(Password) Authenticated Key Establishment: From 2-Party To Group

Michel Abdalla[1], Jens-Matthias Bohli[2], María Isabel González Vasco[3], and Rainer Steinwandt[4]

[1] Departement d'Informatique, École Normale Supérieure, CNRS, 45 Rue d'Ulm, 75230 Paris Cedex 05, France; Michel.Abdalla@ens.fr

[2] Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe, Am Fasanengarten 5, 76128 Karlsruhe, Germany; bohli@ira.uka.de

[3] Departamento de Matemática Aplicada, Universidad Rey Juan Carlos, c/ Tulipán, s/n, 28933, Móstoles, Madrid, Spain; mariaisabel.vasco@urjc.es

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

Abstract. A protocol compiler is described, that transforms any provably secure authenticated 2-party key establishment into a provably secure authenticated group key establishment with 2 more rounds of communication. The compiler introduces neither idealizing assumptions nor high-entropy secrets, e. g., for signing. In particular, applying the compiler to a password-authenticated 2-party key establishment without random oracle assumption, yields a password-authenticated group key establishment without random oracle assumption. Our main technical tools are non-interactive and non-malleable commitment schemes that can be implemented in the common reference string (CRS) model.

Keywords: key establishment, protocol compiler, password-based authentication, common reference string model

1 Introduction

During the last decades, the design of 2-party key establishments has been explored intensively. Certainly not all relevant issues are covered by the available theoretical models, but the techniques at hand proved to be a valuable foundation for the design of practical protocols. On the other hand, the design of group key establishments with $n > 2$ participants is much less understood, and there is a need for significant theoretical progress. In particular for password-authenticated protocols the situation is not very satisfying. A number of protocols have been designed for such a setting, including [24, 1, 2, 28, 13], but it seems to be a non-trivial task to establish strong provable security guarantees without making idealized assumptions.

One valuable tool for breaking down the task of designing a group key establishment protocol into conceptually simpler steps are protocol compilers that build on the security of a given 2-party solution: It seems a plausible design approach to start with a 2-party key establishment and then to apply an efficient compiler which derives the desired n-party solution. Indeed, a number of such generic constructions have been discussed in the literature, including [9, 25, 17]. Remarkably, all proposed constructions rely, to the best of our knowledge, on the use of high-entropy secrets for achieving security against active adversaries. In particular, for the case of password-based authentication in the standard model, no generic 2-to-n compiler seems to be known. The only result in this direction we are aware of is a construction of Abdalla et al. [1, 2] to extend a 2-party solution to the 3-party case.

Our contribution. We describe a compiler that enables the derivation of an authenticated group key establishment protocol from an arbitrary authenticated 2-party key establishment (AKE). In particular, for a password-authenticated 2-party key establishment (PAKE) we obtain a password-authenticated group key establishment. Our compiler does not impose idealizing assumptions or high-entropy secrets for authentication. The suggested construction builds on the use of non-interactive and non-malleable commitments, which in the Common Reference String (CRS) model are known to be implementable through IND-CCA2 secure encryption schemes. For the security proof, we build on a model adapted from [18, 20, 6] which in turn builds on [4, 3]. The structure of our compiler is inspired by the constant-round protocol recently proposed by Bohli et al. [6] which in turn builds on [8, 14, 15]. If the underlying 2-party protocol requires r rounds of communication, the group key establishment output by the compiler takes $r + 2$ rounds.

Organization of the paper. In the next section we recall the basic components of the security framework. We also address some specifics of password-based authentication, a scenario where the application of our protocol compiler seems particularly attractive. Thereafter, we detail the suggested protocol compiler and present the respective security proof. Section 4 indicates some possible applications of our compiler.

2 Security Model and Security Goals

For our compiler, we assume the availability of a common reference string (CRS) which, similarly as in [14, 6], encodes

i) the necessary information for implementing a non-interactive and non-malleable commitment scheme,
ii) a uniformly at random chosen element from a family of universal hash functions and

iii) two values v_0, v_1 that will serve as arguments for a pseudorandom function when computing the session identifier and session key.

The total set of users will be denoted by $\mathcal{P}$ and is assumed to be of polynomial size. By $\mathcal{U} = \{U_1, \ldots, U_n\} \subseteq \mathcal{P}$ we denote the set of protocol participants. We assume that shared (low- or high-entropy) secrets needed for authentication are generated in a trusted initialization phase. During this trusted initialization phase, also possibly needed public keys may be distributed to all potential protocol participants. If authentication is based on shared secrets, we may either assume that each pair of protocol participants $U_i, U_j \in \mathcal{U}$ shares such a secret or that the complete set of protocol participants $\mathcal{U}$ shares one common secret (our compiler is provably secure in either case). We assume that all secrets are chosen independently.

Specifics for password-based authentication. In the case of password-authenticated key establishment, we assume a dictionary $\mathcal{D} \subseteq \{0,1\}^*$ to be publicly available. It is supposed to be efficiently recognizable and of constant or polynomial size. In particular, a polynomially bounded adversary is able to exhaust the complete dictionary $\mathcal{D}$. We assume that all passwords are chosen independently and uniformly at random from $\mathcal{D}$.

2.1 Communication Model and Adversarial Capabilities

As mentioned earlier, our security model is essentially adopted from [6] which in turn builds on [8, 14, 15, 5]. Moreover, as we consider forward secrecy, we also include a Corrupt-oracle. As usual, users are modeled as probabilistic polynomial time (ppt) Turing machines. For our proofs, we may either use uniform or non-uniform Turing machines.

Protocol instances. Each protocol participant $U \in \mathcal{U}$ may execute a polynomial number of protocol *instances* in parallel. A single instance $\Pi_i^{s_i}$ can be interpreted as a process executed by protocol participant U_i. Throughout, the notation $\Pi_i^{s_i}$ ($i \in \mathbb{N}$) will be used to refer to instance s_i of protocol participant $U_i \in \mathcal{U}$. To each instance we assign seven variables:

$\mathsf{used}_i^{s_i}$ indicates whether this instance is or has been used for a protocol run. The $\mathsf{used}_i^{s_i}$ flag can only be set through a protocol message received by the instance due to a call to the Execute- or to the Send-oracle (see below);
$\mathsf{state}_i^{s_i}$ keeps the state information needed during the protocol execution;
$\mathsf{term}_i^{s_i}$ shows if the execution has terminated;
$\mathsf{sid}_i^{s_i}$ denotes a public session identifier that can serve as identifier for the session key $\mathsf{sk}_i^{s_i}$. Note that even though we do not construct session identifiers as session transcripts, the adversary is allowed to learn all session identifiers;
$\mathsf{pid}_i^{s_i}$ stores the set of identities of those users that $\Pi_i^{s_i}$ aims at establishing a key with—including U_i himself;
$\mathsf{acc}_i^{s_i}$ indicates if the protocol instance was successful, i. e., the user accepted the session key;

$\mathsf{sk}_i^{s_i}$ stores the session key once it is accepted by $\Pi_i^{s_i}$. Before acceptance, it stores a distinguished NULL value.

For more details on the usage of the variables we refer to the work of Bellare et al. in [3].

Communication network. We assume arbitrary point-to-point connections among users to be available. The network is non-private and fully asynchronous: The adversary may delay, eavesdrop, insert and delete messages at will.

Adversarial capabilities. We consider ppt adversaries only. Let b be a bit chosen uniformly at random. The capabilities of an adversary $\mathcal{A}$ are made explicit through a number of *oracles* allowing $\mathcal{A}$ to communicate with protocol instances run by the users:

$\mathsf{Send}(U_i, s_i, M)$ This sends message M to the instance $\Pi_i^{s_i}$ and returns the reply generated by this instance. If $\mathcal{A}$ queries this oracle with an unused instance $\Pi_i^{s_i}$ and $M \subseteq \mathcal{P}$ a set of identities of principals, the $\mathsf{used}_i^{s_i}$-flag is set, $\mathsf{pid}_i^{s_i}$ initialized with $\mathsf{pid}_i^{s_i} := \{U_i\} \cup M$, and the initial protocol message of $\Pi_i^{s_i}$ is returned.

$\mathsf{Execute}(\{\Pi_{u_1}^{s_{u_1}}, \dots, \Pi_{u_\mu}^{s_{u_\mu}}\})$ This executes a complete protocol run among the specified unused instances of the respective users. The adversary obtains a transcript of all messages sent over the network. A query to the Execute oracle is supposed to reflect a passive eavesdropping. In particular, for a password-authenticated setting, no online-guess for the secret password can be implemented with a query to this oracle.

$\mathsf{Reveal}(U_i, s_i)$ This yields the value stored in $\mathsf{sk}_i^{s_i}$.

$\mathsf{Test}(U_i, s_i)$ Provided that the session key is defined (i. e. $\mathsf{acc}_i^{s_i} = \mathsf{true}$ and $\mathsf{sk}_i^{s_i} \neq$ NULL) and instance $\Pi_i^{s_i}$ is fresh (see the definition of freshness below), $\mathcal{A}$ can execute this oracle query at any time when being activated. Then, the session key $\mathsf{sk}_i^{s_i}$ is returned if $b = 0$ and a uniformly chosen random session key is returned if $b = 1$. In this model, an arbitrary number of Test queries is allowed for the adversary $\mathcal{A}$, but once the Test oracle returned a value for an instance $\Pi_i^{s_i}$, it will return the same value for all instances partnered with $\Pi_i^{s_i}$ (see the definition of partnering below).

$\mathsf{Corrupt}(U_i)$ This returns all long-term secrets of user U_i. In case of password-based authentication, all passwords held by U_i are returned. In the case of U_i having long-term private keys, e. g., for signing, these private keys are returned.

Remark 1. The model described above seems apparently stronger than those normally used elsewhere since it allows for multiple Test queries. Nevertheless, one can easily show the two notions to be equivalent via a standard hybrid argument with a loss of a factor q in the reduction, with q being the total number of protocol instances. A similar model was also considered by Abdalla et al. in [2] to prove the security of their password-authenticated 3-party key establishment. Fortunately, as pointed out in [2], the loss of a factor q in the reduction can be avoided in most cases as several of the existing schemes (e.g., [19, 20, 15]) already meet this apparently stronger notion of security. This is due to the fact that, in their security proofs, they show that all fresh session keys that can be tested by the adversary are indistinguishable from random.

2.2 Correctness, Integrity and Secrecy

Before we define correctness, integrity and secrecy, we introduce *partnering* to express which instances are associated in a common protocol session.

Partnering. We refer to instances $\Pi_i^{s_i}$, $\Pi_j^{s_j}$ as being *partnered* if $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$, $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$.

To avoid trivial cases, we assume that an instance $\Pi_i^{s_i}$ always accepts the session key constructed at the end of the corresponding protocol run if no deviation from the protocol specification has occurred. Moreover, we want that all users in the same protocol session come up with the same session key, and we capture this in the subsequent notion of correctness.

Correctness. We call a group key establishment protocol P *correct*, if in the presence of a passive adversary $\mathcal{A}$—i. e., $\mathcal{A}$ must neither use the Send nor the $\mathsf{Corrupt}$ oracle—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 $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$.

Key integrity. By definition, correctness takes only passive attacks into account. In contrast, *key integrity* imposes no restrictions on the adversary's oracle access: We say that a correct group key establishment protocol fulfills *key integrity*, if with overwhelming probability all instances of users that have accepted with the same session identifier $\mathsf{sid}_j^{s_j}$ hold identical session keys $\mathsf{sk}_j^{s_j}$ and identical partner identifiers $\mathsf{pid}_j^{s_j}$.

Next, for detailing the security definition, we will have to specify under which conditions a Test-query may be executed.

Freshness. A Test-query should only be allowed to those instances holding a key that is not for trivial reasons known to the adversary. To this aim, an instance $\Pi_i^{s_i}$ is called *fresh* if none of the following 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 $\mathsf{Send}(U_k, s_k, M)$ has taken place, for some message (or set of identities) M and some $U_k \in \mathsf{pid}_i^{s_i}$.
- The adversary earlier queried $\mathsf{Reveal}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered.

The idea of this definition is that revealing a session key from an instance $\Pi_i^{s_i}$ trivially yields the session key of all instances partnered with $\Pi_i^{s_i}$, and hence this kind of "attack" will be excluded in the security definition.

Security/key secrecy. For a secure group key establishment protocol, we have to impose a corresponding bound on the adversary's *advantage*: The advantage $\mathsf{Adv}_{\mathcal{A}}(\ell)$ of a ppt adversary $\mathcal{A}$ in attacking protocol P is a function in the security parameter ℓ, defined as

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

Here Succ is the probability that the adversary queries Test only on fresh instances and guesses correctly the bit b used by the Test oracle (without violating the freshness of those instances queried with Test) :

In the case of password-authenticated key establishment, due to the polynomial size of the dictionary $\mathcal{D}$, we cannot prevent an adversary from correctly guessing shared passwords with non-negligible probability. Thus, for the password-authenticated setting, our goal is to restrict the adversary $\mathcal{A}$ to online-verification of password guesses, namely, to prove that $\mathsf{Adv}_{\mathcal{A}}$ is only negligibly above the probability $\mathcal{A}$ has guessed a shared password online. We introduce a function ε to capture such weaknesses that originate in the employed authentication technique. For the password case, ε should bound $\mathcal{A}$'s probability of guessing a shared password, assuming he is not able to test (online) more than a constant number of passwords per protocol instance.

Remark 2. Following the spirit of [14, 6], it would be desirable to restrict the number of passwords that can be guessed online to *one* per protocol instance. As described, our compiler accesses the underlying authenticated 2-party key establishment as a black-box only, and our security proof does not guarantee that only one password can be verified per instance. For specific instances a tighter security reduction may be possible, however.

Definition 3. *We say that an authenticated group key establishment protocol* P *is* ε-secure*, if for every ppt adversary* $\mathcal{A}$ *the following inequality holds for some negligible function* negl*:*

$$\mathsf{Adv}_{\mathcal{A}}(\ell, q_{\mathrm{send}}) \leq \varepsilon(\ell, q_{\mathrm{send}}) + \mathrm{negl}(\ell), \tag{1}$$

where ℓ *is the security parameter and* q_{send} *is the number of different protocol instances* $\mathcal{A}$ *queries the* Send *oracle with. The function* ε *is expected to be at most linear in its second variable, i.e. the number of* Send *queries.*

Forward Secrecy. We follow the spirit of the definition of forward secrecy from [19], yet our definition is weaker: we consider the "weak corruption model" of [3] in which corrupting a principal means only retrieving his long term secret keys. Forward secrecy is then achieved if such corruption does not give the adversary any information about previously agreed session keys. This same approach has also been taken in [7, 16].

Remark 4. Note that our definition of freshness allows for Test queries to instances such that their (or their partners') long term secret keys have been revealed to the adversary by a $\mathsf{Corrupt}$ query as long as no Send query has been asked to any of these instances (or their partners) after the $\mathsf{Corrupt}$ query. Thus, the above definition of ε-security implies forward secrecy in this sense.

3 From Two to Group: A Compiler

In this section, we describe how an n-party AKE can be derived from any 2-party AKE carrying over its essential security properties. Our compiler assumes the availability of a 2-party key establishment that is ε-secure in the sense of Definition 3, where ε is defined according to the authentication method used. Our construction then yields an $\hat{\varepsilon}$-secure n-party AKE where $\hat{\varepsilon}$ is bounded by $4 \cdot \varepsilon$.

3.1 Tools

For the actual compiler, black-box access to the authenticated 2-party key establishment suffices, and Fig. 1 captures this access with an oracle $\mathsf{2\text{-}AKE}(\cdot,\cdot)$ that upon input of two principals $U_i, U_j \in \mathcal{P}$ (or rather their identities), returns the respective output of the 2-party protocol. We assume this output to be either a secret key $\kappa \in \{0,1\}^k$ or a special symbol $\top$ indicating that the key establishment failed (due to adversarial interference). Additionally, the tools involved in our construction are:

– **a non-interactive non-malleable commitment scheme [12]** $\mathcal{C}$, fulfilling the following requirements:
  1. it must be *perfectly binding*, i. e., every commitment c defines at most one value $\mathsf{decommit}(c)$;
  2. it must achieve *non-malleability for multiple commitments*—if an adversary receives commitments to a (polynomial sized) set of values ν he must not be able to output commitments to a (polynomial sized) set of values β related to ν in a known way.

  Note that in the CRS model with a common reference string ρ, the above commitment schemes $\mathcal{C} = \mathcal{C}_\rho$ can be constructed from any public key encryption scheme that is non-malleable and secure for multiple encryptions (in particular, from any IND-CCA2 secure public key encryption scheme).

– **a collision-resistant pseudorandom function family** $\mathcal{F} = \{F^\ell\}_{\ell \in \mathbb{N}}$ as used by Katz and Shin [21]. We assume $F^\ell = \{F^\ell_\eta\}_{\eta \in \{0,1\}^L}$ to be indexed by a superpolynomial sized set $\{0,1\}^L$ and denote by $v_0 = v_0(\ell)$ a publicly known value such no ppt adversary can find two different indices $\lambda \neq \lambda' \in \{0,1\}^L$ such that $F_\lambda(v_0) = F_{\lambda'}(v_0)$. For deriving the session key we use another public value v_1 which fulfills the above collision-resistance condition as well and is also encoded in the CRS (see [21] for more details).
– **a family of universal hash functions** $\mathcal{UH}$ that maps the concatenation of bitstrings from $\{0,1\}^{kn}$ and a partner $\mathsf{pid}_i^{s_i}$ onto $\{0,1\}^L$. The CRS selects one universal hash function UH from this family. We use UH to select an index within the aforementioned collision-resistant pseudorandom function family.

3.2 Design Rationale

The idea of our compiler is inspired in the classical construction of Burmester and Desmedt [8], where the trick of constructing a group key from pairwise agreed keys among the group principals was first introduced. Further, our construction in some sense generalizes the design of [6], that builds an n-party PAKE on Gennaro and Lindell's 2-party PAKE. Once the pairwise key establishments have been completed, each principal must commit to the XOR-value of the two keys he shares with his neighbors. This value is disclosed in a subsequent round, allowing all principals to derive each of the 2-party keys, from which both the session identifier and the session key will be derived. Intuitively, if an adversary has not been able to pervert the security of any of the 2-party protocol executions involved, neither will he be able to retrieve any information about the resulting group session key (for XORs of "randomly looking" elements should look as well random to him). Moreover, integrity is also provided by an argument similar to the one in [6].

The compiler does not rely on further authentication techniques than those used in the basic 2-party AKE protocol, neither on any further idealization assumption. Also, our design is symmetric in the sense that all users perform the same steps. Fig. 1 shows the three rounds of our construction, adding 2 rounds to those of the underlying 2-party AKE. For the sake of readability, we do not explicitly refer to instances s_i of users. Also, we omit the $\mathsf{pid}_i^{s_i}$-values, assuming that when the protocol is initiated (via a Send or $\mathsf{Execute}$ call) each participant involved receives a message informing him of the actual pid of the session, which in addition makes him aware of his position in the "cycle" of involved principals and therefore the $\mathsf{2\text{-}AKE}$ step (Round 0) can be performed accordingly.

Remark 5. The compiler can be applied to any polynomial number of participants $n \geq 2$. The case $n = 2$ is not excluded, but to some extent pathological: Here the compiler executes the underlying 2-party AKE *twice*, so that each party obtains two independent keys $\overrightarrow{K}_i$, $\overleftarrow{K}_i$, which are then combined to form the actual session key.

3.3 Security Analysis

Assume that we are given a correct and secure authenticated 2-party key establishment protocol. Assume further that $\mathcal{C}$ is a non-interactive non-malleable commitment scheme and $\mathcal{F}$ a collision-resistant pseudorandom function family. In the following, we show that under these assumptions the compiler in Fig. 1 yields a correct and secure group key establishment. In particular, this is true when the underlying 2-party AKE protocol is based on passwords.

Theorem 6. *Let $\mathcal{F}$ be a family of secure collision-resistant pseudorandom functions, let $\mathcal{C}$ be a non-interactive perfectly binding non-malleable commitment scheme, and let $\mathsf{2\text{-}AKE}$ be a correct and ε-*secure *authenticated 2-party key establishment protocol. Then the protocol in Fig. 1 is a correct and $4 \cdot \varepsilon$-*secure *authenticated group key establishment protocol, which also provides key integrity.*

Round 0:

2-AKE: For $i = 1, \ldots, n$ execute 2-AKE(U_i, U_{i+1}).[a] Thus, each user U_i holds two keys $\overrightarrow{K}_i$, $\overleftarrow{K}_i$ shared with U_{i+1} respectively U_{i-1}.

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 $C_i = C_\rho(i, X_i; r_i)$.

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 $X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0$ and the correctness of the commitments. If at least one of these checks fails, set $\mathsf{acc}_i := \mathsf{false}$ and terminate the protocol execution.

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

$$K := (K_1, \ldots, K_n, \mathsf{pid}_i),$$

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

[a] All indices are to be taken in a cycle, i. e., $U_{n+1} = U_1$, etc.

Fig. 1. A protocol compiler.

Proof. Correctness. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same session identifier and session key.

Integrity. Owing to the collision-resistance of the family $\mathcal{F}$, all oracles that accept with identical session identifiers use with overwhelming probability the same index value $\mathrm{UH}(K)$ and therewith also derive the same session key and have identical partner identifiers.

Key secrecy. The proof of key secrecy will proceed in a sequence of games, starting with the real attack against the key secrecy of the group key exchange protocol and ending in a game in which the adversary's advantage is 0, and for which we can bound the difference in the adversary's advantage between any two consecutive games. Following standard notation, we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary $\mathcal{A}$ in Game i. Furthermore, for clarity, we classify the Send queries into 3 categories, depending on the stage of the protocol to which the query is associated, starting with Send-0 and ending with Send-2. Send-t denotes the Send query associated with round t for $t = 0, 1, 2$.

Game 0. This first game corresponds to a real attack, in which all the parameters, such as the public parameters in the common reference string and the long-term secrets associated with each user, are chosen as in the actual scheme. By definition, $\mathsf{Adv}(\mathcal{A}, G_0) = \mathsf{Adv}(\mathcal{A})$.

Game 1. In this game, for $i = 1, \ldots, n$, we modify the simulation of the Send and $\mathsf{Execute}$ oracles so that, whenever an instance $\Pi_i^{s_i}$ is still considered fresh at the end of Round 0, the keys $\overleftarrow{K}_i$ and $\overrightarrow{K}_i$ that it shares with instances $\Pi_{i-1}^{s_{i-1}}$ and $\Pi_{i+1}^{s_{i+1}}$ are replaced with random values from the appropriate set. An instance $\Pi_i^{s_i}$ is considered fresh at the end of Round 0 if it has not halted or rejected and if no query $\mathsf{Corrupt}(U_j)$ for some $U_j \in \mathsf{pid}_i^{s_i}$ has been asked by the adversary before a query of the form $\mathsf{Send}(U_k, s_k, M)$ for some $U_k \in \mathsf{pid}_i^{s_i}$ and some message M.

Note that the distance between this game and the previous one is bounded by the probability that the adversary breaks the security of any of the underlying 2-AKE protocols. More precisely, we have

$$\left|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_0)\right| \leq 2 \cdot \mathsf{Adv}_{\mathsf{2\text{-}AKE}}(\ell, 2 \cdot q_{\mathrm{send}}),$$

where q_{send} represents the number of *different* protocol instances in Send queries. The factor 2 multiplying q_{send} emerges because one instance in the group key protocol builds on two instances of the 2-AKE protocol for the key establishment with the right and left neighbor, respectively. The other factor 2 is due to the security definition which states that the advantage of an adversary is twice its success probability minus 1.

To prove this, we show how an adversary $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ is constructed from a given adversary $\mathcal{A}$ distinguishing Game G_1 from Game G_0.

$\mathcal{A}_{\mathsf{2\text{-}AKE}}$ is given access to a simulation of the 2-AKE protocol as outlined in Section 2. To answer its queries, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ will associate each user instance $\Pi_i^{s_i}$ in the group protocol with two independent instances of the same user in the 2-AKE protocol. Now, whenever $\mathcal{A}$ makes a $\mathsf{Corrupt}$ query, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ answers it by querying the $\mathsf{Corrupt}$ oracle of the 2-AKE protocol and returns the same answer. To answer an $\mathsf{Execute}$ query, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ first queries the $\mathsf{Execute}$ oracle of the 2-AKE protocol with the corresponding instances to obtain the transcript for Round 0. To simulate the following rounds, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ first queries the Test oracle of the 2-AKE protocol with the corresponding instances and uses the returned values as the keys $\overleftarrow{K}_i$ and $\overrightarrow{K}_i$. To answer a $\mathsf{Send\text{-}0}$ query, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ queries the Send oracle of the 2-AKE protocol with the corresponding instance and returns its response. To answer Send queries pertaining rounds 1 and 2, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ first sets the values of the keys $\overleftarrow{K}_i$ and $\overrightarrow{K}_i$ by querying either the Test or Reveal oracle of the 2-AKE protocol with the corresponding instances and proceeds with the simulation as in the previous game. More precisely, if an instance $\Pi_i^{s_i}$ in the group protocol is still considered fresh at the beginning of Round 1, then $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ queries the Test oracle of the 2-AKE protocol with the corresponding instances in the 2-AKE protocol. Otherwise, $\mathcal{A}_{\mathsf{2\text{-}AKE}}$ queries the Reveal oracle.

Finally, one can easily see that the view of $\mathcal{A}$ corresponds to Game G_0 if Test reveals the actually exchanged key and to Game G_1 if Test returns a random element from the key space. Thus, $\mathcal{A}$ succeeds distinguishing Game G_0 and Game G_1 with a probability of at most $\mathsf{Adv}_{\mathsf{2\text{-}AKE}}(\ell, 2 \cdot q_{\text{send}})$.

Game 2. In this game, we change the simulation of the Send oracle so that a *fresh* instance $\Pi_i^{s_i}$ does not accept in Round 2 whenever one commitment C_j for $j \neq i$ it receives in Round 1 was generated by the simulator but not generated by the respective instance $\Pi_j^{s_j}$, $j \neq i$ in the same session. At this, we take two instances $\Pi_{\alpha_0}^{s_{\alpha_0}}$, $\Pi_{\alpha_r}^{s_{\alpha_r}}$ for being in the *same session*, if there is a sequence of instances $(\Pi_{\alpha_\mu}^{s_{\alpha_\mu}})_{0 \leq \mu \leq r}$ such that for each $\mu = 0, \ldots, r-1$ the instances $\Pi_{\alpha_\mu}^{s_{\alpha_\mu}}$ and $\Pi_{\alpha_{\mu+1}}^{s_{\alpha_{\mu+1}}}$ are partnered through an execution of the underlying 2-party key establishment (i.e., they hold a common 2-party session key $\overrightarrow{K}_{\alpha_\mu} = \overleftarrow{K}_{\alpha_{\mu+1}}$ associated with the same session identifier and the same two protocol participants).

The adversary $\mathcal{A}$ can detect the difference to Game G_1 if $\mathcal{A}$ replayed a commitment that should have led to acceptance in Round 2 in that game. Because the committed value X_i is a random value independent of previous messages, the probability for this is negligible.

$$\left|\mathsf{Adv}(\mathcal{A}, G_2) - \mathsf{Adv}(\mathcal{A}, G_1)\right| \leq \text{negl}(\ell)$$

To see why, note that given one session, an instance $\Pi_i^{s_i}$ expects commitments C_j to (j, X_j), such that $X_1 \oplus \cdots \oplus X_n = 0$. $\Pi_i^{s_i}$ will only accept with negligible probability if all commitments where generated by the simulator, however, not being exactly the commitments C_j, $j \neq i$ by the respective oracles $\Pi_j^{s_j}$, $j \neq i$ of the session. This can be seen as follows: The equation

$$X_1 \oplus \cdots \oplus X_n = 0$$

results in

$$\overleftarrow{K}_1 \oplus \overrightarrow{K}_1 \oplus \cdots \oplus \overleftarrow{K}_n \oplus \overrightarrow{K}_n = 0.$$

For $\Pi_i^{s_i}$, $X_i = \overleftarrow{K}_i \oplus \overrightarrow{K}_i$ is given where $\overleftarrow{K}_i$ is shared with U_{i-1} and $\overrightarrow{K}_i$ is shared with U_{i+1}. Because the commitment C_j includes the index of user U_j and is perfectly binding, the adversary $\mathcal{A}$ cannot reveal the commitments if they are permuted within the participants of the session. As by now all keys are random values, the probability for any XOR sum of keys not consisting exactly of the keys in one session (thus canceling each other w.r.t. XOR) to be 0 is only $1/2^k$. The adversary $\mathcal{A}$ is at maximum capable of doing this q_{send} times, giving him a probability $q_{\text{send}}/2^k$ of distinguishing the games.

Game 3. This game reproduces the modification also for *adversary-generated* commitments: The simulation of the Send oracle changes so that a *fresh* instance $\Pi_i^{s_i}$ does not accept in Round 2 whenever one commitment C_j for $j \neq i$ it receives in Round 1 was *adversary-generated*. The adversary's advantage diverges only negligibly from the previous game:

$$\left|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_2)\right| \leq \text{negl}(\ell)$$

To prove this, we construct a malleability attacker $\mathcal{A}_{COM}$ to the commitment scheme from an adversary $\mathcal{A}$ that comes up with a commitment C_j to $\Pi_i^{s_i}$ such that $\Pi_i^{s_i}$ would accept in Game G_2 but not in Game G_3. Our goal is to show that the probability with which $\mathcal{A}_{COM}$ succeeds in outputting a related vector of commitments is related to the probability with which $\mathcal{A}$ can distinguish Games G_3 from G_2.

$\mathcal{A}_{COM}$ is given commitments $C_i = C_\rho(i, X_i; r_i)$ for $i = 1, \ldots, n$ where the X_i values are random bitstrings fulfilling $X_1 \oplus \cdots \oplus X_n = 0$. For bitstrings X'_i, $i = 1, \ldots, n$, the $2n$-ary relation is given by

$$\mathcal{R}(X_1, \ldots, X_n, X'_1, \ldots, X'_n) = 1$$

if and only if

$$X'_1 \oplus \cdots \oplus X'_n = 0 \text{ and } X_i = X'_i \text{ for at least one index } i \in \{1, \ldots, n\}.$$

$\mathcal{A}_{COM}$ starts by guessing the first instance $\Pi_i^{s_i}$ to receive from $\mathcal{A}$ a set of commitments C'_j, for $j \neq i$, with at least one of these commitments being *adversary-generated*. For all sessions other than the one in which $\Pi_i^{s_i}$ is involved, $\mathcal{A}_{COM}$ simulate the oracles exactly as it would in Game G_2. For the session in which $\Pi_i^{s_i}$ is involved, $\mathcal{A}_{COM}$ uses the C_i values that it has received as input to answer $\mathsf{Send\text{-}1}$ queries. Then, as soon as $\mathcal{A}$ provides $\Pi_i^{s_i}$ with a set of commitments C'_j for $j \neq i$, then $\mathcal{A}_{COM}$ halts the simulation and outputs this set along with C_i.

One can easily see that $\mathcal{A}_{COM}$ will succeed in outputting a set of related commitments satisfying the relation $\mathcal{R}$ if it guesses correctly the first instance to receive a set of commitments containing at least one *adversary-generated* commitment and passing the verification test. This is true because games G_3 and G_2 are indistinguishable up to that point and the simulation of the oracles by $\mathcal{A}_{COM}$ is perfect.

By definition of non-malleability, the success probability of $\mathcal{A}_{COM}$ is only negligibly greater than that of an adversary who does not see the list of commitments C_i for $i = 1, \ldots, n$. If no commitments are given, an adversary's probability to send valid commitments C_j for $j \neq i$ such that $X'_1 \oplus \cdots \oplus X_i \oplus \cdots \oplus X'_n = 0$ is $q_{\text{send}}/2^k$ as in the previous game. As a result, the non-malleability of the commitment scheme guarantees that the adversary's success probability with access to these commitments is negligibly close to $q_{\text{send}}/2^k$, thus, being negligible in total.

Game 4. Now the simulation of the $\mathsf{Execute}$ and Send oracles are modified at the point of computing the session key. The simulator keeps a list of assignments $(K_1, \ldots, K_n, \mathsf{sk}_i^{s_i})$. Once an instance receives the last $\mathsf{Send\text{-}2}$ query, the simulator computes $K_1, \ldots, K_n$ and checks if for this sequence a master key was already issued and assigns this key to the instance. If no such entry exists in the list, the simulator chooses a session key $\mathsf{sk}_i^{s_i} \in \{0,1\}^\ell$ uniformly at random.

The master key $K = (K_1, \ldots, K_n, \mathsf{pid}_i^{s_i})$ has, once the X_i are public, sufficient entropy such that the output of the pseudorandom function $F_{\mathrm{UH}(K)}$ is distinguishable from a random $\mathsf{sk}_i^{s_i}$ with negligible probability only.

$$\left|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_3)\right| \le \mathrm{negl}(\ell).$$

In Game G_4, all session keys are chosen uniformly at random and the adversary has no advantage.

$$\mathsf{Adv}(\mathcal{A}, G_4) = 0.$$

□

4 Applications and Comments

The above compiler allows for the construction of very efficient authenticated group key exchange protocols adding up to the "base" 2-AKE only two rounds of communication. As we have remarked, our compiler adds neither any authentication tool nor any additional idealization assumptions to the base scheme.

Example 7. Applying our compiler to a password-authenticated 2-party key establishment offering forward secrecy, we immediately obtain a forward secure password-authenticated group key establishment. It should be pointed out here, however, that stronger notions of forward secrecy than ours can be considered [19]. Actually, it is an interesting question to explore whether the KOY 2-AKE from [19] (or variants of it) can be proven secure in our model—therewith yielding through application of our compiler the first forward secure password-authenticated group key establishment.

Of course, our compiler can also be applied in the random oracle model—in practice this means to replace the "full-fledged" commitment scheme and the family of collision resistant pseudorandom functions through the (more efficient) use of a cryptographic hash function (cf. [21]). Going one step further, from an engineering perspective it is tempting to apply the compiler to an efficient authenticated 2-party key establishment, even if no security proof in the above model is available. Of course, in this case our security analysis does not yield a provable security statement on the resulting group key establishment.

Example 8. A natural starting point for applying our compiler would be the (H)MQV family discussed in [27, 23, 26, 22]. The resulting scheme could be rather efficient in practice, but the available formal security analysis builds on a model due to Canetti and Krawczyk [11]. We have not attempted to carry out a security analysis in the model underlying the above discussion and consequently cannot claim provable security guarantees of a derived group key establishment.

5 Conclusions

The compiler we presented allows the construction of authenticated group key establishment schemes based on any provably secure authenticated 2-party key establishment. At this forward secrecy is taken into account, and the suggested compiler does not introduce new idealizing assumptions or tools for authentication, like an existentially unforgeable signature scheme. In terms of efficiency, adding only two additional rounds to a 2-party solution seems acceptable, too, and renders the compiler an interesting tool for practical protocol design.

Both from a theoretical and from a practical point of view, it seems worthwhile to explore the tightness of the above security proof more closely, when applying the compiler to specific protocols. In the described form, the compiler restricts to black-box access to the underlying two-party key establishment, but for a specific use case, there is no need for such a restriction.

Also, we have not explored the behaviour of our compiler within the universal composability framework. In particular, it would be interesting to explore the security level achieved applying our compiler to universally composable password based two party key exchange protocols, along the lines of [10].

Acknowledgements

The first author was supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure.

References

1. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In Serge Vaudenay, editor, *Public Key Cryptography – PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2005.
2. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. *IEE Proceedings – Information Security*, 153(1):27–39, March 2006.
3. 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.
4. 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.
5. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Secure Group Key Establishment Revisited. Cryptology ePrint Archive, Report 2005/395, 2005. Available at `http://eprint.iacr.org/2005/395/`.
6. Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Password-Authenticated Constant-Round Group Key Establishment with a Common Reference String . Cryptology ePrint Archive: Report 2006/214, 2006. Available at `http://eprint.iacr.org/2006/214`.
7. Victor Boyko, Philip D. MacKenczie, and Sarvar Patel. Provable-Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2000.

8. Mike Burmester and Yvo Desmedt. A Secure and Efficient Conference Key Distribution System. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1995.
9. Mike Burmester and Yvo G. Desmedt. Efficient and Secure Conference-Key Distribution. In T. Mark A. Lomas, editor, *Proceedings of the International Workshop on Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 119–129. Springer, 1996.
10. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip MacKenzie. Universally Composable Password-Based Key Exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3495 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2005.
11. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.
12. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.
13. Ratna Dutta and Rana Barua. Password-Based Encrypted Group Key Agreement. *International Journal of Network Security*, 3(1):23–34, July 2006.
14. Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange. Cryptology ePrint Archive: Report 2003/032, 2003. Available at `http://eprint.iacr.org/2003/032`.
15. Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange (Extended Abstract). In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, 2003.
16. Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *Advances in Cryptology – CRYPTO '01*, pages 408–432, London, UK, 2001. Springer-Verlag.
17. Jung Yeon Hwang, Su-Mi Lee, and Dong Hoon Lee. Scalable key exchange transformation: from two-party to group. *Electronic Letters*, 40(12):728–729, 2004.
18. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2001.
19. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Forward Secrecy in Password-Only Key Exchange Protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks: Third International Conference, SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 2003.
20. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient and Secure Authenticated Key Exchange Using Weak Passwords, 2006. Available at `http://www.cs.umd.edu/~jkatz/papers/password.pdf`.
21. Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. Cryptology ePrint Archive: Report 2005/163, 2005. Available at `http://eprint.iacr.org/2005/163`.
22. Hugo Kawczyck. HMQV: A High-Performance Secure Diffie-Hellman Protocol. Cryptology ePrint Archive: Report 2005/176, 2005. Available at `http://eprint.iacr.org/2005/176`.

23. Hugo Krawczyck. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.
24. Su Mi Lee, Jung Yeon Hwang, and Dong Hoon Lee. Efficient Password-Based Group Key Exchange. In Sokratis Katsikas, Javier Lopez, and Günther Pernul, editors, *Trust and Privacy in Digital Business: First International Conference, TrustBus 2004*, volume 3184 of *Lecture Notes in Computer Science*, pages 191–199. Springer, 2004.
25. Alain Mayer and Moti Yung. Secure Protocol Transformation via "Expansion": From Two-party to Groups. In *Proceedings of the 6th ACM conference on Computer and Communications Security CCS '99*, pages 83–92. ACM Press, 1999.
26. Alfred Menezes. Another look at HMQV. Cryptology ePrint Archive: Report 2005/205, 2005. Available at `http://eprint.iacr.org/2005/205`.
27. Alfred Menezes, Minghua Qu, and Scott A. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Workshop on Selected Areas in Cryptography*, pages 22–32, July 1995.
28. Qiang Tang and Kim-Kwang Raymond Choo. Secure password-based authenticated group key agreement for data-sharing peer-to-peer networks. In *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2006.