $\bar{x}_i \in_R \mathbb{Z}_Q$ and broadcast $\bar{y}_i := g^{\bar{x}_i}$ prior to the execution of the given GroupDH protocol. If the GroupDH protocol requires each user to broadcast a message in the first round, e.g. [16, 31, 32, 39], then the compiler can also append $\bar{y}_i$ to this message, without increasing the number of rounds. After the GroupDH protocol is executed each U_i holds the secret group element k'_i. The GKE+P compiler computes $\mathtt{sid}_i := (U_1|\bar{y}_1, \ldots, U_n|\bar{y}_n)$ and derives the group key $k_i := \mathtt{H_g}(k'_i, \mathtt{sid}_i)$. On-demand, the compiler computes any $k_{i,j} := \mathtt{H_p}(\bar{y}_j^{\bar{x}_i}, U_i|\bar{y}_i, U_j|\bar{y}_j)$.

The key derivation is essentially the same as in PDHKE-MRE. The only difference is that $\mathtt{sid}_i$ is constructed from $\bar{y}_i$ instead of $y_i = g^{x_i}$ for the exponent x_i which is implicit to the original GroupDH protocol. The reason is that y_i may not be available to all users at the end of the protocol. For example, in [24, 40] only two users U_1 and U_2 compute such y_1 and y_2, whereas in [20, 37] each U_i computes y_i but sends it only to some designated subset. Of course, for the latter case it is possible to add a modification to the original protocol by requiring users to broadcast y_i; however, this contradicts to the idea of a compiler, which takes some protocol as a "black-box".

The KE-security of group keys output by our compiler follows from the KE-security of the group elements k' and can be proven similarly to Theorem 1. Note that the replacement of y_i with $\bar{y}_i$ in the computation of $\mathtt{sid}_i$ has no impact since also $\bar{y}_i$ is uniformly distributed in $\mathbb{G}$ for any honest U_i. Since the exponents x_i and $\bar{x}_i$ are independent and values $\bar{y}_i$ and $\bar{y}_j$ exchanged between any two honest users U_i and U_j are not modified during

[4] Note that Definition 4 can be easily adapted by the appropriate modification of the *Test* query.

the transmission (as required by our model) the KE-security of computed p2p keys would follow directly from Corollary 1. We omit the detailed analysis of the GKE+P compiler, which seems fairly natural.

Instead, we focus on the next challenge and investigate whether GroupDH protocols can be merged with PDHKE in order to obtain possibly more efficient GKE+P protocols than those given by our generic compiler. Can we find suitable GroupDH protocols where the implicitly used exponents $x_1, \ldots, x_n$ can be safely *re-used* for the computation of p2p keys? Intuitively, this question should be answered separately for each GroupDH protocol. Due to space limitations, we restrict our analysis to two well-known protocols from [16] and [31] that implicitly require each U_i to broadcast $y_i := g^{x_i}$ and so seem suitable at first sight for the merge with PDHKE.

5.2 PDHKE-BD is Insecure

The Burmester-Desmedt (BD) protocol from [16] is one of the best known unauthenticated GroupDH protocols. It has been formally proven KE-secure under the DDH assumption in [29]. Its technique has influenced many GKE protocols, including [2, 30]. 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 $n + 1 = 1$).

This allows each U_i to compute the secret group element

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

At first sight, BD suits for the merge with PDHKE, i.e. we would have then $k'_i := \mathtt{H}_\mathtt{g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and any $k_{i,j} := \mathtt{H}_\mathtt{p}(y_j^{x_i}, U_i|y_i, U_j|y_j)$. Unfortunately, this merge is insecure. We analyze two distinct cases based on the indices of U_i and U_j.

CASE U_i AND U_{i+1} The attack in this case is trivial since the knowledge of k' and the secret exponents of all other colluding users allows to compute $g^{x_ix_{i+1}}$. This would break the secrecy of the p2p key $k_{i,i+1}$ when derived using $g^{x_ix_{i+1}}$ for any group size $n \geq 3$. Also observe that each U_i sends $z_i = g^{x_{i+1}x_i - x_ix_{i-1}}$; thus every U_{i-1} can individually extract $g^{x_{i+1}x_i}$ and every U_{i+1} is able to compute $g^{x_ix_{i-1}}$, even without colluding with other users.

CASE U_i AND U_j In this case we consider $k_{i,j}$ (w.l.o.g. we assume that $i < j$) computed for a pair of users that do not have neighbor positions within the cycle, i.e. $j \neq i+1$. We demonstrate that also this key remains insecure if derived using $g^{x_ix_j}$. Our attack, which is not as trivial as in the previous case, works because users may collude and misbehave while attacking the secrecy of p2p keys. In particular, we assume that U_{i-2}, U_{i-1}, and U_{i+1} collude and their goal is to obtain $g^{x_ix_j}$ upon the successful execution of the protocol from the perspective of honest U_i and U_j. Due to the collusion of three users the attack works for any group size $n > 4$. The core of the attack is to let U_{i-1} broadcast $y_{i-1} := y_j$, which is possible since the communication is asynchronous and $\mathcal{A}$ can wait for the protocol message of U_j containing y_j; observe that x_j is chosen by U_j and remains unknown to the colluding users. Other malicious users U_{i-2} and U_{i+1} choose their exponents x_{i-2} and x_{i+1} truly at random. As a consequence, in the second round honest U_i broadcasts $z_i = g^{x_{i+1}x_i - x_ix_{i-1}} = g^{x_{i+1}x_i - x_ix_j}$. Then, malicious U_{i+1} can extract $g^{x_ix_j} := y_i^{x_{i+1}}/z_i$. Finally, U_{i-1} without knowing the corresponding exponents x_j and x_i has to broadcast a value of the form $z_{i-1} = g^{x_ix_{i-1} - x_{i-1}x_{i-2}} = g^{x_ix_j - x_jx_{i-2}}$ which can be easily done with the assistance of U_{i+1} that provides $g^{x_ix_j}$ and of U_{i-1} that provides $g^{x_jx_{i-2}} = y_j^{x_{i-2}}$. Thus, through their cooperation malicious users U_{i-2}, U_{i-1}, and U_{i+1} can extract $g^{x_ix_j}$ for any U_j. The above attacks works similarly even if U_{i-1} re-randomizes y_j, i.e. broadcasts $y_{i-1} = y_j^r$ for some $r \in_R \mathbb{Z}_Q$.

A simpler variant of this attack is available for a group of $n = 4$ users, e.g. $U_1, \ldots, U_4$. Assume that U_1 and U_3 are malicious and aim to obtain $g^{x_2x_4}$ (the attack would work similar for the collusion of U_2 and U_4 with respect to $g^{x_1x_3}$). First, U_1 and U_3 wait for honest U_2 and U_4 to broadcast y_2 and y_4, respectively. Then, U_1 broadcasts $y_1 := y_4g^r = g^{x_4+r}$ for some random $r \in_R \mathbb{Z}_Q$ whereas U_3 broadcasts $y_3 = g^r$. In the second round, U_2 and U_4 are first to broadcast $z_2 = g^{rx_2 - x_2(x_4+r)} = g^{x_2x_4}$ (which is the desired Diffie-Hellman secret) and $z_4 = g^{(x_4+r)x_4 - x_4 r} = g^{x_4^2}$, respectively. The knowledge of r allows U_1 to compute the well-formed $z_1 = g^{x_2(x_4+r)-(x_4+r)x_4} := z_1 y_2^r / z_4 y_4^r$ whereas U_3 can compute z_3 as usual using r as his own exponent.

This shows that BD cannot be merged with PDHKE in a secure way. Nevertheless, it can be compiled to a KE-secure GKE+P protocol as discussed in Section 5.1.

5.3 PDHKE-KPT is Secure

Here we focus on the GKE protocols proposed by Kim, Perrig, and Tsudik [31, 32], which in turn extend the less efficient construction by Steer et al. [39]. These protocols belong to a family of the so-called Tree Diffie-Hellman protocols (see also [14, 22]). We analyze whether the protocol from [31], denoted here as KPT, which is more efficient in communication than [32], can be securely merged with PDHKE.

The KPT protocol requires a special group $\mathbb{G} = \langle g \rangle$ of prime order Q, which is a group of quadratic residues modulo a safe prime $P = 2Q + 1$ with the group law defined as $ab := f(ab \mod P)$ for any $a, b \in \mathbb{G}$ where $f : \mathbb{Z}_P \mapsto \mathbb{Z}_Q$ is such that if $z \leq Q$ then $f(z) := z$, otherwise if $Q < z < P$ then $f(z) := P - z$ (see [14,31,32] for more information about $\mathbb{G}$ which equals to $\mathbb{Z}_Q$ as sets). In KPT each U_i derives the secret group element k'_i within two communication rounds (it is assumed that the sequence $U_1, \ldots, U_n$ is ordered):

Round 1 Each U_i broadcasts $y_i := g^{x_i}$ for some random $x_i \in_R \mathbb{Z}_Q$.
Round 2 U_1 computes and broadcasts $(g^{z_2}, \ldots, g^{z_{n-1}})$ whereby $z_2 := y_2^{x_1}$ and each $z_i := y_i^{z_{i-1}}$ for all $i = 3, \ldots, n-1$.

This allows each U_i to compute the common secret $k'_i := z_n$ as follows.

- U_1 computes $k'_1 := y_n^{z_{n-1}}$
- each U_i, $2 \leq i \leq n-1$ recomputes the subsequence $z_i, \ldots, z_{n-1}$ and computes $k'_i := y_n^{z_{n-1}}$; note that U_2 starts with $z_2 := y_1^{x_2}$, whereas U_i, $3 \leq i \leq n-1$, starts with $z_i := (g^{z_{i-1}})^{x_i}$ using $g^{z_{i-1}}$ received from U_1.
- U_n computes $k'_i := (g^{z_{n-1}})^{x_n}$ using $g^{z_{n-1}}$ received from U_1.

Note that each k'_i has an interesting algebraic structure

$$k'_i = g^{x_n g^{x_{n-1} g^{\cdots g^{x_3 g^{x_2 x_1}}}}} .$$

In the following we investigate the possibility of merging KPT with PDHKE, thus using exponents x_i to compute the group key $k_i := \mathtt{H_g}(k'_i, U_1|y_1, \ldots, U_n|y_n)$ and any p2p key $k_{i,j} := \mathtt{H_g}(k'_{i,j}, U_i|y_i, \ldots, U_j|y_j)$ with $k'_{i,j} = g^{x_i x_j}$. Our analysis shows that indeed this construction, which we denote PDHKE-KPT, gives us a KE-secure GKE+P protocol.

Let us first provide some intuition. Note that the only value of the form $g^{x_i x_j}$ which appears in the computations of KPT is $g^{x_1 x_2}$ (given by z_2). Nevertheless, it will be computed only by U_1 and U_2, which is fine since the p2p key should be known only to these users. Further we observe that the broadcast message of U_1 contains $g^{z_2} = g^{g^{x_1 x_2}}$ and so hides $g^{x_1 x_2}$ in the exponent (under the hardness of the DL problem). By computing $k_{1,2} := \mathtt{H_p}(g^{x_1 x_2}, U_1|y_1, U_2|y_2)$ we are able to provide independence between $k_{1,2}$ and g^{z_2} while working in the random oracle model since the corresponding *RevealPeer* query would reveal only $k_{1,2}$ and not $g^{x_1 x_2}$.

We start with the KE-security of group keys. The original KPT protocol has been proven KE-secure in [31] (see also [14]) under the classical DDH assumption. Briefly, the proof considers several hybrid games. In the l-th game, $2 \leq l \leq n$, the simulator embeds a re-randomized DDH tuple (g, g^a, g^b, g^{ab}) to simulate $g^{z_{l-1}} = g^{a\alpha_{l-1}}$, $y_i = g^{b\beta_l}$, and $z_l = g^{ab\alpha_{l-1}\beta_l}$, such that in the final game the value $z_n = k'_i$ is uniformly distributed and independent. In general we can apply a similar simulation technique, however, we should additionally take care of the special dependency $z_2 = k'_{1,2}$. The trick is first to obtain a uniform distribution of $z_2 = k'_{1,2}$ (in $\mathbb{G}$) and its independence from y_1 and y_2 using the above technique and then to compute $k_{1,2}$ completely independent from $k'_{1,2}$, in which case a reduction to the DL problem becomes possible.

Theorem 3. *If both problems DDH and DL are hard in $\mathbb{G}$ then* PDHKE-KPT *provides KE-security of group keys and*

$$\mathsf{Adv}^{\mathsf{ke\text{-}g}}_{\mathrm{PDHKE\text{-}KPT}}(\kappa) \leq \frac{2(N(\mathsf{q_{Ex}}+\mathsf{q_{Se}})^2+\mathsf{q_{H_g}})}{Q} + \frac{(\mathsf{q_{H_g}}+\mathsf{q_{H_p}})^2}{2^{\kappa-1}} + 2(N-1)\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa) + 2\mathsf{q_{H_p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa)$$

with at most $(\mathsf{q_{Ex}}+\mathsf{q_{Se}})$ *sessions being invoked via* Execute *and* Send *queries and at most* $\mathsf{q_{H_g}}$ *and* $\mathsf{q_{H_p}}$ *random oracle queries asked.*

Proof. 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 $\mathbf{G}_0,\ldots,\mathbf{G}_3$ and denote by $\mathsf{Win}^{\mathsf{ke\text{-}g}}_i$ 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 *Test*(Π^s_i) is answered only if the instance Π^s_i is fresh.

Game $\mathbf{G}_0$. This is the real execution of PDHKE-KPT in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances, as specified in $\mathsf{Game}^{\mathsf{ke\text{-}g},b}_{\mathcal{A},\mathrm{PDHKE\text{-}KPT}}(\kappa)$. Additionally, we provide $\mathcal{A}$ with the access to the hash queries for $\mathtt{H_g}$ and $\mathtt{H_p}$, modeled as random oracles.

Game $\mathbf{G}_1$. In this game we exclude the same collisions on transcripts, group and p2p keys as in Game $\mathbf{G}_1$ from the proof of Theorem 1. Since the key derivation mechanisms are exactly the same we can re-use those arguments to obtain

$$|\Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_1] - \Pr[\mathsf{Win}^{\mathsf{ke\text{-}g}}_0]| \leq \frac{N(\mathsf{q_{Ex}}+\mathsf{q_{Se}})^2}{Q} + \frac{(\mathsf{q_{H_g}}+\mathsf{q_{H_p}})^2}{2^\kappa}.$$

Game $\mathbf{G}_{2,l}$ for $l = 2,\ldots,n$. Each Game $\mathbf{G}_{2,l}$ is composed of two hybrid Sub-Games $\mathbf{G}_{2,l,1}$ and $\mathbf{G}_{2,l,2}$ (note that the index l starts with 2 for the convenience of presentation).

Sub-Game $\mathbf{G}_{2,l,1}$. In this game we assume that Δ is given access to some private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$. Further, Δ chooses some random $a, b \in_R \mathbb{Z}_Q$ and in response to each *Execute*$(U_1,\ldots,U_n)$ query proceeds as in the previous game except for the following rules (here we remark that $\mathbb{G}$ and $\mathbb{Z}_Q$ are equal as sets [14]):

Rule 1 If $l = 2$: Simulate $y_1 := g^{a\alpha_1}$ and $y_2 := g^{b\beta_2}$ for some random $\alpha_1, \beta_2 \in_R \mathbb{Z}_Q$ and compute all further y_i according to the protocol specification, thus using truly random exponents $x_i \in_R \mathbb{Z}_Q$. This also implies the simulation of $z_2 := g^{ab\alpha_1\beta_2}$. All further z_i, $3 \leq i \leq n-1$ are computed as specified in the protocol as $z_i := g^{z_{i-1}x_i}$. Note that Δ can easily compose the message $g^{z_2},\ldots,g^{z_{n-1}}$ and compute the group key.

Rule 2 If $l > 2$: Simulate $y_l := g^{b\beta_l}$ using $\beta_l \in_R \mathbb{Z}_Q$ and compute all other y_i using truly random exponents $x_i \in_R \mathbb{Z}_Q$. For each $2 \leq i < l$ simulate $g^{z_i} := g^{a\alpha_i}$ (this simulation can be done without explicitly computing z_i and it applies for the first time in Sub-Game $\mathbf{G}_{2,3,1}$). Simulate $z_l := g^{ab\alpha_{l-1}\beta_l}$, whereas all further z_i, $l < i \leq n-1$ are computed as specified in the protocol as $z_i := g^{z_{i-1}x_i}$. Note that Δ can still easily compose the message $g^{z_2},\ldots,g^{z_{n-1}}$ and compute the group key.

Note that these simulation rules ensure that the product ab is embedded only in the computation of z_l and all z_i with $2 \leq i < l$ are independent due to the use of different blinding factors α_i. Note also that the indices of β are bound to the sub-game index l which starts with 2 (hence, β_1 does not exist).

Additionally, in response to a query *Peer*(Π^s_i, U_j) the simulator proceeds as follows. First, Δ computes $k'_{i,j}$ (w.l.o.g. we consider only the case of $i < j$ since $k'_{i,j} = k'_{j,i}$) depending on the following distinct cases implied by the rules above:

Case R1.1 If $l = 2$ and $j = 2$ then we have $k'_{i,j} = k'_{1,2}$ which will be set equal to z_2 (note that this may occur only in Sub-Game $\mathbf{G}_{2,2,1}$ where $z_2 = g^{ab\alpha_1\beta_2}$ as explained in Rule 1);

Case R1.2 If $l = 2$ and $j > 2$ then we have: any $k'_{1,j} := g^{a\alpha_1 x_j}$, any $k'_{2,j} := g^{b\alpha_2 x_j}$, and any other $k'_{i,j} = g^{x_i x_j}$ (this is consistent to the simulation of y_1, y_2, and every other y_i in case that $l = 2$ as explained in Rule 1);

Case R2.1 If $l > 2$ and $j = 2$ then the computation of $k'_{i,j} = k'_{1,2}$ is omitted (the p2p key $k_{i,j} = k_{1,2}$ will be computed from $\mathtt{H}'$ as described below);

Case R2.2 If $l > 2$ and $j > 2$ then we have: any $k'_{l,j} := g^{b\beta_l x_j}$, any $k'_{i,l} := g^{x_i b\beta_l}$, and any other $k'_{i,j} = g^{x_i x_j}$ (this is consistent to the simulation of y_l and every other y_i in case that $l = 2$ as explained in Rule 2);

Then, $\varDelta$ derives $k_{i,j} := \mathtt{H}_\mathtt{p}(k'_{i,j}, U_i|y_i, U_j|y_j)$ with the only exception that if $l > 2$ and $j = 2$ then $k_{i,j} = k_{1,2} := \mathtt{H}'(U_1|y_1, U_2|y_2)$ that is completely independent of $k'_{1,2}$. Note that $\mathtt{H}'$ is used to compute $k_{1,2}$ in each Sub-Game $\mathbf{G}_{2,l,1}$ with $l \geq 3$. Briefly, starting from Sub-Game $\mathbf{G}_{2,3,1}$ we simulate $g^{z_2} = g^{a\alpha_2}$ using some random $\alpha_2 \in_R \mathbb{Z}_Q$. If we would still simulate $k'_{1,2} = z_2$ (as we do in $\mathbf{G}_{2,2,1}$) then the simulation would no more be perfect since $k'_{1,2}$ must be of the form $g^{x_1 x_2}$ (according to Rule 2). Using the fact that $k'_{1,2}$ cannot be revealed we ignore it in the computation of $k_{1,2}$, thus still allowing the simulation without explicit computation of z_2. This would have an impact on the indistinguishability between the hybrid sub-games $\mathbf{G}_{2,2,2}$ and $\mathbf{G}_{2,3,1}$ as discussed below.

Sub-Game $\mathbf{G}_{2,l,2}$. In this game we assume that $\varDelta$ holds an additional random value $c \in_R \mathbb{Z}_Q$. It performs the simulation exactly as in Sub-Game $\mathbf{G}_{2,l,1}$ except that: In Rule 1 it simulates $z_2 = g^{c\alpha_1\beta_2}$, in Rule 2 it simulates $z_l := g^{c\alpha_{l-1}\beta_l}$ with the obvious consequence for the computation of $k'_{1,2} = z_2$ in case that Rule 1 applies. Observe, that in this way the simulator can embed DDH tuples (g, g^a, g^b, g^{ab}) in $\mathbf{G}_{2,l,1}$ and (g, g^a, g^b, g^c) in $\mathbf{G}_{2,l,1}$. Since the re-randomized instances of the DDH problem are embedded only in z_l we can upper-bound $|\Pr[\mathsf{Win}^{\text{ke-g}}_{2,l,2}] - \Pr[\mathsf{Win}^{\text{ke-g}}_{2,l,1}]| \leq \mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa)$.

Furthermore, $\Pr[\mathsf{Win}^{\text{ke-g}}_{2,l+1,1}] = \Pr[\mathsf{Win}^{\text{ke-g}}_{2,l,2}]$ for the only exception in the transition from $\mathbf{G}_{2,2,2}$ to $\mathbf{G}_{2,3,1}$ due to the use of $\mathtt{H}'$ in the computation of $k_{1,2}$. For this we observe that in $\mathbf{G}_{2,2,2}$ we have $z_2 = k'_{1,2} = g^{c\alpha_1\beta_2}$ which is uniform in $\mathbb{G}$ and independent of $y_1 = g^{a\alpha_1}$ and $y_2 = g^{b\beta_2}$, whereas in $\mathbf{G}_{2,3,1}$ we have $z_2 = g^{a\alpha_2}$, $y_1 = g^{x_1}$, $y_2 = g^{x_2}$ for some random $x_1, x_2 \in_R \mathbb{Z}_Q$ (the distributions of y_1 and y_2 in both games are identical) and $k'_{1,2}$ is no longer used since the p2p key $k_{1,2}$ is derived via $\mathtt{H}'(U_i|y_i, U_j|y_j)$. Obviously, both sub-games remain indistinguishable unless $\mathcal{A}$ asks a query of the form $\mathtt{H}_\mathtt{p}(k'_{1,2}, U_1|y_1, U_2|y_2)$ and the only value that may leak some information about $k'_{1,2}$ in $\mathbf{G}_{2,3,1}$ is g^{z_2}. However, for this case we can construct a solver for the DL problem that would embed some value V for which it has to compute $\log_g V$ in Rule 2 in the simulation of $\mathbf{G}_{2,3,1}$ by defining $g^{z_2} = V$ and so preserving its distribution as there exists obviously some unknown random α_2 such that $\log_g V = a\alpha_2$. If the mentioned hash query is asked then the DL solver can output $\log_g V$. Note that in $\mathbf{G}_{2,3,1}$ the knowledge of z_2 is not necessary for the simulation, i.e. the DL solver can still compute $z_3 := V^{b\beta_3}$. Thus, $|\Pr[\mathsf{Win}^{\text{ke-g}}_{2,3,1}] - \Pr[\mathsf{Win}^{\text{ke-g}}_{2,2,2}]| \leq \mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa)$.

Further, observe that by construction in Sub-Game $\mathbf{G}_{2,2,1}$ (the first sub-game in the sequence, since the index l starts with 2) the distributions remain identical to the real protocol execution, implying $\Pr[\mathsf{Win}^{\text{ke-g}}_{2,2,1}] = \Pr[\mathsf{Win}^{\text{ke-g}}_{0}]$. And since l is a running variable from 2 to n and $n \leq N$ we can upper-bound the probability difference between Game $\mathbf{G}_{2,n}$ (which ends with Sub-Game $\mathbf{G}_{2,n,2}$) and $\mathbf{G}_1$ as follows:

$$|\Pr[\mathsf{Win}^{\text{ke-g}}_{2,n}] - \Pr[\mathsf{Win}^{\text{ke-g}}_{1}]| \leq \sum_{l=2}^{N} \mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa) + \mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa) = (N-1)\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa) + \mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa).$$

Observe that at the end of Game $\mathbf{G}_{2,n}$ we have $k'_i = z_n = g^{c\alpha_{n-1}\beta_n}$ which is uniform in $\mathbb{G}$ and independent from $g^{z_{n-1}}$ and y_n (the independence from other g^{z_i}, $2 \leq i < n-1$ and y_i, $1 \leq i < n$ is implied by the hybrid games).

According to Game $\mathbf{G}_1$ the transcript $U_1|y_1, \ldots, U_n|y_n$ used by some honest user U_i in addition to k'_i as input to $\mathtt{H}_\mathtt{g}$ for the computation of the group key k_i is unique. According to the instance freshness, the instance $\varPi^s_i$ to which the *Test* query is asked must belong to an honest user. Since $\mathtt{H}_\mathtt{g}$ is modeled as a random oracle the probability that $\mathcal{A}$ wins without querying $\mathtt{H}_\mathtt{g}(k'_i, y_1, U_1|y_1, \ldots, U_n|y_n)$ is given by a random guess $1/2$, on the other hand the probability that $\mathcal{A}$ asks such a query is given by $\mathsf{q}_{\mathtt{H}_\mathtt{g}}/Q$ (for the guess of k'_i which is uniform in $\mathbb{G}$). Hence, $\Pr[\mathsf{Win}^{\text{ke-g}}_{1}] \leq 1/2 + \mathsf{q}_{\mathtt{H}_\mathtt{g}}/Q$. Summarizing the above equations we obtain a negligible advantage:

$$\begin{aligned}\mathsf{Adv}^{\text{ke-g}}_{\text{PDHKE-KPT}}(\kappa) &= |2\Pr[\mathsf{Win}^{\text{ke-g}}_{0}] - 1| \\ &\leq \frac{2(N(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2 + \mathsf{q}_{\mathtt{H}_\mathtt{g}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^{\kappa-1}} + 2(N-1)\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa) + 2\mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa).\end{aligned}$$

□

Finally, we prove that on-demand p2p keys computed in PDHKE-KPT are also KE-secure. In general we can follow the proof of Theorem 2 based on the DDH assumption, however, we have also to take care of the special case $(i, j) = (1, 2)$. Observe that if $k_{1,2}$ becomes a subject of the attack then U_1 and U_2 must be honest, in which case we can still apply the above trick.

Theorem 4. *If both problems DDH and DL are hard in* $\mathbb{G}$ *then* PDHKE-KPT *provides KE-security of p2p keys and*

$$\mathsf{Adv}^{\mathsf{ke\text{-}p}}_{\text{PDHKE-KPT}}(\kappa) \leq \frac{N(2(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathsf{Se}})^2 + \mathsf{q}_{\mathsf{Se}}\mathsf{q}_{\mathtt{H_p}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H_g}} + \mathsf{q}_{\mathtt{H_p}})^2}{2^{\kappa-1}} + N\mathsf{q}_{\mathsf{Se}}\left(\mathsf{Adv}^{\mathtt{DDH}}_{\mathbb{G}}(\kappa) + \mathsf{q}_{\mathtt{H_p}}\mathsf{Succ}^{\mathtt{DL}}_{\mathbb{G}}(\kappa)\right)$$

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

Proof. In the following we show that the advantage of $\mathcal{A}$ in distinguishing some p2p key $k_{i,j}$ from some random value in $\{0, 1\}^\kappa$ is negligible. We construct a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_5$ and denote by $\mathsf{Win}^{\mathsf{ke\text{-}p}}_i$ 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 *TestPeer*(Π^s_i, U_j) is answered only for some instance-user pair (Π^s_i, U_j) which is fresh.

Game $\mathbf{G}_0$. This is the real execution of PDHKE-KPT in which a simulator Δ truly answers all queries of $\mathcal{A}$ on behalf of the instances of users as specified in $\mathsf{Game}^{\mathsf{ke\text{-}p},b}_{\mathcal{A},\text{PDHKE-KPT}}(\kappa)$. We also give $\mathcal{A}$ access to the hash queries for the hash functions $\mathtt{H_g}$ and $\mathtt{H_p}$ modeled as random oracles.

Game $\mathbf{G}_1$. In this game we exclude the same collisions on transcripts, group and p2p keys as in Game $\mathbf{G}_1$ from the proof of Theorem 2. Since the key derivation mechanisms are exactly the same we can apply the same arguments and get

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

Game $\mathbf{G}_2$. In this game Δ randomly selects two user identities $U_{i^*}, U_{j^*} \in_R \mathcal{U}$ and a value $q^* \in_R [1, \mathsf{q}_{\mathsf{Se}}]$ and aborts the simulation (setting bit b' at random) if $\mathcal{A}$ asks the *TestPeer* query on an input which is different from $(\Pi^s_{i^*}, U_{j^*})$ or $(\Pi^t_{j^*}, U_{i^*})$ in a session which is not the q^*-th session. Based on the arguments in Game $\mathbf{G}_2$ from the proof of Theorem 2 we get

$$\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_1] = \frac{N\mathsf{q}_{\mathsf{Se}}}{2}\left(\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2] - \frac{1}{2}\right) + \frac{1}{2}.$$

Game $\mathbf{G}_3$. In this game we assume that Δ chooses random $a, b \in_R \mathbb{Z}_Q$ and in response to the q^*-th session invoked via *Send*$(U_{i^*}, ('start', U_1, \ldots, U_n))$ or *Send*$(U_{j^*}, ('start', U_1, \ldots, U_n))$ with $U_1, \ldots, U_n$ including U_{i^*} and U_{j^*} (due to the guesses of the previous game) proceeds as follows:

Case $(i^*, j^*) = (1, 2)$ In the first round it simulates $y_1 := g^{a\alpha_1}$ and $y_2 := g^{b\alpha_2}$ for some random $\alpha_1, \alpha_2 \in_R \mathbb{Z}_Q$. All further values y_i on behalf of honest users are computed truly according to the protocol specification. Since $i^* = 1$ the simulator must also compose the message $g^{z_2}, \ldots, g^{z_{n-1}}$. For this it proceeds according to the protocol specification. Note that this implies $z_2 := g^{ab\alpha_1\alpha_2}$.
In this case the simulation leads to the on-demand computation of $k'_{1,2} := g^{ab\alpha_1\alpha_2} = z_2$; note that dealing in the authentication links model all protocol messages exchanged between the instances of U_1 and U_2, in particular values y_1 and y_2, are truly forwarded by $\mathcal{A}$ and no framing attacks on U_1 and U_2 take place.

Case $(i^*, j^*) \neq (1, 2)$ This case is the same as the previous one. However, there is some subtle difference. If U_1 in the q^*-th session is honest then Δ has still to simulate the message $g^{z_2}, \ldots, g^{z_{n-1}}$. In this case it would also compute $z_2 := y_2^{x_1}$ for some random exponent x_1 chosen on behalf of U_1 (whereby y_2 may also be chosen by $\mathcal{A}$). However, also in this case Δ will compute on-demand $k'_{i^*,j^*} := g^{ab\alpha_{i^*}\alpha_{j^*}}$.

Obviously, the simulation preserves the original distributions so that $\Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_2] = \Pr[\mathsf{Win}^{\mathsf{ke\text{-}p}}_3]$.

Game $\mathbf{G}_4$. This game proceeds identical to the previous one, except that Δ chooses an additional random $c \in_R \mathbb{Z}_Q$ and modifies the simulation in the q^*-th session as follows. In case that $(i^*, j^*) = (1, 2)$ it embeds c into the computation of $z_2 = g^{c\alpha_1\alpha_2}$ and on-demand $k'_{1,2} := g^{c\alpha_1\alpha_2} = z_2$ and if $(i^*, j^*) \neq (1, 2)$ then c is embedded only in $k'_{i^*,j^*} := g^{c\alpha_{i^*}\alpha_{j^*}}$. Both games remain indistinguishable in case that the DDH assumption holds in $\mathbb{G}$. Thus,

$$|\Pr[\mathsf{Win}_4^{\mathsf{ke\text{-}p}}] - \Pr[\mathsf{Win}_3^{\mathsf{ke\text{-}p}}]| \le \mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa).$$

This game implies the uniformity of k'_{i^*,j^*} in $\mathbb{G}$ unless $(i^*, j^*) = (1, 2)$. In the latter case both values $k'_{1,2}$ and z_2 are equal but uniform in $\mathbb{G}$.

Game $\mathbf{G}_5$. In this game we assume that Δ is given access to a private random oracle $\mathtt{H}' : \{0,1\}^* \to \{0,1\}^\kappa$ and change only the simulation of the on-demand p2p key k_{i^*,j^*}. Namely, we let Δ compute $k_{i^*,j^*} := \mathtt{H}'(U_{i^*}|y_{i^*}, U_{j^*}|y_{j^*})$, thus completely independent from k'_{i^*,j^*}. Note that since both users U_{i^*} and U_{j^*} are honest as required by the definition of the instance-user freshness the transcript $(U_{i^*}|y_{i^*}, U_{j^*}|y_{j^*})$ is unique according to Game $\mathbf{G}_1$.

Both games remain indistinguishable unless $\mathcal{A}$ queries $\mathtt{H}_\mathtt{p}(k'_{i^*,j^*}, U_{i^*}|y_{i^*}, U_{j^*}|y_{j^*})$. From the previous game we know that for the case $(i^*, j^*) \neq (1, 2)$ the value k'_{i^*,j^*} is uniform in $\mathbb{G}$. Therefore, the probability that $\mathcal{A}$ asks the mentioned hash query in this case is upper-bounded by a random guess of k'_{i^*,j^*}, i.e. by $\mathsf{q}_{\mathtt{H}_\mathtt{p}}/Q$. However, in case $(i^*, j^*) = (1, 2)$ we have to exclude the possibility of $\mathcal{A}$ obtaining $k'_{1,2}$ from g^{z_2} (this is the only value which may leak information about $k'_{1,2}$), which is implied by the hardness of the DL problem. Thus,

$$|\Pr[\mathsf{Win}_5^{\mathsf{ke\text{-}p}}] - \Pr[\mathsf{Win}_4^{\mathsf{ke\text{-}p}}]| \le \frac{\mathsf{q}_{\mathtt{H}_\mathtt{p}}}{Q} + \mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{DL}}(\kappa).$$

Since in this game k_{i^*,j^*} is given by the output of the private random oracle $\mathtt{H}'$ it follows that $\Pr[\mathsf{Win}_5^{\mathsf{ke\text{-}p}}] = 1/2$. Summarizing the above equations we obtain a negligible advantage

$$\begin{aligned}\mathsf{Adv}_{\text{PDHKE-KPT}}^{\mathsf{ke\text{-}p}}(\kappa) &= |2\Pr[\mathsf{Win}_0^{\mathsf{ke\text{-}p}}] - 1| \\ &\le \frac{N(2(\mathsf{q}_{\mathtt{Ex}} + \mathsf{q}_{\mathtt{Se}})^2 + \mathsf{q}_{\mathtt{Se}}\mathsf{q}_{\mathtt{H}_\mathtt{p}})}{Q} + \frac{(\mathsf{q}_{\mathtt{H}_\mathtt{g}} + \mathsf{q}_{\mathtt{H}_\mathtt{p}})^2}{2^{\kappa-1}} + N\mathsf{q}_{\mathtt{Se}}\left(\mathsf{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\kappa) + \mathsf{q}_{\mathtt{H}_\mathtt{p}}\mathsf{Succ}_{\mathbb{G}}^{\mathtt{DL}}(\kappa)\right).\end{aligned}$$

□

6 Performance Comparison and Discussion

In Table 1 we present a brief comparison of the complexity of the mentioned GKE+P solutions. 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 $|x_i| \gg n$). From the latter we exclude the costs needed to compute a Diffie-Hellman secret $k'_{i,j}$ that requires constantly one exponentiation per each U_j. For the GKE+P compiler from Section 5.1 with the prefix '+' we indicate the increase to the original costs of the given GroupDH protocol when combined with PDHKE; we also mention the compiled GKE+P version of the BD protocol as a special case. Note that the PDHKE-KPT protocol has asymmetric costs, depending on the position of U_i in the ordered sequence $U_1, \dots, U_n$; this may have benefits in groups with heterogeneous devices.

Table 1. Communication and Computation Costs of Introduced GKE+P Protocols

GKE+P Protocols	Communication (in $\log Q$ bits)	Computation (in mod. exp. per U_i)
PDHKE-MRE	$n^2 + n$	$2n$
GKE+P compiler	$+n$	$+1$
BD (as a special case)	$3n$	3
PDHKE-KPT	$2n - 2$	$n + 2 - i$ ($2n - 2$ for U_1)

From Table 1 we highlight that PDHKE-KPT has better communication complexity than the compiled version of the BD protocol, but (in general much) worse computation complexity. The same holds for the original

KPT and BD protocols. Therefore, we do not claim that GroupDH protocols when merged with PDHKE in an optimized way (via exponent re-use) would result in more efficient constructions compared to other protocols obtained via our GKE+P compiler. Nevertheless, with PDHKE-KPT we could show that there exist GKE protocols that provide the property of non-interactive, on-demand computation of p2p keys almost "for free" (if one neglects the computation costs needed for the derivation of keys then the costs of PDHKE-KPT from Table 1 are identical to those of KPT).

7 Adding Authentication to GKE+P Protocols

Yet, we were assuming that described GKE+P protocols are executed over authenticated links and focused on the KE-security of the computed group and p2p keys. On the other hand, it is well-known that any KE-secure GKE protocol can be converted into an AKE-secure protocol (preserving its forward secrecy) using the classical and inexpensive compilation technique from [29] which assumes for each user U_i a long-lived digital signature key pair (sk_i, pk_i) such that in the preliminary protocol round users exchange their nonces r_i and then sign each l-th round message m_l concatenated with $U_1|r_1|\dots|U_n|r_n$ prior to the transmission. The EUF-CMA security of the digital signature and the negligible collision probability for the session nonces protects against impersonation and replay attacks.

The following theorem shows that this technique is also sufficient to obtain AKE-security of group and p2p keys in GKE+P protocols.

Theorem 5. *If $\mathcal{P}$ is a GKE+P protocol that provides KE-security of group/p2p keys then $\mathcal{P}$ compiled with the technique from [29] results in a GKE+P protocol $\mathcal{P}'$ that provides AKE-security of group/p2p keys.*

Proof Idea: Theorem 5 can be proven in two steps (one for group keys, another one for p2p keys) using the same strategy as in the proof of [29, Theorem 2]. Briefly, in each of the both steps the proof first eliminates signature forgeries and replay attacks and then constructs an adversary $\mathcal{A}$ against the KE-security of group/p2p keys that interacts with the user instances and also simulates the additional authentication steps while answering the queries of an adversary $\mathcal{A}'$ against the AKE-security of group/p2p keys. In case of group keys $\mathcal{A}$ will need to guess the session in which the *Test*(Π_i^s) query will be asked in order to simulate the protocol execution in that session through the authentication of the transcript, which $\mathcal{A}$ obtains initially via own *Execute* query. In case of p2p keys $\mathcal{A}$ will need to guess the session in which the *TestPeer*(Π_i^s, U_j) query will be asked *and* two corresponding identities U_i and U_j of honest users in order to add authentication to their messages, which $\mathcal{A}$ obtains by relaying the *Send* queries of $\mathcal{A}'$. We omit the details.

8 Conclusion

We discussed the enrichment of GKE protocols with the property of non-interactive, on-demand derivation of peer-to-peer keys, which allows for the establishment of a secure group channel and up to n independently secure peer-to-peer channels through a single run of the protocol. We extended the standard GKE security model capturing independence of group and p2p keys as well as possible collusion attacks against the secrecy of the latter and proposed several provably secure solutions with varying efficiency. With PDHKE-KPT we demonstrated the existence of GKE protocols that implicitly allow derivation of p2p keys without any increase of their original communication complexity. Future work may include consideration of the optional insider threats against the group keys computed in GKE+P protocols in the spirit of [12,13,28]. Another interesting direction is to investigate to what extent (x_i, g^{x_i}) often computed in GroupDH protocols can be used as key pairs in digital signatures, public-key encryption schemes, etc.

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. Abdalla, D. Catalano, C. Chevalier, and D. Pointcheval. Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework. In *CT-RSA 2008*, LNCS 4964, pp. 335–351. Springer, 2008.
4. M. Bellare, A. Boldyreva, and J. Staddon. Randomness Re-use in Multi-Recipient Encryption Schemeas. In *PKC 2003*, LNCS 2567, pp. 85–99, Springer 2003.
5. M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols.In *ACM STOC'98*, pp. 419–428. ACM, 1998.
6. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *EUROCRYPT 2000*, LNCS 1807, pp. 139–155. Springer, 2000.
7. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *CRYPTO 1993*, LNCS 773, pp. 232–249. Springer, 1994.
8. 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.
9. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
10. 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.
11. 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.
12. 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.
13. E. Bresson and M. Manulis. Contributory Group Key Exchange in the Presence of Malicious Participants. *IET Inf. Sec.*, 2(3):85–93, 2008.
14. E. Bresson and M. Manulis. Securing Group Key Exchange against Strong Corruptions. In *ACM ASIACCS'08*, pp. 249–260. ACM Press, 2008.
15. 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.
16. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT 1994*, LNCS 950, pp. 275–286. Springer, 1994.
17. 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.
18. R. Canetti and H. Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In *EUROCRYPT 2002*, LNCS 2332, pp. 337–351. Springer, 2002.
19. 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.
20. Y. Desmedt and T. Lange. Revisiting Pairing Based Group Key Exchange. In *FC 2008*, LNCS 5143, pp. 53–68. Springer, 2008.
21. W. Diffie and M. E. Hellman. New Directions in Cryptography *IEEE Tran. on Inf. Th.*, 22(6):644–654, 1976.
22. R. Dutta, R. Barua, and P. Sarkar. Provably Secure Authenticated Tree Based Group Key Agreement. In *ICICS 2004*, LNCS 3269, pp. 92–104. Springer, 2004.
23. T. E. Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO 1984*, LNCS 196, pp. 10–18. Springer, 1985.
24. I. Ingemarsson, D. T. Tang, and C. K. Wong. A Conference Key Distribution System. *IEEE Tran. on Inf. Th.*, 28(5):714–719, 1982.
25. S. Jarecki, J. Kim, and G. Tsudik. Authentication for Paranoids: Multi-party Secret Handshakes. In *ACNS 2006*, LNCS 3989, pp. 325–339. Springer, 2006.
26. S. Jarecki, J. Kim, and G. Tsudik. Group Secret Handshakes Or Affiliation-Hiding Authenticated Group Key Agreement. In *CT-RSA 2007*, LNCS 4377, pp. 287–308. Springer, 2007.
27. I. R. Jeong and D. H. Lee. Parallel Key Exchange. *J. of Univ. Comp. Sci.*, 14(3):377–396, 2008.
28. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *ACM CCS'05*, pp. 180–189. ACM Press, 2005.
29. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO 2003*, LNCS 2729, pp. 110–125. Springer, 2003.

30. 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.
31. Y. Kim, A. Perrig, and G. Tsudik. Group Key Agreement Efficient in Communication. *IEEE Tran. on Comp.*, 53(7):905–921, 2004.
32. Y. Kim, A. Perrig, and G. Tsudik. Tree-Based Group Key Agreement. *ACM Trans. on Inf. and Syst. Sec.*, 7(1):60–96, 2004.
33. K. Kurosawa. Multi-Recipient Public-Key Encryption with Shortened Ciphertext. In *PKC 2002*, LNCS 2274, pp. 48–63. Springer, 2002.
34. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger Security of Authenticated Key Exchange. In *ProvSec 2007*, LNCS 4784, pp. 1–16. Springer, 2007.
35. M. Manulis. Security-Focused Survey on Group Key Exchange Protocols. *Cryptology ePrint Archive*, Report 2006/395, 2006.
36. 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.
37. 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.
38. V. Shoup. On Formal Models for Secure Key Exchange (Version 4). TR RZ 3120, IBM Research, 1999.
39. 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.
40. 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.
41. S. Wolf. *Information-Theoretically and Computationally Secure Key Agreement in Cryptography*. PhD thesis, ETH Zürich, 1999.