symmetry

MDPI

Article

Building Group Key Establishment on Group Theory: A Modular Approach

Jens-Matthias Bohli [1], María I. González Vasco [2,*] and Rainer Steinwandt [3]

[1] Department of Information Technology, Mannheim University of Applied Sciences, 68163 Mannheim, Germany; j.bohli@hs-mannheim.de
[2] MACIMTE, U. Rey Juan Carlos, 28933 Móstoles, Madrid, Spain
[3] Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL 33431, USA; rsteinwa@fau.edu
* Correspondence: mariaisabel.vasco@urjc.es

Received: 26 December 2019; Accepted: 19 January 2020; Published: 30 January 2020

Abstract: A group key establishment protocol is presented and proven secure in the common reference string mode. The protocol builds on a group-theoretic assumption, and a concrete example can be obtained with a decision Diffie–Hellman assumption. The protocol is derived from a two-party solution by means of a protocol compiler presented by Abdalla et al. at TCC 2007, evidencing the possibility of meaningfully integrating cryptographic and group-theoretic tools in cryptographic protocol design. This compiler uses a standard ring configuration, where all users behave symmetrically, exchanging keys with their left and right neighbor, which are later combined to yield a shared group key.

Keywords: group key establishment; group theory; provable security; protocol compiler

1. Introduction

Cryptography is the science of handling, storing, transmitting, and processing information securely, even in the presence of adversaries. For centuries, cryptographic techniques were developed for diplomatic or military scenarios, while nowadays individuals and institutions (often obliviously) make use of cryptographic tools every day. As a complex discipline, cryptography builds upon physics, mathematics, and different research areas within computer science. Mathematics are the main source of tools for cryptographic developments, in which, security is often demonstrated using the hardness of well understood mathematical problems. This paper is concerned with the construction of a widely used cryptographic tool, a group key exchange, using group theory as a base. Key exchange allows a number of users to establish a common secret value which will be subsequently used to secure their communication. Such cryptographic tools are often constructed from number theoretical problems (described in finite cyclic groups), and a challenging research question is whether secure constructions can be derived from different problems arising in group theory.

In recent years, not only due to the advent of quantum computation, significant efforts have been made to identify new mathematical platforms for implementing cryptographic schemes. One of the explored candidate platforms is the theory of finitely presented groups, where, in particular, a number of works on key establishment have been published. The first constructions in this direction where published about twenty years ago [1–3], and different approaches towards secure constructions have been explored regularly, such as [4–9], and the more recently [10]. Unfortunately, most of the proposed protocols have not been analyzed in a modern cryptographic security model (like [11–16]), and only few group-theoretic constructions with a rigorous security analysis seem to be known. This lack of formalism has resulted in weaknesses being overlooked (see, for instance, [17]).

One approach to facilitate the synergy between group-theoretic and cryptographic tools is the identification of general constructions that under suitable group-theoretic conditions yield an (efficient) cryptographic scheme with provable security guarantees. As examples for research along this line of thought, proposals for constructing IND-CCA secure asymmetric encryption schemes can be mentioned [18,19]. Also, constructions for building provably secure group key establishment schemes have been proposed (cf. [20,21]), but identifying practical non-abelian instances still appears to be a challenging problem. In this contribution, we build on [21], and try to extend and simplify their approach in the following sense:

- Instead of the random oracle model, we use the common reference string model. An (expected) price we pay for this, is the need of a decisional assumption instead of a computational one that is used in [21].
- Instead of setting out for a *group* key establishment directly, we suggest a construction for the two-party case and thereafter apply a protocol compiler of Abdalla et al. [22].

In terms of round complexity, we lose some efficiency through the modular design approach we chose. On the other hand, this modular design approach illustrates how an integration of group-theoretic and cryptographic tools can look like. Moreover, we obtain a comparatively clear group-theoretic condition which hopefully stimulates further research on finding concrete non-abelian instances. Concrete examples of our protocol can be derived from a decision Diffie–Hellman assumption, but we hope that in subsequent work also concrete non-abelian instances can be identified.

2. Preliminaries: Security Model and Protocol Goals

To explore the security of our protocol, we adopt the model used by Abdalla et al. [22], which can be traced back to [23–27]. Both to formulate our two-party solution and to use the "2-to-n compiler" from [22], we assume a common reference string (CRS) to be available that encodes the following information:

- Two values v_0, v_1. These will be the input for a pseudorandom function at the time of computing the session identifier and session key;
- The information necessary to implement a non-interactive and non-malleable commitment scheme (see Section 3.1 for further details);
- Two elements, chosen independently and uniformly at random, each taken from a family of universal hash functions (one as needed for the compiler in [22] and one for our two-party solution as detailed in Section 3.1).

This is similar to the constructions for password-authenticated key establishment in [24,27].

2.1. *Communication Model and Adversarial Capabilities*

As usual, we model protocol participants as probabilistic polynomial time (ppt) Turing machines (all our proofs hold for both uniform and non-uniform machines). We denote by $\mathcal{P}$ the total set of users which is assumed to be of polynomial size and by $\mathcal{U} = \{U_0, \ldots, U_{n-1}\} \subseteq \mathcal{P}$ the set of protocol participants. To enable authentication among the protocol participants, we assume that an existentially unforgeable signature scheme is available with all signing keys being chosen independently in a trusted initialization phase. The verification keys are assumed to be distributed in a trusted initialization phase, prior to the protocol execution.

2.1.1. Protocol Instances

We allow each protocol participant $U_i \in \mathcal{U}$ to execute polynomially many protocols *instances* in parallel. Each single instance $\Pi_i^{s_i}$ may be understood as a process executed by participant U_i. We will denote by $\Pi_i^{s_i}$ $(s_i \in \mathbb{N})$ the $s_i - th$ instance of user $U_i \in \mathcal{U}$, and the following seven variables are assigned to each instance:

$\mathsf{used}_i^{s_i}$	will indicate whether this instance is or has been used for a protocol run. The $\mathsf{used}_i^{s_i}$ flag is set through a protocol message received by the corresponding instance due to a call to the Send oracle (see below);
$\mathsf{state}_i^{s_i}$	stores the state information needed during the protocol execution;
$\mathsf{term}_i^{s_i}$	indicates if the execution has terminated;
$\mathsf{sid}_i^{s_i}$	denotes a session identifier (which may be public) which may be later use as identifier for the session key $\mathsf{sk}_i^{s_i}$ (in particular, the adversary is thus allowed to learn session identifiers);
$\mathsf{pid}_i^{s_i}$	stores the user identities that $\Pi_i^{s_i}$ aims at establishing a key with. This set includes U_i himself;
$\mathsf{acc}_i^{s_i}$	indicates that the protocol instance completed a protocol successfully. That is, whether the involved user accepted the session key or not;
$\mathsf{sk}_i^{s_i}$	stores a distinguished NULL value in the beginning. After a session key is accepted by $\Pi_i^{s_i}$, this session key replaces the NULL value.

We refer to a paper of Bellare et al. [14] for more details on the usage of these variables.

2.1.2. Communication Network

The network is considered to be fully asynchronous and under complete control of the adversary. Arbitrary point-to-point connections among users are available, but the adversary may delay, eavesdrop, insert, and delete messages at will.

2.1.3. Adversarial Capabilities

We restrict to adversaries $\mathcal{A}$ running in probabilistic polynomial time, whose capabilities are made explicit through the four *oracles* listed below. These oracles formalize the interaction between $\mathcal{A}$ and the protocol instances run by the users. For the description of the Test oracle, we denote by b a bit that is chosen uniformly at random.

$\mathsf{Send}(U_i, s_i, M)$	This oracle sends a message M to instance $\Pi_i^{s_i}$ and returns the message generated by this instance. In case the instance $\Pi_i^{s_i}$ is previously unused and the message $M \subseteq \mathcal{P}$ contains a set of user identities, the $\mathsf{used}_i^{s_i}$-flag is set, $\mathsf{pid}_i^{s_i}$ initialized with $\mathsf{pid}_i^{s_i} := \{U_i\} \cup M$. $\Pi_i^{s_i}$ initiates the protocol with the first message which is returned.
$\mathsf{Reveal}(U_i, s_i)$	This outputs the computed key of the instance stored in $\mathsf{sk}_i^{s_i}$.
$\mathsf{Test}(U_i, s_i)$	If the corresponding session key is defined (i. e., $\mathsf{acc}_i^{s_i} = \mathsf{true}$ and $\mathsf{sk}_i^{s_i} \neq \text{NULL}$) and instance $\Pi_i^{s_i}$ is fresh (see Definition 4), $\mathcal{A}$ can execute this oracle query at any time when being activated. Then, if $b = 0$ the session key $\mathsf{sk}_i^{s_i}$ is returned, while if $b = 1$ a uniformly chosen random session key is returned. An arbitrary number of Test queries is allowed for the adversary $\mathcal{A}$, but once the Test oracle returned a value for an instance $\Pi_i^{s_i}$, the same value will be returned for all instances partnered with $\Pi_i^{s_i}$ (see Definition 3).
$\mathsf{Corrupt}(U_i)$	This oracle models forward secrecy, as this query will output the secret signing key of user U_i.

2.2. *Goals of a Key Establishment Protocol: Correctness, Integrity, and Security*

We assume that an instance $\Pi_i^{s_i}$ always accepts the session key constructed at the end of a protocol run if no deviation from the protocol specification has occurred. The subsequent definition of correctness captures the protocol goal that, if the adversary is passive, all users involved in the same protocol session should come up with the same session key. By $\mathcal{A}$ being *passive*, we mean that $\mathcal{A}$ must not use the Corrupt oracle, and may query the Send oracle for the purpose of executing honest protocol executions only.

Definition 1 (Correctness)**.** *A group key establishment protocol* P *is* correct, *if in the presence of a passive adversary* $\mathcal{A}$ *the following holds: for all* i, j *with both* $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ *and* $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$, *we have* $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq \text{NULL}$ *and* $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$.

Unlike correctness, the concept of integrity imposes no restrictions on the adversary's behavior:

Definition 2 (Key Integrity)**.** *A correct group key establishment protocol fulfills* key integrity, *if all instances of users that have accepted with the same session identifier* $\mathsf{sid}_j^{s_j}$ *hold with overwhelming probability identical session keys* $\mathsf{sk}_j^{s_j}$ *and identical partner identifiers* $\mathsf{pid}_j^{s_j}$.

Finally, for defining security, we detail our interpretation of partnering and freshness:

Definition 3 (Partnering)**.** *Instances* $\Pi_i^{s_i}$ *and* $\Pi_j^{s_j}$ *are* partnered *if* $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$, $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, *and* $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.

The idea of freshness is to characterize those instances where the adversary does not know the secret session key for trivial reasons. In particular, note that after revealing a session key from instance $\Pi_i^{s_i}$, the session keys of all instances partnered with $\Pi_i^{s_i}$ are known, too:

Definition 4 (Freshness)**.** *An instance* $\Pi_i^{s_i}$ *is called* fresh *provided that none of the following condition holds:*

- *For some* $U_j \in \mathsf{pid}_i^{s_i}$ *a query* $\mathsf{Corrupt}(U_j)$ *was executed before a query of the form* $\mathsf{Send}(U_k, s_k, *)$ *has taken place where* $U_k \in \mathsf{pid}_i^{s_i}$.
- *The adversary queried* $\mathsf{Reveal}(U_j, s_j)$ *with* $\Pi_i^{s_i}$ *and* $\Pi_j^{s_j}$ *being partnered.*

Now the advantage $\mathsf{Adv}_{\mathcal{A}}(\ell)$ of a probabilistic polynomial time adversary $\mathcal{A}$ in attacking a key establishment protocol P is the function

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

in the security parameter ℓ. Here, $\mathsf{Succ}_{\mathcal{A}}$ denotes the probability that $\mathcal{A}$ queries Test only on fresh instances and correctly outputs the bit b used by the Test oracle while preserving the freshness of all instances queried to Test.

Definition 5. *We say that an authenticated group key establishment protocol* P *is* secure, *if the following inequality holds for every probabilistic polynomial time adversary* $\mathcal{A}$ *some negligible function* $\mathrm{negl}(\ell)$ *in the security parameter* ℓ: $\mathsf{Adv}_{\mathcal{A}}(\ell) \leq \mathrm{negl}(\ell)$

As in [22], our security definition above implies forward secrecy. Specifically, our freshness definition (Definition 4) allows Test queries to an instances, for which the long term secret key has been revealed by a Corrupt query (or is partnered with a instance that has be queried Corrupt) as long as the adversary has not asked a Send query to any of these instances (or their partners) after the Corrupt query.

3. Building on a Group-Theoretic Assumption

As already indicated, we construct our group key establishment protocol in two steps: In Section 3.1 we describe a two-party solution, which subsequently is lifted to an n-party solution by means of the protocol compiler in [22].

3.1. A Two-Party Solution

On the cryptographic side, our two-party solution mainly builds on three technical tools:

- A **non-interactive non-malleable commitment scheme** $\mathcal{C}$, satisfying the following requirements:
  - It is *perfectly binding* in the sense that every commitment can be decommitted to at most one value.
  - It is *non-malleable for multiple commitments*. This means that an adversary who knows commitments to a polynomial sized set of values ν, will not be able to output commitments to a polynomial sized set of values β related to ν in a meaningful way. It is well-known that in the CRS model such a commitment scheme can be implemented by means of any IND-CCA2 secure public key encryption scheme, for instance.
- A **family of universal hash functions** $\mathcal{UH}$ mapping triples consisting of two elements from G and a $\mathsf{pid}_i^{s_i}$-value onto a superpolynomial sized set $\{0,1\}^L$. A universal hash function UH will be selected by the CRS from this family.
- A **collision-resistant pseudorandom function family** $\mathcal{F} = \{F^\ell\}_{\ell \in \mathbb{N}}$(see Katz and Shin [28]). We assume $F^\ell = \{F^\ell_\eta\}_{\eta \in \{0,1\}^L}$ to be indexed by $\{0,1\}^L$ and further denote by $v_0 = v_0(\ell)$ a publicly known value such that no ppt adversary can find two different indices $\lambda \neq \lambda' \in \{0,1\}^L$ such that $F_\lambda(v_0) = F_{\lambda'}(v_0)$. We further use another public value v_1, fulfilling the same requirement as v_0 for deriving the session key (this can also be included in the CRS—see [28] for more details).

Our protocol builds on [21], and for the security proof we have to assume that the underlying group G (respectively, the family of groups $G = G(\ell)$, indexed by the security parameter) satisfies a number of conditions. Besides assuming products and inverses of group elements to be computable by efficient (ppt) algorithms, we further assume G to have a ppt computable canonical representation of elements. The latter allows us to identify group elements with their canonical representation. Furthermore, as in [21], we need three algorithms to perform the computations occurring in a protocol execution:

- DomPar, the *domain parameter generation* algorithm, is a (stateless) ppt algorithm that, upon input of the security parameter 1^ℓ, outputs a finite sequence S of elements in G. The subgroup of G spanned by S, $\langle S \rangle$, will be publicly known. Note that, for the special case of applying our framework to a DDH-assumption, S specifies a public generator of a cyclic group.
- SamAut, the *automorphism group sampling* algorithm, is a (stateless) ppt algorithm that, upon input of the security parameter 1^ℓ 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 efficiently evaluated. For example, for a cyclic group, ϕ could be given as an exponent, or for an inner automorphism the conjugating group element could be specified.
- SamSub, the *subgroup sampling* algorithm, is a (stateless) ppt that, upon input of the security parameter 1^ℓ and a sequence S output by DomPar, returns a word $x(S)$ 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 $\mathrm{Aut}(\langle S \rangle)$. Thus, our protocol requires an explicit representation of x in terms of the generators S.

With this notation, we can now define a decision problem, whose supposed difficulty will be essential for our security proof. As usual, with the notation $o \leftarrow \mathsf{A}(i)$ we describe that algorithm A upon receiving input i outputs o:

Definition 6 (Decision Automorphism Application). *Suppose that we have fixed a quadruple* $(G, \mathsf{DomPar}, \mathsf{SamAut}, \mathsf{SamSub})$. *Then the* decision automorphism application *(DAA) assumption states that for all ppt algorithms* $\mathcal{A}$ *the advantage function* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DAA}} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{DAA}}(\ell) :=$

$$\left|\Pr\left(\mathcal{A}(S,x,(\phi_i(S),\phi_i(x))_{i=1,2}) = 0 \,\middle|\, \begin{array}{ll} S \leftarrow \mathsf{DomPar}(1^\ell), & x \leftarrow \mathsf{SamSub}(1^\ell, S), \\ (\phi_i \leftarrow \mathsf{SamAut}(1^\ell, S))_{i=1,2} & \end{array}\right) - \right.$$
$$\left.\Pr\left(\mathcal{A}(S,r,(\phi_i(S),\phi_i(x))_{i=1,2}) = 0 \,\middle|\, \begin{array}{ll} S \leftarrow \mathsf{DomPar}(1^\ell), & x \leftarrow \mathsf{SamSub}(1^\ell, S), \\ (\phi_i \leftarrow \mathsf{SamAut}(1^\ell, S))_{i=1,2}, & r \leftarrow \mathsf{SamSub}(1^\ell, S) \end{array}\right)\right|$$

is negligible.

Example 1 (Building on decision Diffie–Hellman). *Let* G *be a finite cyclic group and* $S := \langle g \rangle$ *a prime order subgroup with generator* g *of order* q. *If we let* SubSam *choose uniformly at random an exponent* $x \in \{1, \ldots, q-1\}$ *and* SamAut *uniformly at a random exponent* $\phi \in \{1, \ldots, q-1\}$, *then the DAA problem just described can be recognized as polynomial-time equivalent to a decision Diffie–Hellman (DDH) problem:*

"DDH solution ⇒ DAA solution": *When facing, the DAA problem, we obtain as input a tuple* $(g, g^y, (g^{\phi_i}, g^{x\phi_i})_{i=1,2})$ *where either* $y = x$, *or* y *has been chosen uniformly at random from* $\{1, \ldots, q-1\}$*—independently of* x *and the* ϕ_i*s. Given a DDH oracle, we just query it with* $(g, g^y, g^{\phi_1}, g^{x\phi_1})$ *to see with non-negligible success probability which is the case.*

"DDH solution ⇐ DAA solution": *When facing the DDH problem, we obtain as input a tuple* $(g, g^{\phi_1}, g^x, g^y)$, *where either* $y = \phi_1 x \bmod q$, *or* y *has been chosen uniformly at random from* $\{1, \ldots, q-1\}$*—independently of* x *and* ϕ_1. *Choosing another random* $\phi_2 \in \{1, \ldots, q-1\}$, *we can compute the input*

$$(g^{\phi_1}, g^y, ((\underbrace{g}_{=(g^{\phi_1})^{\phi_1^{-1}}}, \underbrace{g^x}_{=(g^{\phi_1 x})^{\phi_1^{-1}}}), (\underbrace{g^{\phi_2}}_{=(g^{\phi_1})^{\phi_1^{-1}\phi_2}}, \underbrace{(g^x)^{\phi_2}}_{=(g^{\phi_1 x})^{\phi_1^{-1}\phi_2}})))$$

needed for a DAA attacker. Running a successful DAA attacker with this input, we immediately obtain the desired DDH attacker.

A two-party key establishment protocol building on the DAA assumption is presented in Figure 1. The figure describes the operations to be performed by instance $\Pi_i^{s_i}$ of U_i. For the sake of readability we name the users trying to establish a common key as U_0 and U_1, and here, as in the sequel, we often omit making explicit the identifiers s_i of the instances $\Pi_i^{s_i}$ involved in the protocol execution and just write sid_i instead of $\mathsf{sid}_i^{s_i}$, for instance. The common reference string is denoted by ρ, and for a commitment to a value x involving random choices r we write $C_\rho(x; r)$. Finally, S denotes the subgroup generators which are to be fixed prior to the protocol execution by means of DomPar (and may also be included in the CRS ρ).

In the subsequent section we prove the following result:

Proposition 1 (Security of the Two-Party Protocol). *Assume that for each ppt time algorithm* $\mathcal{A}$, *its advantage* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sig}}$ *of achieving an existential forgery under the adaptive chosen-message attack for the underlying signature scheme, and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DAA}}$, *its advantage of solving DAA, can be bounded by a negligible function (in* ℓ*). Then the protocol in Figure 1 is a correct and secure two-party key establishment protocol fulfilling key integrity.*

In Figure 2, we describe the group key establishment protocol obtained from a given two party group key establishment protocol 2-AKE via the compiler from [22]. We note here that given the result of Proposition 1, we can apply [22, Theorem 1] (which, as noted by Nam et al. in [29] is only valid if the underlying two party construction fulfills integrity) to obtain our desired security result:

Corollary 1 (Security of the n-Party Protocol)**.** *Denoting the two-party key establishment protocol in Figure 1 by* 2-AKE*, the protocol described in Figure 2 is a secure group key establishment fulfilling key integrity.*

Round 1:

Initialization: For $i = 0, 1$ the variables of involved oracles $\Pi_i^{s_i}$ are set as $\mathsf{pid}_i := \{U_0, U_1\}$, $\mathsf{used}_i := \mathsf{true}$.
Also, for $i = 0, 1$, choose $(\phi_i, (\phi_i)^{-1}) \leftarrow \mathsf{SamAut}(1^\ell, S)$, $x_i \leftarrow \mathsf{SamSub}(1^\ell, S)$.

Computation: User U_i, for $i = 0, 1$ chooses a random r_i and constructs a commitment

$$c_i := C_\rho(x_i; r_i).$$

Communication: User U_i, $i = 0, 1$, sends $m_i^1 := (U_i, \phi_i(S), c_i)$ to U_{1-i}.

Round 2:

Computation: User U_i, $i = 0, 1$, computes $\phi_{1-i}(x_i)$ and a signature σ_i of $(U_i, \phi_{1-i}(x_i))$ (using the representation of $x_i = x_i(S)$ in terms of the generators S and the images $\phi_i(S)$ of the subgroup generators).

Communication: Each user U_i, $i = 0, 1$, sends $m_i^2 := (U_i, \phi_{1-i}(x_i), \sigma_i)$ to U_{1-i}.

Key Generation:

Computation: Compute x_{1-i} by applying $(\phi_i)^{-1}$ to $\phi_i(x_{1-i})$, and define the master key

$$K := (x_0, x_1, \mathsf{pid}_i).$$

Verification: Check the correctness of the commitment c_{1-i} and the signature σ_{1-i}.
If true, set $\mathsf{sk}_i := F_{\mathrm{UH}(K)}(v_1)$, $\mathsf{sid}_i := F_{\mathrm{UH}(K)}(v_0)$ and $\mathsf{acc}_i := \mathsf{term}_i := \mathsf{true}$.
Else set $\mathsf{acc}_i := \mathsf{false}$, $\mathsf{term}_i := \mathsf{true}$

Figure 1. A two-party key establishment protocol in the common reference string (CRS) model.

Round 0:

2-AKE**:** For $i = 0, \ldots, n-1$ execute 2-AKE(U_i, U_{i+1}), (where, as customary, all indices are to be taken mod n, i. e., $U_n = U_0$, etc.).
Thus, each user U_i holds two keys $\overrightarrow{K}_i$, $\overleftarrow{K}_i$. shared with U_{i+1} respectively U_{i-1} and (non-secret) corresponding session identifiers $\overrightarrow{\mathsf{sid}}_i$, $\overleftarrow{\mathsf{sid}}_i$.

Round 1:

Computation: Each U_i computes

$$X_i := \overrightarrow{K}_i \oplus \overleftarrow{K}_i$$

and chooses a random r_i to compute a commitment $C_i = C_\rho(U_i, X_i; r_i)$.

Broadcast: Each U_i broadcasts $M_i^1 := (U_i, C_i)$

Round 2:

Broadcast: Each U_i broadcasts $M_i^2 := (U_i, X_i, r_i)$

Check: Each U_i checks that $X_0 \oplus X_1 \oplus \cdots \oplus X_{n-1} = 0$ and the correctness of the commitments.

Computation: Each U_i sets $K_i := \overleftarrow{K}_i$ and computes the $n-1$ values

$$K_{i-j} := \overleftarrow{K}_i \oplus X_{i-1} \oplus \cdots \oplus X_{i-j} \quad (j = 1, \ldots, n-1),$$

defines a master key

$$K := (K_0, \ldots, K_{n-1}, \mathsf{pid}_i),$$

and sets $\mathsf{sk}_i := F_{\mathrm{UH}(K)}(v_1)$, $\mathsf{sid}_i := F_{\mathrm{UH}(K)}(v_0)$ and $\mathsf{acc}_i := \mathsf{true}$.

Figure 2. The protocol compiler from [22].

3.2. Security Analysis for the Two-Party Case: Proof of Proposition 1

Correctness and Integrity. Due to the collision-resistance of the family $\mathcal{F}$, all oracles that accept with identical session identifier use the same index value $\mathrm{UH}(K)$ and therewith also obtain the same session key and have identical pid_i-values with overwhelming probability.
Security. Let q_s and q_t denote the (polynomially bounded) number of adversarial queries to the Send and Test oracle, respectively.

We consider a simulator simulating all oracles and instances for the adversary. The proof is thus set up following a sequence of experiments or games, where from game to game the simulator's behavior deviates from the previous in a certain controlled way. We follow standard notation and we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary when confronted with Game *i* and by $\mathsf{Succ}(\mathcal{A}, \mathsf{G_i})$ the success probability of $\mathcal{A}$ winning in Game *i*. As usual, the security parameter will be denoted denoted by ℓ.

Game 0. All oracles are simulated as defined in the model. Thus, $\mathsf{Adv}(\mathcal{A}, G_0)$ is exactly $\mathsf{Adv}_{\mathcal{A}}$ and $\mathsf{Succ}(\mathcal{A}, G_0)$ is the probability of violating the security of our key exchange protocol.
Game 1. In this game, the simulator keeps a list with entries (i, M, σ_M) for every message M and corresponding signature σ_M he has produced and returned to the adversary $\mathcal{A}$ in a Round 2 message following a Send query.

By Forge we denote the event that $\mathcal{A}$ queries the Send oracle with a message M containing a valid signature σ_M of an uncorrupted principal U_i and with (i, M, σ_M) not being contained in the simulator's list. If the event Forge occurs, we abort the simulation and take the adversary $\mathcal{A}$ for being successful in breaking the security of the protocol. Thus,

$$|\mathsf{Succ}(\mathcal{A}, G_1) - \mathsf{Succ}(\mathcal{A}, G_0)| \leq P(\mathsf{Forge}) \tag{1}$$

Lemma 1. *If the signature scheme used in the above protocol is existentially unforgeable under adaptive chosen-message attacks, then $P(\mathsf{Forge})$ is negligible: $P(\mathsf{Forge}) \leq |\mathcal{P}| \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sig}}$.*

Proof. Any ppt adversary $\mathcal{A}$ provoking the event Forge can be turned into an attacker against the underlying signature scheme by means of our simulator: The simulator obtains the public verification key PK and access to a signing oracle. In the initialization phase of the protocol, the simulator assigns the key PK uniformly at random to one of the at most $|\mathcal{P}|$ users the adversary can involve. Whenever during the subsequent simulation a signature for this user has to be generated, the simulator queries the signing oracle.

If $\mathcal{A}$ comes up with a message/signature pair that is not stored in the simulator's list, the simulator returns this message as existential forgery. If $\mathcal{A}$ does not come up with such a message, the simulator outputs $\perp$. Having chosen the party U_i uniformly at random, the simulator's success probability for an existential forgery is at least $1/|\mathcal{P}| \cdot P(\mathsf{Forge})$, and we get $P(\mathsf{Forge}) \leq |\mathcal{P}| \cdot \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Sig}}$. □

Thus, from Equation (1), we get

$$|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_0)| \leq \mathrm{negl}(\ell) \tag{2}$$

Game 2. Now the simulation of the Test oracle is modified, so that, on input of a fresh instance, it will always output an element selected uniformly at random in the key space. Thus, $\mathsf{Adv}(\mathcal{A}, G_2) = 0$.

Suppose that $\mathcal{A}$ is able to distinguish between Game 2 and Game 1. We construct an attacker D, that breaks the DAA assumption and uses $\mathcal{A}$ as a black-box. The attacker D will start by setting up the instances with key pairs for the signature scheme and receive a DAA-instance as a challenge. Further, D will choose an index $a \in \{1, \ldots, q_t\}$ uniformly at random and select two values $u, v \in \{1, \ldots, q_s\}$ chosen independently and uniformly at random subject to the condition $u \neq v$. Then the adversary

$\mathcal{A}$ is started. D will simulate the model as in Game 1 except for the uth and vth instance activated by the adversary $\mathcal{A}$ and the answers to the Test query. For the uth and vth instances activated by $\mathcal{A}$, the messages will be constructed from the DAA challenge. If these two instances do not end up in the same session, D aborts the simulation and starts anew. The same happens, if $\mathcal{A}$ does not query his ath Test query to one of these two instances.

D will simulate the Test oracle as follows: The first $a-1$ queries of Test will be answered with the real session key, in the ath query, D will return the challenge, and from query $a+1$ on, D will always answer with a random element.

By a standard hybrid argument, D will win the challenge in $1/q_t$ of the cases where $\mathcal{A}$ distinguished Game 1 and Game 2. Excluding the necessary aborts (namely, if the instances that were chosen were not those used in the ath query of Test), we have:

$$|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)| \leq q_s^2 q_t \mathsf{Adv}^{\mathsf{DAA}} \tag{3}$$

Combining Equations (2) and (3) yields the desired negligible upper bound for $\mathsf{Adv}_{\mathcal{A}}$.

4. Conclusions

Our discussion evidences the possibility of meaningfully integrating tools from group theory and cryptography. Unfortunately, so far we cannot provide a concrete non-abelian example, but a concrete instance of our protocol can be derived by means of the decision Diffie–Hellman assumption. We hope, however, that the modular approach taken above facilitates the design of group key establishment schemes building on group-theoretic tools and fertilizes the exchange of ideas between group theory and cryptography.

Author Contributions: All authors contributed equally to this paper, and were cooperatively involved in conceptualization, investigation, formal analysis and writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was sponsored in part by the NATO Science for Peace and Security Programme under grant G5448 and in part by Spanish MINECO under grant MTM2016-77213-R.

Acknowledgments: This paper was written in grateful memory of our advisor and friend Thomas Beth.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Anshel, I.; Anshel, M.; Goldfeld, D. An Algebraic Method for Public-Key Cryptography. *Math. Res. Lett.* **1999**, *6*, 287–291. [CrossRef]
2. Ko, K.H.; Lee, S.J.; Cheon, J.H.; Han, J.W.; Kang, J.S.; Park, C. New Public-Key Cryptosystem Using Braid Groups. In Proceedings of the Advances in Cryptology—CRYPTO 2000, Santa Barbara, CA, USA, 20–24 August 2000; pp. 166–183.
3. Anshel, I.; Anshel, M.; Fisher, B.; Goldfeld, D. New Key Agreement Protocols in Braid Group Cryptography. In Proceedings of the Topics in Cryptology—CT-RSA 2001, San Francisco, CA, USA, 8–12 April 2001; pp. 13–27.
4. Grigoriev, D.; Ponomarenko, I. Constructions in public-key cryptography over matrix groups. In *Contemporary Mathematics: Algebraic Methods in Cryptography*; American Mathematical Society: Providence, RI, USA, 2006; Volume 418, pp. 103–119.
5. Lee, H.K.; Lee, H.S.; Lee, Y.R. An Authenticated Group Key Agreement Protocol on Braid groups. Cryptology ePrint Archive: Report 2003/018. 2003. Available online: http://eprint.iacr.org/2003/018 (accessed on 1 December 2019).
6. Shpilrain, V.; Ushakov, A. Thompson's Group and Public Key Cryptography. In Proceedings of the ACNS 2005—Third International Conference on Applied Cryptography and Network Security, New York, NY, USA, 7–10 June 2005; Volume 3531, pp. 151–163.

7. Shpilrain, V.; Zapata, G. Combinatorial group theory and public key cryptography. *Appl. Algebra Eng. Commun. Comput.* **2006**, *17*, 291–302. [CrossRef]
8. Shpilrain, V.; Ushakov, A. A new key exchange protocol based on the decomposition problem. In *Contemporary Mathematics: Algebraic Methods in Cryptography*; American Mathematical Society: Providence, RI, USA, 2006; Volume 418, pp. 161–167.
9. Anshel, I.; Anshel, M.; Goldfeld, D.; Lemieux, S. Key agreement, the Algebraic Eraser^{TM}, and lightweight cryptography. In *Contemporary Mathematics: Algebraic Methods in Cryptography*; American Mathematical Society: Providence, RI, USA, 2006; Volume 418, pp. 1–34. [CrossRef]
10. Anshel, I.; Atkins, D.; Goldfeld, D.; Gunnells, P.E. Ironwood Meta Key Agreement and Authentication Protocol. *arXiv* **2017**, arXiv:1702.02450.
11. Bellare, M.; Rogaway, P. Entitiy Authentication and Key Distribution. In Proceedings of the CRYPTO 1993—13th Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, CA, USA, 22–26 August 1993; Volume 773, pp. 232–249.
12. Bellare, M.; Canetti, R.; Krawczyk, H. 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, Dallas, TX, USA, 24–26 May 1998; pp. 319–428.
13. Shoup, V. On Formal Models for Secure Key Exchange (Version 4). Revision of IBM Research Report RZ 3120 (April 1999). 1999. Available online: http://www.shoup.net/papers/skey.pdf (accessed on 1 December 2019).
14. Bellare, M.; Pointcheval, D.; Rogaway, P. Authenticated Key Exchange Secure Against Dictionary Attacks. In Proceedings of the EUROCRYPT 2000—Advances in Cryptology, Bruges, Belgium, 14–18 May 2000; Volume 1807, pp. 139–155.
15. Bresson, E.; Chevassut, O.; Pointcheval, D.; Quisquater, J.J. Provably Authenticated Group Diffie–Hellman Key Exchange. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*; Samarati, P., Ed.; ACM Press: New York, NY, USA, 2001; pp. 255–264.
16. Canetti, R.; Krawczyk, H. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology—EUROCRYPT 2001*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045, pp. 453–474.
17. Ben-Zvi, A.; Blackburn, S.R.; Tsaban, B. A Practical Cryptanalysis of the Algebraic Eraser. In *Advances in Cryptology—CRYPTO 2016 Proceedings, Part I*; Lecture Notes in Computer Science; Robshaw, M., Katz, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9814, pp. 179–189.
18. Cramer, R.; Shoup, V. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Advances in Cryptology—EUROCRYPT 2002*; Lecture Notes in Computer Science; Knudsen, L., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2332, pp. 45–64.
19. González Vasco, M.I.; Martínez, C.; Steinwandt, R.; Villar, J.L. A new Cramer-Shoup like methodology for group based provably secure schemes. In *Proceedings of the 2nd Theory of Cryptography Conference (TCC 2005)*; Lecture Notes in Computer Science; Kilian, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3378, pp. 495–509.
20. Catalano, D.; Pointcheval, D.; Pornin, T. IPAKE: Isomorphisms for Password-based Authenticated Key Exchange. In *Advances in Cryptology—CRYPTO 2004*; Lecture Notes in Computer Science; Franklin, M.K., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3152, pp. 477–493.
21. Bohli, J.M.; Glas, B.; Steinwandt, R. Towards Provably Secure Group Key Agreement Building on Group Theory. In *Proceedings of VietCrypt 2006*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4341, pp. 322–336.
22. Abdalla, M.; Bohli, J.; González Vasco, M.I.; Steinwandt, R. (Password) Authenticated Key Establishment: From 2-Party to Group. In *Proceedings of the 4th Theory of Cryptography Conference TCC 2007*; Lecture Notes in Computer Science; Vadhan, S.P., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4392, pp. 499–514.
23. Burmester, M.; Desmedt, Y. A Secure and Efficient Conference Key Distribution System. In *Advances in Cryptology—EUROCRYPT'94*; Lecture Notes in Computer Science; Santis, A.D., Ed.; Springer: Berlin/Heidelberg, Germany, 1995; Volume 950, pp. 275–286.
24. Gennaro, R.; Lindell, Y. A Framework for Password-Based Authenticated Key Exchange. Cryptology ePrint Archive: Report 2003/032. 2003. Available online: http://eprint.iacr.org/2003/032 (accessed on 1 December 2019).

25. Gennaro, R.; Lindell, Y. A Framework for Password-Based Authenticated Key Exchange (Extended Abstract). In *Advances in Cryptology—EUROCRYPT 2003*; Lecture Notes in Computer Science; Biham, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2656, pp. 524–543.
26. Bohli, J.M.; González Vasco, M.I.; Steinwandt, R. Secure group key establishment revisited. *Int. J. Inf. Secur.* **2007**, *6*, 243–254. [CrossRef]
27. Bohli, J.M.; González Vasco, M.I.; Steinwandt, R. Password-authenticated group key establishment from smooth projective hash functions. *Int. J. Appl. Math. Comput. Sci.* **2019**, *29*, 797–815. Available online: http://eprint.iacr.org/2006/214 (accessed on 1 December, 2019). [CrossRef]
28. Katz, J.; Shin, J.S. Modeling insider attacks on group key-exchange protocols. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*; Atluri, V., Meadows, C.A., Juels, A., Eds.; ACM: New York, NY, USA, 2005; pp. 180–189. Available online: http://eprint.iacr.org/2005/163 (accessed on 1 December, 2019).
29. Nam, J.; Paik, J.; Won, D. A security weakness in Abdalla et al.'s generic construction of a group key exchange protocol. *Inf. Sci.* **2011**, *181*, 234–238. [CrossRef]