Towards Provably Secure Group Key Agreement Building on Group Theory

Jens-Matthias Bohli[1], Benjamin Glas[2], and Rainer Steinwandt[3]

[1] Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik, Am Fasanengarten 5, 76128 Karlsruhe, Germany,
`bohli@ira.uka.de`

[2] Institut für Technik der Informationsverarbeitung, Fakultät für Elektrotechnik & Informationstechnik, Engesserstraße 5, 76128 Karlsruhe, Germany,
`glas@itiv.uka.de`

[3] Department of Mathematical Sciences, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA,
`rsteinwa@fau.edu`

Abstract. Known proposals for key establishment schemes based on combinatorial group theory are often formulated in a rather informal manner. Typically, issues like the choice of a session identifier and parallel protocol executions are not addressed, and no security proof in an established model is provided. Successful attacks against proposed parameter sets for braid groups further decreased the attractivity of combinatorial group theory as a candidate platform for cryptography.

We present a 2-round group key agreement protocol that can be proven secure in the random oracle model if a certain group-theoretical problem is hard. The security proof builds on a framework of Bresson et al., and explicitly addresses some issues concerning malicious insiders and also forward secrecy. While being designed as a tool for basing group key agreement on non-abelian groups, our framework also yields a 2-round group key agreement basing on a Computational Diffie-Hellman assumption.

Keywords: group key establishment, provable security, conjugacy problem, automorphisms of groups

1 Introduction

While in recent years cryptographic proposals building on combinatorial group theory, in particular braid groups, proliferated, repeated cryptanalytic success also diminished the initial optimism on the subject significantly. Dehornoy's paper [15] gives a good survey on the state of the subject, and evidently significant research is still needed to reach a definite conclusion on the cryptographic potential of braid groups. As far as key establishment is concerned, especially an idea of Anshel et al. [2, 1] received a lot of attention (e. g., [16, 19, 27]). Several further ideas for deriving a key establishment scheme from combinatorial group theory have been put forward, including the work in [22, 23, 26, 28, 25]. Unfortunately, to the best of our knowledge for none of these proposals a modern security analysis in an established cryptographic framework like [5, 3, 24, 4, 11, 12] has been carried out. It should be mentioned, however, that the 2-party construction considered by Catalano et al. in [13] seems suitable for a non-abelian setting, but no further exploration in this direction is known to us.

One approach to build a key establishment protocol on non-abelian groups is to prove a scheme secure against passive adversaries, followed by applying a generic compiler that establishes stronger security guarantees (cf. [21], for instance). In this contribution we focus on group[1] key establishment. So far, the only known proposals for basing a group key establishment on non-abelian groups we are aware of are due to Lee et al. [23] and Grigoriev and Ponomarenko [18]. Unfortunately the former builds on ideas from a two-party protocol presented in [22], so Cheon and Jun's polynomial time solution to the Braid Diffie-Hellman conjugacy problem [14] impairs the attractivity of Lee et al.'s scheme. Grigoriev and Ponomarenko use a different approach to build a group key establishment. They build on ideas from [2, 1] and make repeated use of a 2-party protocol.

As an intermediate step, we build a key encapsulation mechanism from a group theoretic problem and then construct a group key establishment protocol on top. We do not follow the indicated 2-step approach (proving security in the passive case, followed by, say, applying the compiler of Katz and Yung [21])—aiming at a 2-round solution, a direct design approach appears to be no less attractive. While we cannot give a concrete non-abelian instance of our scheme, a concrete protocol can be derived from a Computational Diffie-Hellman (CDH) assumption in a cyclic group. In a sense, our approach can be seen in the spirit of [17], which takes a similar effort to identify requirements on finite non-abelian groups that allow to implement a provably IND-CCA secure public key encryption scheme.

Our security proof makes use of an existentially unforgeable signature scheme and the random oracle, and it is fair to ask whether there is really a need for another protocol in such a setting. For instance, in terms of communication complexity Boyd and Nieto's 1-round protocol from [9] certainly can be seen as superior to the 2-round proposal below. However, the latter protocol lacks forward secrecy, and to our knowledge it is not known whether a one-round protocol achieving forward secrecy can be constructed at all [8]. Moreover, we aim at a 2-round protocol offering security guarantees in the presence of malicious insiders. Currently the treatment of malicious insiders in group key establishment receives increasing attention—including the work in [7, 20, 6]. At the current state of the art, to design a 2-round protocol with security guarantees against malicious insiders and offering forward secrecy, a "one step" design strategy building on a random oracle and a signature scheme still appears to be fair.

[1] Unfortunately, in this paper the term *group* has to express two different meanings; here it refers to a set of principals.

2 Preliminaries

Giving a general introduction to the existing models for group key establishment is beyond the scope of this paper, and we refer to the standard reference [10] for this. Moreover, the background in group theory needed for describing our protocol is extremely modest. Hence, we restrict to recalling some details of the cryptographic proof model used for the security proof below. A more detailed discussion of this model can be found in [4, 11, 7].

2.1 Security model

Participants. A finite set $\mathcal{P}$ of probabilistic polynomial time (ppt) Turing machines U_i models the users that constitute the (potential) protocol participants. Each user $U_i \in \mathcal{P}$ may execute a polynomial number of protocol instances in parallel. We denote the instance $s \in \mathbb{N}$ of principal $U_i \in \mathcal{P}$ by Π_i^s. Each instance Π_i^s may be taken for a process executed by U_i and has assigned seven variables state_i^s, sid_i^s, pid_i^s, sk_i^s, term_i^s, used_i^s and acc_i^s:

- used_i^s is initialized with false and set to true as soon as the instance begins a protocol run triggered by a call to the Execute-oracle or a call to the Send-oracle (see below);
- state_i^s stores the state information during the protocol execution;
- term_i^s is initialized with false and set to true when the execution has terminated;
- sid_i^s holds the (non-secret) session identifier that serves as identifier for the session key sk_i^s and is initialized with a distinguished NULL value—the adversary has access to all sid_i^s-values;
- pid_i^s stores the set of user identities that Π_i^s aims at establishing a key with—it also includes U_i itself;
- acc_i^s is initialized with false and set to true if the protocol execution terminated successfully (i. e., the principal accepted the session key for use with users pid_i^s in session sid_i^s);
- sk_i^s contains the session key after the execution is accepted by instance Π_i^s. Before acceptance, it stores a distinguished NULL value.

Initialization. In a one-time initialization phase, before the first execution of the key establishment protocol, for each user $U_i \in \mathcal{P}$ a secret key/public key pair (SK_i, PK_i) is generated. The secret key SK_i is only revealed to U_i, the corresponding public key PK_i is given to all users.[2]

Communication network. We assume arbitrary point-to-point connections to be available between the users. However, the connections are insecure and fully asynchronous, modeled by an active adversary with full control over the network (cf. the adversarial model below).

[2] We assume these keys to be generated and distributed honestly by a trusted party.

Adversarial model. The adversary $\mathcal{A}$ interacts with the user instances via a set of oracles Execute, Send, Reveal, Corrupt and Test. We call the adversary *passive* if no access to the Send- and Corrupt-oracle is granted.

$\mathsf{Execute}(\{U_1, U_2, \ldots, U_r\})$ This query executes a protocol run between unused instances Π_i^s of the specified users and returns a transcript of all messages sent during the protocol execution.

$\mathsf{Send}(U_i, s, M)$ This query sends the message M to instance Π_i^s and returns the reply generated by this instance. A special message $M = \{U_1, \ldots, U_r\}$ sent to an unused instance will set $\mathsf{pid}_i^s := M$, $\mathsf{used}_i^s := \mathsf{true}$ and provoke Π_i^s to begin with the protocol execution.

$\mathsf{Reveal}(U_i, s)$ returns the session key sk_i^s.

$\mathsf{Corrupt}(U_i)$ returns the long-term secret key SK_i that U_i holds. We will refer to a user U_i as *honest* if no query of the form $\mathsf{Corrupt}(U_i)$ was made.

$\mathsf{Test}(U_i, s)$ The adversary is allowed to use this query only once. Provided that $\mathsf{sk}_i^s \neq \text{NULL}$, a random bit b is drawn and depending on b with probability $1/2$ the session key sk_i^s and with probability $1/2$ a uniformly chosen random session key is returned. The adversary is allowed to query other oracles after its Test-query, but no query that would repeal the freshness of Π_i^s is allowed.

Correctness. To exclude "useless" protocols, we take a group key establishment protocol P for *correct* if in the presence of a passive adversary a single execution of P among arbitrary participants $U_1, \ldots, U_r$ involves r instances $\Pi_1^{s_1}, \ldots, \Pi_r^{s_r}$ and ensures that with overwhelming probability all instances accept a matching session key with a common partner identifier and a common and unique session identifier. More formally, with overwhelming probability the following conditions have to hold:

- $\mathsf{used}_1^{s_1} = \cdots = \mathsf{used}_r^{s_r} = \mathsf{true}$;
- $\mathsf{acc}_1^{s_1} = \cdots = \mathsf{acc}_r^{s_r} = \mathsf{true}$;
- $\mathsf{sk}_1^{s_1} = \cdots = \mathsf{sk}_r^{s_r}$;
- $\mathsf{sid}_1^{s_1} = \cdots = \mathsf{sid}_r^{s_r}$ globally unique;
- $\mathsf{pid}_1^{s_1} = \mathsf{pid}_2^{s_2} = \cdots = \mathsf{pid}_r^{s_r} = \{U_1, \ldots, U_r\}$.

Freshness. For the security definition, we have to specify which instances are fresh, i. e., hold a session key that should be unknown to the adversary. As a first step we define the notion of partnering.

Definition 1 (Partnering). *Two instances $\Pi_i^{s_i}$, $\Pi_j^{s_j}$ are* partnered *if $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.*

Now the freshness of an instance is defined as follows.

Definition 2. *We call a user instance $\Pi_i^{s_i}$ that has accepted, i. e., $\mathsf{acc}_i^{s_i} = \mathsf{true}$,* fresh *if none of the following two conditions holds:*

- *For a $U_j \in \mathsf{pid}_i^{s_i}$ a $\mathsf{Corrupt}(U_j)$ query was executed before a query of the form $\mathsf{Send}(U_\ell, s_\ell, M)$ with $U_\ell \in \mathsf{pid}_i^s$ has taken place.*

– *A* $\mathsf{Reveal}(U_j, s_j)$ *was executed where* $\Pi_i^{s_i}$ *and* $\Pi_j^{s_j}$ *are partnered.*

We say that an adversary $\mathcal{A}$ was successful if $\mathcal{A}$, after interacting with the oracles including one $\mathsf{Test}(\Pi_i^{s_i})$ query for a fresh oracle $\Pi_i^{s_i}$, outputs a bit d and it holds that $d = b$ for the bit b used by the Test-oracle. We denote this probability by Succ and define $\mathcal{A}$'s advantage to be

$$\mathsf{Adv}_{\mathcal{A}} := |2 \cdot \mathsf{Succ} - 1|.$$

Definition 3 (Key secrecy/(basic) security). *We call the group key establishment protocol* P secure *if for all ppt adversaries* $\mathcal{A}$ *the function* $\mathsf{Adv}_{\mathcal{A}} = \mathsf{Adv}_{\mathcal{A}}(k)$ *is negligible in the security parameter* k.

Forward secrecy is addressed in the usual manner:

Definition 4 (Forward secrecy). *We say the group key establishment protocol* P *fulfills* forward secrecy*, if the disclosure of the private long-term keys used in the protocol execution does not compromise earlier derived session keys.*

The following extended security properties aim at avoiding further attacks imposed by malicious participants:

Definition 5 (Strong entity authentication). Strong entity authentication *to an oracle* $\Pi_i^{s_i}$ *is provided if both* $\mathsf{acc}_i^{s_i} = \mathsf{true}$ *and for all honest* $U_j \in \mathsf{pid}_i^{s_i}$ *with overwhelming probability there exists an oracle* $\Pi_j^{s_j}$ *with* $\mathsf{sid}_j^{s_j} = \mathsf{sid}_i^{s_i}$ *and* $U_i \in \mathsf{pid}_j^{s_j}$.

Definition 6 (Integrity). *We say a correct group key establishment protocol fulfills* integrity *if with overwhelming probability all oracles of honest principals that have accepted with the same session identifier* $\mathsf{sid}_j^{s_j}$ *hold identical session keys* $\mathsf{sk}_j^{s_j}$*, and associate this key with the same principals* $\mathsf{pid}_j^{s_j}$.

2.2 Assumptions on the underlying group

For the security proof of our protocol, the underlying group G (resp. family of groups $G = G(k)$, indexed by the security parameter) has to satisfy certain requirements. In particular, we assume products and inverses of group elements to be computable by ppt algorithms. For the sake of simplicity, we also assume that G allows a ppt computable canonical representation of elements, so that we can identify group elements with their canonical representation. To generate the group elements needed in a protocol execution, we rely on the existence of three algorithms, that capture the problem of creating "good instances":

– DomPar denotes a (stateless) ppt *domain parameter generation* algorithm that upon input of the security parameter 1^k outputs a finite sequence S of elements in G. The subgroup $\langle S \rangle$ of G spanned by S will be publicly known. For the special case of applying our framework to a CDH-assumption, S specifies a public generator of a cyclic group.

– SamAut denotes a (stateless) ppt *automorphism group sampling* algorithm that upon input of the security parameter 1^k and a sequence S output by DomPar returns a description of an automorphism ϕ on the subgroup $\langle S\rangle$, so that both ϕ and ϕ^{-1} can be evaluated efficiently. E. g., for a cyclic group, ϕ could be given as an exponent, or for an inner automorphism the conjugating group element could be specified.
– SamSub denotes a (stateless) ppt *subgroup sampling* algorithm that upon input of the security parameter 1^k and a sequence S output by DomPar returns a word $x(S)$ in the generators S (and their inverses) representing an element $x \in \langle S\rangle$. Intuitively, SamSub chooses a random $x \in \langle S\rangle$, so that it is hard to recognize x if we know elements of x's orbit under $\mathrm{Aut}(G)$. Our protocol needs an explicit representation of x in terms of the generators S.

With this notation, we can define a computational problem of parallel automorphism application, where $o \leftarrow \mathsf{A}(i)$ denotes that algorithm A outputs o upon receiving input i:

Definition 7 (Parallel automorphism application). *Let $r \in \mathbb{N}_{>0}$ be a natural number. By the* problem of r-fold parallel automorphism application (r-PAA) *w. r. t. the quadruple* $(G, \mathsf{DomPar}, \mathsf{SamAut}, \mathsf{SamSub})$ *we mean the task of finding an algorithm $\mathcal{A}$ which on input of S, $\phi_i(S) := (\phi_i(s))_{s\in S}$ for $i = 1, \dots, r$ and $\phi_1(x), \dots, \phi_r(x)$ outputs the group element x represented by the word $x(S)$, where*

– $S \leftarrow \mathsf{DomGen}(1^k)$,
– $x(S) \leftarrow \mathsf{SamSub}(1^k, S)$,
– $(\phi_i, \phi_i^{-1}) \leftarrow \mathsf{SamAut}(1^k, S)$ $(i = 1, \dots, r)$.

To capture the assumption needed in the security proof below, we also define the advantage of an adversary in solving the above problem:

Definition 8 (r-PAA advantage). *For an algorithm $\mathcal{A}$ trying to solve r-PAA, we denote its advantage as a function in the security parameter k and its runtime t by $\mathsf{Adv}_{\mathcal{A}}^{r\text{−PAA}} = \mathsf{Adv}_{\mathcal{A}}^{r\text{−PAA}}(k, t) =$*

$$\Pr\left(x \leftarrow \mathcal{A}(S, (\phi_i(S), \phi_i(x))_{1\le i\le r}) \;\middle|\; \begin{array}{l} S \leftarrow \mathsf{DomGen}(1^k), x(S) \leftarrow \mathsf{SamSub}(1^k, S), \\ (\phi_i, \phi_i^{-1}) \leftarrow \mathsf{SamAut}(1^k, S)\ (i = 1, \dots, r) \end{array} \right).$$

Our security proof builds on the assumption that for any ppt adversary $\mathcal{A}$ the advantage $\mathsf{Adv}_{\mathcal{A}}^{r\text{−PAA}}$ is negligible. For the case of ϕ being an inner automorphism, r-PAA expresses a kind of parallel conjugacy problem. Note however, that instead of looking for concrete instances building on a non-abelian group, we may apply our framework to an "ordinary" Computational Diffie-Hellman (CDH) setting, too:

Example 1 (Basing on CDH). Let G be a cyclic group and choose for $S := (g)$ an element $g \in G$ of prime order q. Now let SamSub choose uniformly at random an exponent $x \in \{1, \dots, q\}$. Similarly, we specify SamAut to choose uniformly at random an exponent $\phi \in \{1, \dots, q-1\}$. Then r-PAA is polynomial time equivalent to the CDH-problem in $\langle g\rangle$:

"CDH solution ⇒ r-PAA solution": A CDH-oracle allows to find g^x from a single pair $(g^\phi, g^{x\phi})$ as follows. First compute $g^{\phi^{-1} \bmod q}$ by using the CDH-oracle to multiply the exponents of (g, g) with $g = \left(g^\phi\right)^{\phi^{-1}}$ taken for a power of the group generator g^ϕ. Next we can obtain g^x by applying the CDH-oracle to $g^{\phi^{-1} \bmod q}$ and $g^{x\phi}$.

"CDH solution ⇐ r-PAA solution": Given g^{ϕ_1}, g^v $(\phi_1, v \in \{1, \ldots, q-1\})$, we can use an oracle solving r-PAA to compute $g^{v\phi_1^{-1}}$: We can interpret v as having the form $v = x \cdot \phi_1 \bmod q$, and by raising g^{ϕ_1} and g^v to uniformly at random chosen powers $\phi_1^{-1}\phi_i \in \{1, \ldots, q-1\}$ $(i = 2, \ldots, r)$, we obtain the input needed by an oracle solving r-PAA to compute $g^x = g^{v\phi_1^{-1}}$. Hence, given a pair (g^u, g^v) we can compute g^{uv} as follows:

1. Apply the above method to (g^u, g^v), yielding $g^{vu^{-1}}$.
2. Apply the above method to $(g^v, g^{vu^{-1}})$, yielding $g^{u^{-1}}$.
3. Apply the above method to $(g^{u^{-1}}, g^v)$, yielding g^{uv}.

2.3 Groundwork of the protocol

The r-PAA assumption is in quintessence a variant of a *key encapsulation mechanism* (KEM). A KEM provides the public key algorithm that is needed for constructing a hybrid encryption system. In contrast to public key encryption it is not necessary to be able to encrypt arbitrary messages, but only random messages, which don't need to be given as input to the algorithm.

Smart [29] extends KEM to mKEM which captures key encapsulation to multiple parties. An mKEM consists of Algorithms Gen, Enc and Dec. At this Gen will take the domain parameters as input and output a public/private key pair (pk, sk). The algorithm Enc takes as input a list of public keys $(pk_1, \ldots, pk_n)$ and outputs a pair consisting of a key K and a ciphertext C. Finally, Dec takes as input a ciphertext C and a private key sk_i and outputs the key K.

The assumption about the group in Section 2.2 resembles an mKEM. However, for a KEM, the key space will consist of a finite set, such that K is indistinguishable from an element chosen uniformly at random. The value x, sampled by the algorithm SamSub will generally not offer this property, so a randomness extraction has to be applied on x to build a KEM. We now give the interpretation of the PAA as an mKEM, using a random oracle H as a randomness extractor. Note that the protocol in Section 3 will not need the randomness extraction for an individual x, but only for the collection of the x-values of all participants.

After having generated domain parameters with DomPar, SamAut produces the automorphism ϕ on the subgroup $\langle S\rangle$. The images $(\phi_i(t))_{t\in S}$ of the generators S will act as public key and ϕ_i^{-1} as private key. This will provide the algorithm Gen. Given the subgroup generators S, SamSub returns a word $x(S)$ in the generators S. Then, given any number of public keys $\phi_i(S)$ for the subset S, the ciphertext $\phi_i(x(S))$ can be computed. Thus, the combination of SamSub and application of ϕ_i can be seen as providing Enc. Again, Dec is only given implicitly, as the application of ϕ^{-1} to $\phi_i(x(S))$ is straightforward.

$$
\begin{array}{ll}
\text{Domain parameter } \mathbf{D}: & S \leftarrow \mathsf{DomGen}(1^k) \\
(pk, sk) \leftarrow \mathsf{Gen}(\mathbf{D}): & (\phi, \phi^{-1}) \leftarrow \mathsf{SamAut}(1^k, S) \\
& pk = (\phi(t))_{t \in S} \\
& sk = \phi^{-1} \\
(K, C) \leftarrow \mathsf{Enc}(pk_1, \dots, pk_n): & x(S) \leftarrow \mathsf{SamSub}(1^k, S) \\
& K = H(x(S)) \\
& C = (\phi_1(x(S)), \dots, \phi_n(x(S))) \\
K \leftarrow \mathsf{Dec}(C, sk_i, (pk_1, \dots, pk_n)): & K = H(\phi_i^{-1}(\phi_i(x(S)))
\end{array}
$$

Security of r-PAA as an mKEM. With the interpretation as above, intuitively r-PAA is secure as an mKEM. Using the random oracle to derive the key K, transforms the indistinguishability of keys in the mKEM into the problem to compute the preimage, as the r-PAA advantage. However, Smart [29] defines an r out of n security where the adversary is offered n public keys and can chose a set of r on which he will mount his attack. In this respect, the above r-PAA problem yields an r out of r secure mKEM. Though, the weaker requirements for a secure r-PAA might help the construction of concrete instances.

On Burmester-Desmedt style key agreements. The Burmester-Desmedt principle constructs a group key by arranging the participants in a circle, establishing keys between neighbors and broadcasting information, that allows anyone who knows one key in the circle, to compute all other keys. Having in mind the construction of a 2-round protocol, the key establishment should be possible in one round. However, forward secrecy requires ephemeral public keys, such that in order to establish a key, first U_i has to execute Gen and send the result to U_j who has to execute Enc and return the ciphertext to U_i. As this requires already 2 rounds, we have chosen a different approach, which is similar to [9] but guarantees forward secrecy in addition.

Protocol idea. The idea for the protocol is now as follows: In the first round, all participants will generate an ephemeral key, what will be necessary to achieve forward secrecy. In the next round, each participant U_i will use the encryption algorithm of the KEM to obtain a key contribution K_i and a ciphertext C, and broadcast the ciphertext C. Finally, the participants compute a group key from the contributions U_j, $j = 1, \dots, n$.

3 A 2-Round Protocol for Group Key Agreement

To discuss our group key agreement protocol we adopt the common assumption that, from some protocol-external context, the set of protocol participants $\mathcal{U} \subseteq \mathcal{P}$ is known to all $U_i \in \mathcal{U}$. To simplify notation, w. l. o. g. we assume $\mathcal{U} = \{U_1, \dots, U_r\}$. Moreover, we assume that an asymmetric signature scheme is available that is existentially unforgeable under adaptive chosen message attacks. The respective signing and verification keys are to be fixed and distributed throughout the initialization phase mentioned in Section 2.1, and we denote a signature of a protocol participant U_i on a message M by $\mathsf{Sig}_i(M)$.

3.1 Description of the protocol

Having fixed the security parameter k, first we have to run $\mathsf{DomGen}(1^\mathsf{k})$ to generate the public subgroup generators S. Hereafter, for an instance $\Pi_i^{s_i}$ of a protocol participant U_i a single protocol run can be described as shown in Figure 1. At this, *Broadcast:* M means that message M is sent to all other participants $U_j \in \mathcal{U}$ over *point-to-point* connections, i. e., the adversary is allowed to delay, suppress or modify some or all of the transmitted messages. In contrast to the idea in the last section, it is possible to separate portions for each participant $\phi_i(x(S))$ and instead of broadcasting all ciphertexts, every participant gets only the necessary part. Moreover, the randomness extraction is only applied on the list $x_1, \ldots, x_n$, instead on every x_i. Finally, H denotes a cryptographic hash function which will be modeled as a random oracle.

Round 1: Initialization Set $\mathsf{pid}_i^{s_i} := \mathcal{U}$, $\mathsf{used}_i^{s_i} := \mathsf{true}$.
Choose $(\phi_i^{s_i}, (\phi_i^{s_i})^{-1}) \leftarrow \mathsf{SamAut}(1^k)$, $x_i^{s_i}(S) \leftarrow \mathsf{SamSub}(1^k, S)$, and compute the message $m_1^{s_i}(U_i) := (U_i, (\phi_i^{s_i}(t))_{t\in S}, H(x_i^{s_i}))$.
Broadcast: $m_1^{s_i}(U_i)$.
Round 2: Key Exchange Set $\mathsf{sid}_i^{s_i} := H\left(m_1^{s_1}(U_1), \ldots, m_1^{s_r}(U_r), \mathsf{pid}_i^{s_i}\right)$.
Compute and send $m_2^{s_i}(U_i, U_j) := \left(U_i, \phi_j^{s_j}(x_i^{s_i}), \mathsf{Sig}_i(\mathsf{sid}_i^{s_i})\right)$ to each participant $U_j \in \mathsf{pid}_i^{s_i}$, $j \neq i$. (To compute $\phi_j^{s_j}(x_i^{s_i})$ use the representation of $x_i^{s_i} = x_i^{s_i}(S)$ in terms of the generators S.)
Key Generation Compute from $\phi_i^{s_i}(x_j^{s_j})$ the original $x_j^{s_j}$ for all $j \neq i$ by applying the inverse of $\phi_i^{s_i}$.
Compute the common session key $K := H\left(x_1^{s_1}, \ldots, x_r^{s_r}, \mathsf{pid}_i^{s_i}\right)$.
Verification Check for all $U_j \in \mathsf{pid}_i^{s_i}$ if $\mathsf{Sig}_j(\mathsf{sid}_j^{s_j})$ is a valid signature for $\mathsf{sid}_i^{s_i}$ and if for $x_j^{s_j}$ the received hash value $H(x_j^{s_j})$ in $m_1^{s_j}(U_j)$ was correct.
If true, set $\mathsf{acc}_i^{s_i} := \mathsf{term}_i^{s_i} := \mathsf{true}$, and $\mathsf{sk}_i^{s_i} := K$.
Else set $\mathsf{acc}_i^{s_i} := \mathsf{false}$, $\mathsf{term}_i^{s_i} := \mathsf{true}$.

Fig. 1. A 2-round group key agreement protocol basing on r-PAA

At first glance, Round 1 of the protocol may look suspicious: The message is not signed and hence an attacker may tamper with this message. The underlying idea here is, that any tampering with the message in Round 1 will result in a failed signature verification in Round 2, because the session identifier $\mathsf{sid}_i^{s_i}$ computed and signed by $\Pi_i^{s_i}$ involves the correct message from Round 1. Further on, one may wonder whether one shouldn't simply fix the $\phi_i^{s_i}(S)$-values and include them in the public key of user U_i. Effectively, the latter would render $\phi_i^{s_i}$ a long-term secret and the protocol would no longer achieve forward secrecy.

Remark 1. Having in mind instances of the protocol in Figure 1 where the ϕ_i are inner automorphisms, it is worth noting that the protocol is symmetric in the sense that all participants perform the same steps: Differing from Anshel et al.'s 2-party construction, the key computation for the initiator is the same as for the other protocol participants.

3.2 Security analysis

Correctness of the protocol in Figure 1 is immediate. To prove its security, we first observe that the constructed session identifier is with overwhelming probability globally unique:

Lemma 1. *If for all ppt adversaries $\mathcal{A}$ the advantage $\mathsf{Adv}_{\mathcal{A}}$ in solving r-PAA is negligible, then the session identifier $\mathsf{sid}_i^{s_i}$ constructed in the above protocol is with overwhelming probability globally unique.*

Proof. The assumption of the lemma implies in particular that the probability of SamSub outputting twice the same value in a ppt number of executions is negligible. Thus the collision-freeness of H yields the desired uniqueness of the session identifier. □

Next, before looking at (basic) security, we note that the above protocol also offers strong entity authentication and integrity:

Proposition 1. *The protocol provides strong entity authentication according to Definition 5 and integrity according to Definition 6.*

Proof. Strong entity authentication. Consider an arbitrary instance $\Pi_i^{s_i}$ of an uncorrupted participant U_i that has accepted with session identifier $\mathsf{sid}_i^{s_i}$. Let $U_j \in \mathsf{pid}_i^{s_i}$ be some other uncorrupted participant. Instance $\Pi_i^{s_i}$ must have received a message of U_j with a signature on U_j's session identifier $\mathsf{sid}_j^{s_j}$. By unforgeability of the signature scheme, uniqueness of the session identifier $\mathsf{sid}_j^{s_j} = \mathsf{sid}_i^{s_i}$, and the collision resistance of the hash function we obtain $\mathsf{pid}_j^{s_j} = \mathsf{pid}_i^{s_i}$ with overwhelming probability.

Integrity. Consider any two instances $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ that both have accepted with $\mathsf{sid} = \mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ and where the participants U_i and U_j are honest. By unforgeability of the signature scheme, uniqueness of the session identifier sid, and the collision resistance of the hash function, with overwhelming probability we get $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ and the equivalence of the messages $m_1^{s_\ell}(U_\ell)$ they received in Round 1. Those messages include hash values $H(x_\ell^{s_\ell})$ from all protocol participants and before accepting, all participants check if the computed values $x_\ell^{s_\ell}$ in Round 2 are consistent with the $H(x_\ell^{s_\ell})$. Unless a collision of H occurs they compute the same key. □

For the ease of presentation, in the proof of the basic security property we imagine the protocol without the hash value $H(x_i^{s_i})$ in Round 1. This simplification can be justified with a standard random oracle argument as in the proof of Lemma 3.

Proposition 2. *Denote the maximal number of protocol participants by* $n = |\mathcal{P}|$*, and let* $\mathcal{A}$ *be an adversary that is allowed at most* $q_\mathrm{s}, q_\mathrm{ex}, q_H$ *queries to the* Send*,* $\mathsf{Execute}$ *and random oracle* H*, respectively. Moreover, let* $\mathsf{Adv}^{(n-1)-\mathrm{PAA}}$ *resp.* $\mathsf{Adv}^{\mathrm{Sig}}$ *be the maximum advantage of a ppt algorithm solving* $(n-1)$*-PAA resp. achieving an existential forgery in running time* t*. Then*

$$\mathsf{Adv}_\mathcal{A} = |\mathsf{Succ} - 1/2| \leq n \cdot (q_\mathrm{s} + q_\mathrm{ex})^n \cdot q_H \cdot \mathsf{Adv}^{(n-1)-\mathrm{PAA}} + n \cdot \mathsf{Adv}^{\mathrm{Sig}} + \mathrm{negl}(k)$$

where $\mathrm{negl}(k)$ *is negligible in* k*.*

Proof. Let $\mathsf{Succ} := (\mathsf{Adv}_\mathcal{A} + 1)/2$ be the success probability of adversary $\mathcal{A}$ to win the experiment. Imagine $\mathcal{A}$ now to be connected to a simulator *Sim* that simulates the oracles. We consider a sequence of games and bound the difference of the adversary's success probability between subsequent games.

In Game 0 the simulator *Sim* simulates the oracles and principals' instances faithfully. Thus, there is no difference for the adversary and denoting $\mathcal{A}$'s success probability in Game i by $\mathsf{Succ}_{\mathrm{Game}\ i}$, we have $\mathsf{Succ}_{\mathrm{Game}\ 0} = \mathsf{Succ}$.

In Game 1 the simulator will keep a list with an entry $(i, \mathsf{sid}_i^{s_i})$ for every session identifier $\mathsf{sid}_i^{s_i}$ the simulator signs with the secret key of user U_i and returns it in a Round 2 message to $\mathcal{A}$ following an $\mathsf{Execute}$-query or on a Send-query. We define the event Forge to occur, if $\mathcal{A}$ comes up with a query $\mathsf{Send}(*, *, M)$ where M includes a signature $\mathsf{Sig}(\mathsf{sid}_i^{s_i})$, signed by an uncorrupted principal U_i and $(i, \mathsf{sid}_i^{s_i})$, does not appear in the simulator's list. In this case we abort the experiment and count it as success for the adversary. Thus we have:

$$|\mathsf{Succ}_{\mathrm{Game}\ 1} - \mathsf{Succ}_{\mathrm{Game}\ 0}| \leq \Pr(\mathsf{Forge}).$$

Lemma 2. *If the signature scheme is existentially unforgeable, the probability of* Forge *is negligible. Formally:*

$$\Pr(\mathsf{Forge}) \leq n \cdot \mathsf{Adv}^{\mathrm{Sig}}$$

Proof. The simulator can use an adversary that can reach Forge with a non-negligible probability as black box to forge a signature from the underlying signature scheme.

The simulator is given a public key PK and a signing oracle. In the initialization it will uniformly choose one user U_i and assign the key PK as PK_i to U_i. If in the following simulation *Sim* has to generate a signed message for U_i it will use the signing oracle to sign the message. If $\mathcal{A}$ will send a message $(*, \sigma)$, where σ is a signature of a session identifier $\mathsf{sid}_i^{s_i}$ that is not in the simulator's list, the simulator will return $(\mathsf{sid}_i^{s_i}, \sigma)$ as existential forgery. Otherwise the simulator returns $\perp$. As i was chosen uniformly the simulator will succeed with a probability of $1/n \cdot \Pr(\mathsf{Forge})$, thus $\Pr(\mathsf{Forge}) \leq n \cdot \mathsf{Adv}^{\mathrm{Sig}}$. $\square$

In Game 2 the simulator will keep a list with entries

$$(m_1^{s_1}(U_1), \ldots, m_1^{s_n}(U_n), H(m_1^{s_1}(U_1), \ldots, m_1^{s_n}(U_n)))$$

for every computation of a session identifier invoked by an $\mathsf{Execute}$-query or Send-query and all entries $(M, H(M))$ where $\mathcal{A}$ queried the random oracle directly. We define the event $\mathsf{Collision}$ to occur, if the simulator computes a session identifier $H(m_1^{s_1}(U_1), \ldots, m_1^{s_n}(U_n))$ which equals a session identifier that $\mathcal{A}$ obtained previously with *non-identical* messages. In this case we abort the experiment and count it as success for the adversary. From H being a random oracle, we conclude that $|\mathsf{Succ}_{\text{Game 2}} - \mathsf{Succ}_{\text{Game 1}}|$ is negligible.

In Game 3 the simulation of the Test oracle is modified. On a query $\mathsf{Test}(U_i, s_i)$, the simulator checks if $\Pi_i^{s_i}$ is fresh. If so, then *Sim* will not query the random oracle, but return a random value in any case. As now no information about the Test-oracle's secret bit b is given to $\mathcal{A}$ in Game 3, the success probability is $\mathsf{Succ}_{\text{Game 3}} = 1/2$.

Now we have to determine the difference in the adversary's success probability between Game 2 and Game 3. For $\mathcal{A}$, a random value and the random oracle's answer are indistinguishable as long as $\mathcal{A}$ does not know the actual query to the random oracle. The success probabilities can only differ, if $\mathcal{A}$ queries $H(x_1^{s_1}, \ldots, x_r^{s_r}, \mathsf{pid}_i^{s_i})$ to the random oracle. Denoting this event by Random, we have

$$|\mathsf{Succ}_{\text{Game 3}} - \mathsf{Succ}_{\text{Game 2}}| \leq \Pr(\mathsf{Random}).$$

Lemma 3. *The probability* $\Pr(\mathsf{Random})$ *of the event* Random *to occur is negligible if* n *is constant and* $\mathsf{Adv}^{(n-1)-\mathrm{PAA}}$ *is negligible.*

Proof. The simulator is given an instance $(S, (\phi_i(S), \phi_i(x))_{1 \leq i \leq n-1)})$ of the $(n-1)$-PAA problem. In the initialization phase, the simulator will give S as parameter to $\mathcal{A}$ and uniformly choose n random numbers $\alpha_i \in \{1, q_s + q_{ex}\}$ $(i = 1, \ldots, n)$ to point to the instances $\Pi_i^{\alpha_i}$. The simulator will choose uniformly at random $\beta \in \{1, \ldots, n\}$ to select one distinguished instance $\Pi_\beta^{\alpha_\beta}$ among them.

When the simulator has to process Round 1 for one instance $\Pi_i^{\alpha_i}, i = 1, \ldots, n$, $i \neq \beta$, *Sim* will use the given $\phi_i(S)$ instead of computing a new ϕ_i with SamAut. For instance $\Pi_\beta^{\alpha_\beta}$ the simulator will use the given $x(S)$. If instance $\Pi_\beta^{\alpha_\beta}$ does not only get messages containing $\phi_i(S), (i = 1, \ldots, n, i \neq \beta)$ the simulator aborts and outputs $\perp$. Also, if the simulator ever has to apply a ϕ_i^{-1} it aborts and outputs $\perp$ (this will only happen from a Reveal-query).

Because $\Pi_\beta^{\alpha_\beta}$ is uniformly selected out of a set of $n \cdot (q_s + q_{ex})$ potential instances, it will be used in the Test-query with a probability of $(n \cdot (q_s + q_{ex}))^{-1}$. To be able to apply the Test-query the adversary has to let $\Pi_\beta^{\alpha_\beta}$ accept. All $U_i \in \mathsf{pid}_\beta^{\alpha_\beta}$ have to be uncorrupted. Then by uniqueness of the session identifier (Lemma 1) the messages $\Pi_\beta^{\alpha_\beta}$ must have got in Round 1 were generated by the same instances as the messages $\Pi_\beta^{\alpha_\beta}$ received in Round 2. These have to be the distinguished oracles $\Pi_i^{\alpha_i}$ for $U_i \in \mathsf{pid}_\beta^{\alpha_\beta}$. For the principals $U_j \notin \mathsf{pid}_\beta^{\alpha_\beta}$, $\Pi_j^{\alpha_j}$

must be an oracle that was not revealed. There must be at least one potential instance that is not used for each $U_j \notin \mathsf{pid}_\beta^{\alpha_\beta}$. Consequently, with a probability of $1/n \cdot (q_\mathrm{s} + q_\mathrm{ex})^n$ the principals are distributed as needed.

If $\mathcal{A}$ halts, the simulator chooses uniformly one of the at most q_H queries to the random oracle, extracts x_β (assuming it is of the form $H(x_1^{s_1}, \ldots, x_r^{s_r}, \mathsf{pid}_i^{s_i})$) and answers this to the $(n-1)$-PAA challenge. The probability to pick the correct query is $1/q_H \cdot \Pr(\mathsf{Random})$. With a probability of at least

$$\Pr((n-1)\mathrm{-}\mathsf{PAAsolved}) \geq \frac{1}{n \cdot (q_\mathrm{s} + q_\mathrm{ex})^n \cdot q_H} \cdot \Pr(\mathsf{Random})$$

the simulator solves the challenge. □

Putting it all together we see that $\mathsf{Adv}_\mathcal{A} = |\mathsf{Succ} - 1/2|$ is smaller or equal than

$$n \cdot (q_\mathrm{s} + q_\mathrm{ex})^n \cdot q_H \cdot \mathsf{Adv}^{(n-1)-\mathrm{PAA}} + n \cdot \mathsf{Adv}^{\mathrm{Sig}} + \mathrm{negl}(k) \quad .$$

□

Remark 2. If instead of a constant number n of potential protocol participants, we want to allow a size $\mathcal{P}$ of polynomial size, the bound in Proposition 2 is in general no longer negligible. However, if we base on a CDH-assumption as in Example 1, we can allow for a set $\mathcal{P}$ of polynomial size: With the argument given in Example 1 we see that in this case solving 1-PAA is equivalent to solving r-PAA for an arbitrary r of polynomial size. In the security proof, this reduction allows us to replace the exponent n by the constant 2.

Also, an r-PAA that is secure even if the adversary can choose the r public keys out of a polynomial sized set will help. Because the definition of an mKEM takes such a choice into account, the protocol allows a polynomial sized set of users, if it bases on such an mKEM.

Finally, forward secrecy follows with the standard argument that the long-term keys are used for message authentication exclusively, and we obtain:

Proposition 3. *The protocol in Figure 1 fulfills forward secrecy in the sense of Definition 4.*

4 Conclusion

In this contribution we have described a 2-round group key agreement and showed it to be secure under the assumption that certain group-theoretical tools are available. In addition to the "standard" security requirement, the proposed protocol also offers strong entity authentication and integrity. While our framework is primarily geared towards building a provably secure group key agreement on non-abelian groups, it also allows to derive a 2-round group key agreement from a CDH assumption.

Acknowledgment

We are indebted to Dennis Hofheinz and Jonathan Katz for valuable discussions and comments.

References

1. Iris Anshel, Michael Anshel, Benji Fisher, and Dorian Goldfeld. New Key Agreement Protocols in Braid Group Cryptography. In David Naccache, editor, *Topics in Cryptology, Proceedings of CT-RSA 2001*, number 2020 in Lecture Notes in Computer Science, pages 13–27. Springer-Verlag, 2001.
2. Iris Anshel, Michael Anshel, and Dorian Goldfeld. An Algebraic Method for Public-Key Cryptography. *Mathematical Research Letters*, 6:287–291, 1999.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing STOC*, pages 319–428, 1998.
4. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000.
5. Mihir Bellare and Phillip Rogaway. Entitiy Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, 1994.
6. Jens-Matthias Bohli. A Framework for Robust Group Key Agreement. In Marina L. Gavrilova, Osvaldo Gervasi, Vipin Kumar, Chih Jeng Kenneth Tan, Antonio Laganà David Taniar, Youngsong Mun, and Hyunseung Choo, editors, *Computational Science and Its Applications – ICCSA 2006*, volume 3982 of *Lecture Notes in Computer Science*, pages 355–364. Springer, 2006.
7. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. Cryptology ePrint Archive, Report 2005/395, 2005. `http://eprint.iacr.org/2005/395/`.
8. Dan Boneh and Alice Silverberg. Applications of Multilinear Forms to Cryptography. *Contemporary Mathematics*, 324:71–90, 2003.
9. Colin Boyd and Juan Manuel González Nieto. Round-Optimal Contributory Conference Key Agreement. In Yvo Desmedt, editor, *Public Key Cryptography, Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2002.
10. Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography; Texts and Monographs. Springer, 2003.
11. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In Pierangela Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 255–264. ACM Press, 2001.
12. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

13. Dario Catalano, David Pointcheval, and Thomas Pornin. IPAKE: Isomorphisms for Password-based Authenticated Key Exchange. In Matthew K. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 477–493. Springer, 2004.
14. Jung Hee Cheon and Byungheup Jun. A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 212–225. Springer, 2003.
15. Patrick Dehornoy. Braid-based cryptography. In Alexei G. Myasnikov, editor, *Group Theory, Statistics, and Cryptography*, number 360 in Contemporary Mathematics, pages 5–33. ACM Press, 2004. Online available at `http://www.math.unicaen.fr/~dehornoy/Surveys/Dgw.ps`.
16. David Gerber, Shmuel Kaplan, Mina Teicher, Boaz Tsaban, and Uzi Vishne. Probabilistic solutions of equations in the braid group. *Advances in Applied Mathematics*, 35(3):323–334, 2005.
17. María Isabel González Vasco, Consuelo Martínez, Rainer Steinwandt, and Jorge L. Villar. A new Cramer-Shoup like methodology for group based provably secure schemes. In Joe Kilian, editor, *Proceedings of the 2nd Theory of Cryptography Conference TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 495–509. Springer, 2005.
18. Dimitri Grigoriev and Ilia Ponomarenko. Constructions in public-key cryptography over matrix groups. arXiv preprint, 2005. Online available at `http://arxiv.org/abs/math.GR/0506180`.
19. Dennis Hofheinz and Rainer Steinwandt. A Practical Attack on Some Braid Group Based Cryptographic Primitives. In Yvo Desmedt, editor, *Public Key Cryptography, Proceedings of PKC 2003*, number 2567 in Lecture Notes in Computer Science, pages 187–198. Springer-Verlag, 2002.
20. Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *12th ACM Conference on Computer and Communications Security*, pages 180–189. ACM Press, 2005.
21. Jonathan Katz and Moti Yung. Scalable Protocols for Authenticated Group Key Exchange. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.
22. Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju sung Kang, and Choonsik Park. New Public-Key Cryptosystem Using Braid Groups. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2000.
23. Ho-Kyu Lee, Hyang-Sook Lee, and Young-Ran Lee. Cryptology ePrint Archive: Report 2003/018, 2003. `http://eprint.iacr.org/2003/018`.
24. Victor Shoup. On Formal Models for Secure Key Exchange (version 4). Revision of IBM Research Report RZ 3120 (April 1999), November 1999. Online available at `http://www.shoup.net/papers/skey.pdf`.
25. Vladimir Shpilrain and Alexander Ushakov. A new key exchange protocol based on the decomposition problem. Cryptology ePrint Archive: Report 2005/447, 2005. `http://eprint.iacr.org/2005/447`.
26. Vladimir Shpilrain and Alexander Ushakov. Thompson's Group and Public Key Cryptography. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security: Third International Conference, ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 151–163, 2005.

27. Vladimir Shpilrain and Alexander Ushakov. The conjugacy search problem in public key cryptography: unnecessary and insufficient. *Applicable Algebra in Engineering, Communication and Computing*, to appear. Online available at `http://www.sci.ccny.cuny.edu/~shpil/csp.pdf`.
28. Vladimir Shpilrain and Gabriel Zapata. Combinatorial group theory and public key cryptography. *Applicable Algebra in Engineering, Communication and Computing*, to appear. Online available at `http://www.sci.ccny.cuny.edu/~shpil/pkc.pdf`.
29. Nigel Smart. Efficient Key Encapsulation to Multiple Parties. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks – SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 2005.