Group key exchange protocols withstanding ephemeral-key reveals

María Isabel González Vasco[*], Angel L. Pérez del Pozo

Dpto. de Matemática Aplicada, Universidad Rey Juan Carlos
C/ Tulipán s/n. 28933, Móstoles, Madrid, Spain
{mariaisabel.vasco, angel.perez}@urjc.es

Adriana Suarez Corona
Departamento de Matemáticas, Universidad de León
Escuela de Ingenierías Industrial e Informática
Campus de Vegazana, s/n, 24071 León, Spain
asuac@unileon.es

December 20, 2016

Abstract

When a group key exchange protocol is executed, the session key is typically extracted from two types of secrets; long-term keys (for authentication) and freshly generated (often random) values. The leakage of this latter so-called *ephemeral keys* has been extensively analyzed in the 2-party case, yet very few works are concerned with it in the group setting. We provide a generic group key exchange construction that is strongly secure, meaning that the attacker is allowed to learn both long-term and ephemeral keys (but not both from the same participant, as this would trivially disclose the session key). Our design can be seen as a compiler, in the sense that it builds on a 2-party key exchange protocol which is strongly secure and transforms it into a strongly secure group key exchange protocol by adding only one extra round of communication. When applied to an existing 2-party protocol from Bergsma et al., the result is a 2-round group key exchange protocol which is strongly secure in the standard model, thus yielding the first construction with this property.

1 Introduction

Group key establishment (GKE) protocols are a fundamental cryptographic building block allowing $n \geq 2$ participants to agree upon a common secret key. It is usually assumed that these participants hold both *long-term* secrets, which are typically used for authentication and *ephemeral* secrets, which are session-specific randomly generated values that provide enough entropy for the key to be indistinguishable from random in some sense.

The way to define and handle key privacy in GKE is highly dependent on the amount of information the adversary is supposed to obtain from the two types of secrets described above. In the literature, leakage of ephemeral secrets is often modeled through a $\mathsf{RevealState}$ oracle, which when invoked by the adversary outputs either *ephemeral keys* as described above or a larger set containing them, typically referred to as the *full state* of the attacked user. Unfortunately, as first pointed out by Cremers in [16], the meaning of *full state* is scarcely defined within the security model and often the output of the corresponding oracle calls is only made explicit when proving particular protocols secure. Generally speaking, ephemeral key leakage refers to the exposure of user-generated fresh randomness, while full state compromise involves in addition other values computed/stored by the user —yet never any long-term keys.

[*] contact author

Previous work. *Strong security* for GKE protocols was first considered in [9]. As it is also the case with subsequent proposals dealing with this type of leakage [8, 24, 26], in this paper it is assumed that the adversary can not access any ephemeral secret *of the attacked session*.

In order to subsume a wider class of attacks, other works have removed this restriction excluding only reveals of *both* the ephemeral and long-term secrets of the same user (as, in this case, the session key would be trivially disclosed). Some examples of secure proposals along these lines are the NAXOS protocol [35] in the 2-party setting and [36, 20] for the case of 3 users. In the general multi-user group setting Zhao et al. [42] modified a protocol by Bohli et al. [6] to obtain strong security. This proposal was however found flawed in [14] where an improvement was proposed, which is proven secure in the random oracle model.

Many of these previous works have in common that the access to the ephemeral secrets is modelled granting the adversary a $\mathsf{RevealState}$ oracle, which, when queried, outputs the contents of a variable state linked to the execution. As pointed out by Cremers [16], the security in these models is highly dependent on how the state variable is defined for each concrete protocol. In addition Cremers shows that the NAXOS protocol, proven secure in the model with a different formalism (namely, defining a so-called $\mathsf{RevealEphemeralKey}$ oracle), is insecure when more powerful state reveals are allowed. Also in this spirit, in a recent work from PKC2015, Bergsma et al. [5] present a generic 1-round 2-party key exchange construction in the standard model. The authors also propose a strong security model which builds on previous ones and captures both perfect forward secrecy and ephemeral secrets leakage. The latter is modeled by a $\mathsf{RevealRand}$ oracle which outputs the local randomness selected by the user in a protocol execution.

Our contributions. We propose a security model for GKE capturing the leakage of ephemeral secrets even within the attacked session. To avoid any ambiguity we define, in the line of [5] in the 2-party setting, a variable rand that stores, for each instance of a participant, all the session values that cannot be computed from long-term secret keys or other values received/computed previously in the session. Typically these values are chosen uniformly at random from a prescribed set, therefore the name of the variable. The adversary is given access to an oracle $\mathsf{RevealRand}$ which outputs the values stored in rand when queried. The strength of our security model is comparable to that of [42], yet in their treatment ephemeral values involved in the authentication procedure are not included in rand as they are in our case.[1]

In addition, the main contribution of this work is a generic protocol or *compiler* that, building on any strongly secure 2-party authenticated key exchange (AKE) protocol, produces a group AKE which is strongly secure in our model, by adding only one round of communication.[2] Further, we highlight that:

- Our construction is the first to expressly take into account the randomness used for authentication. We do so by expliciting that the nonces involved in any signature produced are part of the rand variable linked to the signing instance, and therefore allowing the adversary to obtain these values. This improves previous works, such us [14, 42] where the random values for the signature are supposed to be protected or absent (by using a deterministic signature).

- When instantiated with a 1-round 2-party protocol strongly secure in the standard model, for instance [5], our compiler produces a 2-round GKE which is strongly secure in the standard mode. This scheme is the first construction achieving such strong security guarantees in only two communication rounds.

2 Security Model

2.1 Description and strong security

Our security model is a modification of that of Bohli et al [6], which in turn builds in [4, 3, 33]. Furthemore, we treat

[1]Actually, as evidenced in [41], the scheme proposed in [42] cannot be proven secure otherwise. We remark here however that in [41] no augmented security notion for group key agreement is put forward along this line. In particular, the scheme of Zhao et al. [42] attacked in this paper is not proven to be secure if (as suggested by the authors) the underlying signature scheme resists randomness leakage.

[2]A preliminary version of this generic protocol can be found in the short abstract [23], where no security proof is provided and mutual authentication is not considered.

ephemeral reveals in a similar way as [42, 14].[3]

Protocol instances. Users are modeled as probabilistic polynomial time (ppt) Turing machines. Each user from a set $\mathcal{U}$ of possible participants may execute a polynomial number of protocol *instances* in parallel. To refer to instance s_i of a user $U_i \in \mathcal{U}$ we use the notation $\Pi_i^{s_i}$ $(i, s_i \in \mathbb{N})$. To each instance we assign seven variables, informally described next:

- $\mathsf{used}_i^{s_i}$ indicates whether this instance is or has been used for a protocol run;
- $\mathsf{state}_i^{s_i}$ keeps the internal state of the Turing machine that executes the protocol;
- $\mathsf{rand}_i^{s_i}$ keeps the session-specific atomic secret values —typically values generated uniformly at random— which will be referred to as *ephemeral keys*. More precisely, these are any values that cannot be computed from long-term secret keys or other values received/computed previously in the session;
- $\mathsf{term}_i^{s_i}$ shows if the execution has terminated;
- $\mathsf{sid}_i^{s_i}$ denotes a session identifier;
- $\mathsf{pid}_i^{s_i}$ 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 user accepted the session key;
- $\mathsf{sk}_i^{s_i}$ stores the session key once it is accepted by $\Pi_i^{s_i}$.

Communication network. Arbitrary point-to-point connections among the users are assumed to be available. The network is non-private, however, and fully asynchronous. More specifically, it is controlled by the adversary, who may delay, insert and delete messages at will.

Adversarial capabilities. We restrict to ppt adversaries. 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 $M \subseteq \mathcal{U}$, 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 $\mathsf{Execute}$ oracle is supposed to reflect a passive eavesdropping.
- $\mathsf{Reveal}(U_i, s_i)$ Yields the session key $\mathsf{sk}_i^{s_i}$ along with the session identifier $\mathsf{sid}_i^{s_i}$.
- $\mathsf{Test}(U_i, s_i)$ Provided that the session key is defined (i. e. $\mathsf{acc}_i^{s_i} = \mathtt{true}$ and $\mathsf{sk}_i^{s_i} \neq \perp$) and instance $\Pi_i^{s_i}$ is fresh (we define freshness later on), $\mathcal{A}$ 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 $b = 1$, where b is a hidden bit chosen at random prior to the first call. Namely, 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 (see the definition of partnering bellow) with $\Pi_i^{s_i}$.[4]
- $\mathsf{RevealRand}(U_i, s_i)$ This oracle returns the value stored in $\mathsf{rand}_i^{s_i}$.
- $\mathsf{Corrupt}(U_i)$ This oracle returns the long term key hold by U_i.

[3] Interpreting their $\mathsf{RevealEphemeralSecret}$ oracle as equivalent to our $\mathsf{RevealRand}$, which is not the only possible interpretation.

[4] This is the so-called Real or Random model, which can be proven equivalent to the usual model allowing for only one Test query with a loss of a factor m in the reduction, m being the number of involved protocol instances. See, for instance [1, 2].

Remark 2.1 *Following [25], we say that the* instance $\Pi_i^{s_i}$ *is* uncorrupted *if $\mathcal{A}$ has not made a call* $\mathsf{RevealRand}(U_i, s_i)$ *previously (this notion is relevant when introducing so-called* mutual authentication*, see bellow). On the other hand, we say user U_i is* honest *or* uncorrupted *if $\mathcal{A}$ has not made a call* $\mathsf{Corrupt}(U_i)$ *previously. Note that despite user U_i being corrupted, it can well be the case that an instance $\Pi_i^{s_i}$ remains uncorrupted.*

We aim at two basic goals for our protocol: correctness and strong security. A protocol is correct if all users involved in an execution in the presence of a passive adversary compute the same session key. Our notion of strong security ensures key privacy in the presence of an active adversary which is given access to all the oracles we have described. Before formally defining correctness and strong security, we introduce *partnering* and *freshness*, to express which instances are associated in a common protocol session and limit when the adversary is allowed to call the Test oracle.

Partnering. We refer to instances $\Pi_i^{s_i}, \Pi_j^{s_j}$ as being *partnered* if

$$\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}, \mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j} \text{ and } \mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}.$$

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

- $\mathsf{acc}_i^{s_i} = \mathsf{true}$;
- $\mathcal{A}$ never called $\mathsf{Reveal}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered;
- if $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ are partnered and $\mathcal{A}$ called $\mathsf{Corrupt}(U_j)$, then any message sent to $\Pi_i^{s_i}$ on behalf of $\Pi_j^{s_j}$ must indeed come from $\Pi_j^{s_j}$ intended to $\Pi_i^{s_i}$;
- $\mathcal{A}$ never called both $\mathsf{Corrupt}(U_j)$ and $\mathsf{RevealRand}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered.

Remark 2.2 *Note that each user is, in particular, partnered with itself in our definition. Therefore, if an instance $\Pi_i^{s_i}$ is fresh, then* $\mathsf{Reveal}(U_i, s_i)$ *cannot have been queried, neither both* $\mathsf{Corrupt}(U_i)$ *and* $\mathsf{RevealRand}(U_i, s_i)$*.*

Definition 2.3 (Correctness) *We call a group key establishment protocol $\mathcal{P}$* correct*, if in the presence of a passive adversary $\mathcal{A}$ —i. e., $\mathcal{A}$ must not use the* Send *oracle— the following holds: for all i, j with $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ and $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$, we have*

$$\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j} \neq null.$$

Definition 2.4 (Strong security) *Let $\mathcal{A}$ be an adversary making at most q_s and q_e queries to the* Send *and* Execute *oracles respectively. Let $k \in \mathbb{N}$ be the security parameter and denote by $\mathsf{Succ}_{\mathcal{A}}(k, q_s, q_e)$ the probability that $\mathcal{A}$ queries* Test *only on fresh instances and guesses correctly the bit b used by the* Test *oracle in a moment when all these instances are still fresh.*

*We say a group key establishment protocol is (q_s, q_e)-*strongly secure *if the advantage $\mathsf{Adv}_{\mathcal{A}-\mathtt{SGAKE}}(k, q_s, q_e)$ of any ppt adversary $\mathcal{A}$ in attacking the protocol is bounded by another function $\mathsf{Adv}_{\mathtt{SGAKE}}(k, q_s, q_e)$ which is negligible in k, where the aforementioned advantage is defined as*

$$\mathsf{Adv}_{\mathcal{A}-\mathtt{SGAKE}}(k, q_s, q_e) := \|2 \cdot \mathsf{Succ}_{\mathcal{A}}(k, q_s, q_e) - 1.\|$$

Note that our definition provides a strong notion of key privacy, including *perfect forward secrecy* [27, 18] and resistance to *key compromise impersonation (KCI) attacks* against key privacy [25]:

Perfect forward secrecy. An adversary getting the long-term key of a user should not gain any information on the session keys previously established by that user. Our definition captures perfect forward secrecy, since an adversary $\mathcal{A}$ is allowed to obtain the long-term keys of all users without violating freshness, provided he does not send any "relevant" messages after having received these long-term keys.

Key compromise impersonation resilience (against key privacy). An adversary is said to *impersonate* a user B to another user A if B is honest and the protocol instance at A accepts the session with B as one of his session peers, but there is no such partnered instance at B. An adversary $\mathcal{A}$ is considered successful in mounting a *key compromise impersonation* (**KCI**) attack knowing a user A's long-term private key if he manages to impersonate an honest party B to A. As pointed out in [25], when the goal of the adversary is to break the confidentiality of the session key, it only makes sense to consider an *outsider* adversary (see also [25] for precise definitions of *outsider* and *insider adversaries*). Our security definition takes this kind of attacks into account, since if $\Pi_i^{s_i}$ is the Test session then U_i may be corrupted (although the adversary cannot be active with respect to a partner $\Pi_j^{s_j}$ of $\Pi_i^{s_i}$ without violating freshness).

2.2 Further security properties

In addition to key privacy, several other security requirements such as *unknown key-share resilience*, *key confirmation*, *explicit key authentication* and *mutual authentication* are desirable for a group key exchange protocol. All of them are covered by the notion of *MA-security* [10], which was enhanced in [25] to deal with outsider and insider KCI attacks. Here we adopt the stronger one, MA-security with insider KCIR (key compromise impersonation resilience), yet slightly modifying the definition given in [25] as we consider $\mathsf{RevealRand}$ instead of $\mathsf{RevealState}$ queries.

Definition 2.5 (MA-security with insider KCIR) *Consider an adversary $\mathcal{A}$ against the MA-security of a correct GKE protocol, namely, $\mathcal{A}$ is allowed to query* Send, $\mathsf{Execute}$, $\mathsf{RevealRand}$, Reveal *and* $\mathsf{Corrupt}$ *oracles. Then $\mathcal{A}$ is said to violate the* MA property *if at some point, there exist an uncorrupted instance $\Pi_i^{s_i}$ (although U_i may be corrupted) that has accepted with $sk_i^{s_i}$ and another party $U_j \in \mathsf{pid}_i^{s_i}$ that is uncorrupted at the time $\Pi_i^{s_i}$ accepts such that*

- *there is no instance $\Pi_j^{s_j}$ with $(\mathsf{pid}_j^{s_j}, \mathsf{sid}_j^{s_j}) = (\mathsf{pid}_i^{s_i}, \mathsf{sid}_i^{s_i})$, or*
- *there is an instance with $(\mathsf{pid}_j^{s_j}, \mathsf{sid}_j^{s_j}) = (\mathsf{pid}_i^{s_i}, \mathsf{sid}_i^{s_i})$ that has accepted with $sk_j^{s_j} \neq sk_i^{s_i}$.*

Let us denote by $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{ma}}(k)$ the success probability of any ppt adversary $\mathcal{A}$ violating the MA property. Then, we say a group key establishment provides MA-security in the presence of insiders *if $\mathsf{Succ}_{\mathcal{A}}^{\mathsf{ma}}(k)$ is negligible in the security parameter k.*

3 Proposal of a secure protocol

3.1 Signatures withstanding randomness reveals

Our proposal of a secure protocol will make use of a signature scheme for authentication. As our security model allows the adversary to access the random coins involved in a protocol execution by means of the oracle $\mathsf{RevealRand}$, we assume that this oracle also outputs the randomness used for signing (if any). Note that this issue was mentioned in [14] but not considered in the proposed construction, as the authors suggest using a trusted device to protect this value or a deterministic signature scheme. As evidenced in [41], the security of this scheme is jeopardized if such precautions are not taken.

We are thus in need of stronger security guarantees, and therefore introduce a security notion for signature schemes, which we call *existential unforgeability under adaptive chosen message and randomness reveal attacks* (`EUF-CMRA`), capturing the property of remaining secure even if the randomness used when signing is leaked. This notion is identical to *unforgeability under adaptive chosen message attacks and ephemeral secret leakage attacks security* defined independently by Tseng et al in [41][5].

Before providing the definition, let us recap some terminology. A public key signature scheme $\mathcal{S}$ is explicited by three algorithms ($\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}$), where KeyGen, on input the security parameter, outputs a pair $(vk, sigk)$,

[5] At the writting of this paper we were not aware of this work, and have further decided to keep the name we had initially chosen for this notion.

the public verification key and the secret signing key respectively; Sign outputs a signature $\sigma = \mathsf{Sign}(sigk, m, sigr)$ where m is the signed message and $sigr$ is a random value chosen from an appropriate set every time Sign is invoked. Further, Verify is the (publicly available) verification algorithm. Note that we are considering that Sign takes the random coins $sigr$ as explicit input: this covers probabilistic and deterministic signature schemes; for the later we allow $sigr$ to be the empty string.

Now the standard security definition for signature schemes, i.e. existential unforgeability under adaptive chosen message attacks (EUF−CMA), is strengthened by giving the adversary access to a more powerful oracle, that also provides the randomness used when generating the signature. More formally, the adversary $\mathcal{A}$ will play the following game (EUF−CMRA, from *existential unforgeability under adaptive chosen message and randomness reveal attacks*):

1. $(vk, sigk)$ is generated with KeyGen and vk is provided to $\mathcal{A}$;
2. the adversary is given access to a signing oracle $\mathcal{O}_{sigk}(.)$ such that, every time a message m_j is queried, a random value $sigr_j$ is chosen as specified in the signing algorithm, a signature $\sigma_j = \mathsf{Sign}(sigk, m_j, sigr_j)$ is generated and $(\sigma_j, sigr_j) = \mathcal{O}_{sigk}(m_j)$ is returned to $\mathcal{A}$;
3. after adaptively querying the oracle, $\mathcal{A}$ outputs a pair (m, σ).

We say that $\mathcal{A}$ wins the EUF−CMRA game if m has not been queried to $\mathcal{O}_{sigk}(.)$ and σ is a valid signature for m. Let $\mathsf{Adv}_{\mathcal{A}-\texttt{EUF-CMRA}}(k, q)$ denote the probability that an adversary $\mathcal{A}$, making at most q calls to the signing oracle $\mathcal{O}_{sigk}(.)$, wins the EUF−CMRA game when the security parameter is k.

Definition 3.1 (EUF−CMRA security) *The signature scheme $\mathcal{S}$ is q-*existentially unforgeable under adaptive chosen message and randomness reveal attacks *(q − EUF−CMRA) if for every probabilistic polynomial time adversary $\mathcal{A}$ making at most q calls to the signing oracle, the function $\mathsf{Adv}_{\mathcal{A}-\texttt{EUF-CMRA}}(k, q)$ is bounded by another function $\mathsf{Adv}_{\texttt{EUF-CMRA}}(k, q)$ which is negligible in the security parameter k.*

Remark 3.2 *This security notion is trivially achieved if $\mathcal{S}$ is a* EUF−CMA *signature scheme which is either deterministic or such that the randomness is part of the signature.*

As pointed out in [40], in many existing signature schemes, the randomness is generated in the signing phase and provided to the verifier as part of the signature. For instance, signature schemes in [7, 12, 13, 15, 19, 22, 28, 29, 30, 37, 39, 43] fulfill this property and, therefore satisfy our EUF−CMRA security notion.

3.2 Collision resistant pseudorandom function families

In our construction we use a pseudorandom function (PRF) family $\mathcal{F} = \{\mathcal{F}^\ell\}_{\ell \in \mathbb{N}}$, $\mathcal{F}^\ell = \{F_\alpha\}_{\alpha \in \{0,1\}^\ell}$, which has the additional property of being *collision resistant* (see, for example, [32]). Next we recall the definition. Consider the game where an element v_ℓ in the domain of all functions in $\mathcal{F}^\ell$, is chosen according to a randomized sampling algorithm and given to an adversary $\mathcal{A}$.[6] The adversary is also given access to an oracle $F_{(.)}(v_\ell)$ and is said to win the game if it outputs a pair of indexes $\alpha \neq \alpha' \in \{0,1\}^\ell$ such that $F_\alpha(v_\ell) = F_{\alpha'}(v_\ell)$, without having queried both indexes to the oracle. We denote by $\mathsf{Adv}_{\mathcal{A}\text{-}\texttt{COLL-PRF}}(\ell, q)$ the probability that an adversary $\mathcal{A}$, making at most q queries to the oracle, wins the game.

Definition 3.3 *The PRF family $\mathcal{F} = \{\mathcal{F}^\ell\}_{\ell \in \mathbb{N}}$ is q-*collision resistant *if for every probabilistic polynomial time adversary $\mathcal{A}$ making at most q calls to the evaluation oracle, the function $\mathsf{Adv}_{\mathcal{A}\text{-}\texttt{COLL-PRF}}(\ell, q)$ is bounded by another function $\mathsf{Adv}_{\texttt{COLL-PRF}}(\ell, q)$ which is negligible in ℓ.*

In addition and for subsequent use in our security statements, we will denote by $\mathsf{Adv}_{\texttt{PRF}}(\ell, q)$ the function which upper bounds the advantage of any adversary trying to distinguish a function in $\mathcal{F}^\ell$ from a random one (making at

[6] For simplicity, we may assume all functions in $\mathcal{F}^\ell$ to have the same domain $\{0,1\}^{r_\ell}$ and range $\{0,1\}^{\kappa_\ell}$, with r_ℓ, κ_ℓ polynomial in ℓ.

most q queries to the function oracle). Note that, due to the PRF property of the family $\mathcal{F}$, $\mathsf{Adv}_{\mathrm{PRF}}(\ell, q)$ is negligible in the security parameter ℓ.

In our design, we will use a family of universal hash functions $\mathcal{UH} = \{\mathcal{UH}_\ell\}_{\ell\in\mathbb{N}}$, such that, for a given $\ell \in \mathbb{N}$ every function in $\mathcal{UH}_\ell$ maps bitstrings of a fixed size t_ℓ onto $\{0,1\}^\ell$. The family $\mathcal{UH}$ will be used to select an index within a collision-resistant pseudorandom function family $\mathcal{F} = \{\mathcal{F}^\ell\}_{\ell\in\mathbb{N}}$. In the sequel, both parameters ℓ and t_ℓ are assumed to be sufficiently large and polynomial in the security parameter k. Due to the universal property of the family $\mathcal{UH}$, the probability of any function UH $\in \mathcal{UH}_\ell$ producing the same output with two different inputs is at most $1/2^{t_\ell}$ (see, for instance, [38]).

3.3 From 2-Party to group keeping strong security

In this section we present a one-round compiler, which applied to a strongly secure 2-party key exchange 2-SAKE yields a strongly secure group key exchange, adding only one communication round. Our construction does not involve any idealized assumption, thus if the 2-SAKE is in the standard model, so will the resulting n-party protocol be.

Our design is detailed in Figure 1, where the **Set up** phase can be realized by means of a public key infraestructure (PKI) —and should thus be assumed to involve a trusted entity. At this, users are supposed to be somewhat organized in a cycle (a la Burmester-Desmedt, see [11]); thus, user indices i are to be taken modulo n. Note that we further assume that there might be *independent* authentication keys used for the 2-party and group setting, namely, the compiler will call for (freshly generated) signing keys for a dedicated signature scheme (which we will denote by $(vk_i, sigk_i)$) while we also explicit each user may have generated a pair of long-term keys $(2pk_i, 2sk_i)$ for 2-SAKE.[7]

Before moving on to the security statements let us specify how the RevealRand and Corrupt oracles work for the compiled scheme. A query $\mathsf{Corrupt}(U_j)$ is answered with the long-term secret key $(2sk_i, sigk_i)$ of U_i. A query $\mathsf{RevealRand}(U_i, s_i)$ returns $(\overrightarrow{r}_i, \overleftarrow{r}_i, r_i, sigr_i^0, sigr_i^1)$ where $\overrightarrow{r}_i, \overleftarrow{r}_i$ are the random coins used in the two executions of the 2-SAKE, r_i is the random nonce used in Round 1 of the compiler, and $sigr_i^j$, for $j = 0, 1$ are the nonces involved in the two signatures enforced by the compiler. Now it is easy to argue that RevealRand is not useful for the adversary, and thus the strong security of our scheme follows. Indeed, $\mathsf{RevealRand}(U_i)$ returns:

a) the randomness used by U_i in the 2-SAKE protocol, which is of no use for the adversary due to the strong security of 2-SAKE;

b) the signing nonces $sigr_i^0, sigr_i^1$, which will also be useless if the signature scheme is secure in the sense of EUF−CMRA;

c) the nonce r_i, which is anyway public, as it is broadcast in Round 1.

Theorem 3.4 *Assuming $\mathcal{S}$ is an* EUF−CMRA *signature scheme,* 2-SAKE *is strongly secure, $\mathcal{F}$ is a collision-resistant PRF and $\mathcal{UH}$ is a universal hash function family, the protocol from Figure 1 is correct and strongly secure.*
More precisely, let k and ℓ be as in the protocol specification from Figure 1. Then, for any probabilistic polynomial time adversary $\mathcal{A}$ making at most q_s calls to the Send *oracle and q_e calls to the* Execute *oracle,* $\mathsf{Adv}_{\mathcal{A}-\mathrm{SGAKE}}(k, q_s, q_e)$ *is upper bounded by*

$$\frac{(nq_e + q_s)^2}{2^{\ell+1}} + n\,\mathsf{Adv}_{\mathrm{EUF-CMRA}}(k, 2q_e + q_s) +$$
$$+\frac{1}{2^{t_\ell}}(nq_e + \frac{q_s}{2}) + \mathsf{Adv}_{\mathrm{COLL-PRF}}(\ell, 3nq_e + q_s) +$$
$$+2 \cdot \mathsf{Adv}_{\text{2-SAKE}}(k, q_s, nq_e) + \mathsf{Adv}_{\mathrm{PRF}}(\ell, 3nq_e + q_s),$$

where $\mathsf{Adv}_{\text{2-SAKE}}(\cdot,\cdot,\cdot)$ *is the advantage in regard to* 2-SAKE, *specified as in Definition 2.4 and t_ℓ is polynomial in k.*

[7] This statement is quite general; note that these might not even be signing keys (as it would happen if 2-SAKE is the NAXOS scheme).

Set up:

Fix $\ell \in \mathbb{N}$ polynomial in the security parameter k. Let $\mathcal{UH}$ be a family of universal hash functions ranging in $\{0,1\}^\ell$ and $\mathcal{F}$ be a collision-resistant pseudorandom function family.
A function $\mathrm{UH} : \{0,1\}^{t_\ell} \mapsto \{0,1\}^\ell$ from $\mathcal{UH}_\ell$ and a description of $\mathcal{F}^\ell$ are made public together with a value v in the domain of all functions from $\mathcal{F}^\ell$.[a]

We assume all users to know a priori their partners and have set the variable pid accordingly.

Furthermore, a pair of keys $(vk_i, sigk_i)$ for the signature scheme $\mathcal{S}$ is generated for each U_i, which gets the secret key $sigk_i$ while vk_i is publicized.

Round 0:

Usage of 2-SAKE.

- For $i = 1, \ldots, n$ execute 2-SAKE(U_i, U_{i+1});
  after that each user U_i holds two keys $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ shared with U_{i+1} and U_{i-1} respectively.
- Additionally, in the last round of the 2-SAKE, each U_i
  - chooses a random nonce $r_i \in_R \{0,1\}^\ell$;
  - computes a signature σ_i^0 of (U_i, r_i);
  - broadcasts $M_i^0 := (U_i, r_i, \sigma_i^0)$.

Round 1:

Computation. Each U_i :

- Checks the signatures σ_j^0; if something fails, aborts;
- Sets $\mathsf{sid}_i := \mathsf{pid}_i | r_1 | \ldots | r_n$;
- Computes $X_i := \overrightarrow{K}_i \oplus \overleftarrow{K}_i$;
- Computes the confirmation strings $\overleftarrow{\rho}_i = F_{\mathrm{UH}(U_{i-1}, U_i, \overleftarrow{K}_i, \mathsf{pid}_i)}(v)$ and $\overrightarrow{\rho}_i = F_{\mathrm{UH}(U_i, U_{i+1} \overrightarrow{K}_i, \mathsf{pid}_i)}(v)$;
- Computes a signature σ_i^1 of $(U_i, \mathsf{sid}_i, X_i, \overleftarrow{\rho}_i, \overrightarrow{\rho}_i)$.

Broadcast. Each U_i broadcasts $M_i^1 := (U_i, \mathsf{sid}_i, X_i, \overleftarrow{\rho}_i, \overrightarrow{\rho}_i, \sigma_i^1)$.

Key Computation.

Check. Each U_i checks all the signatures, equality of pid's, sid's, consistency of $\overleftarrow{\rho}_i$ and $\overrightarrow{\rho}_i$; if something fails, aborts.

Computation. Each U_i

- for $j = 1, \ldots, n$, computes $\overleftarrow{K}_j$ and sets $K_j := \overleftarrow{K}_j$;
- sets $K := (K_1, \ldots, K_n, \mathsf{sid}_i)$;
- accepts $\mathsf{sk}_i := F_{\mathrm{UH}(K)}(v)$.

[a] We assume t_ℓ to be sufficiently large, and the input values to UH to be padded consistently.

Figure 1: A Compiler for achieving group AKE with strong security

PROOF. Checking the correctness of the protocol is straightforward: if all the participants follow the protocol description and there is no active adversarial intervention, then all checks will succeed and every participant will set the same pid and sid, obtain the same $\{K_j\}_{j=1}^n$ from the broadcast $\{X_j\}_{j=1}^n$ and consequently compute the same session key.

The proof for the strong security is conducted through a sequence of games. Following standard notation, we denote by $\mathsf{Adv}(\mathcal{A}, G_i)$ the advantage of the adversary when confronted with Game i. The security parameter is denoted by k. Further, in the sequel we let q_e and q_s denote the maximum number of calls made by the adversary to the $\mathsf{Execute}$ and Send oracles (resp.).

Game 0. All the oracles are simulated as in the real protocol; thus, $\mathsf{Adv}(\mathcal{A}, G_0)$ is exactly $\mathsf{Adv}_{\mathcal{A}-\mathtt{SGAKE}}(k, q_s, q_e)$ as in the definition of strong security from Section 2.

Game 1. This game is identical to Game 0, except that the execution is aborted if the event Repeat occurs. This is defined to happen if an uncorrupted participant chooses in Round 0 a nonce r_i that was previously used by an oracle of some principal.

As q_e and q_s denote the maximum number of calls to the $\mathsf{Execute}$ and Send oracles respectively, the number of nonces generated by honest users during the game is at most $nq_e + q_s$. Therefore the probability of Repeat is upper bounded by the probability of collision when choosing $nq_e + q_s$ values among 2^ℓ possible ones, which is in turn upper bounded by $(nq_e + q_s)^2/(2 \cdot 2^\ell)$ (see, for example, appendix A.4 in [31]). As a result,

$$|\mathsf{Adv}(\mathcal{A}, G_0) - \mathsf{Adv}(\mathcal{A}, G_1)| \quad \leq \quad P(\mathsf{Repeat}) \leq \frac{(nq_e + q_s)^2}{2^{\ell+1}}.$$

Game 2. This is identical to Game 1, except that now the execution is aborted if the adversary succeeds in forging an authenticated message $M\|\sigma$ for participant U_i without having queried $\mathsf{Corrupt}(U_i)$ and where M was not output by any of U_i's instances. Let us call this event Forge.

Indeed, an adversary $\mathcal{A}$ that can reach Forge can be used for forging a signature in a $\mathtt{EUF-CMRA}$ game: the given public key is assigned randomly to one of the n users in the group and all other parties are initialized following the protocol specification; then all the queries in the *strong security* game are answered faithfully and whenever a signature for a message of the selected user is needed, the signing oracle of the $\mathtt{EUF-CMRA}$ game is queried to produce the signature. Note that the number of such queries is upper bounded by $2q_e + q_s$.
The probability of the adversary choosing the "right" user U_i when assigning the public key for the signature equals $1/n$, therefore we have $\mathsf{Adv}_{\mathtt{EUF-CMRA}}(k, 2q_e + q_s) \geq \frac{1}{n}\, P(\mathsf{Forge})$ which yields

$$|\mathsf{Adv}(\mathcal{A}, G_1) - \mathsf{Adv}(\mathcal{A}, G_2)| \leq P(\mathsf{Forge})$$

$$P(\mathsf{Forge}) \leq n\ \mathsf{Adv}_{\mathtt{EUF-CMRA}}(k, 2q_e + q_s).$$

Game 3. In this game, we impose that a fresh instance $\Pi_i^{t_i}$ does not accept in Round 1 whenever it receives a message M_j^1 not generated by the respective instance $\Pi_j^{t_j}$, $j \neq i$ in the same session. At this, we take two instances $\Pi_{\alpha_0}^{t_{\alpha_0}}$, $\Pi_{\alpha_r}^{t_{\alpha_r}}$ for being in the *same session*, if there is a sequence of instances $(\Pi_{\alpha_\mu}^{t_{\alpha_\mu}})_{0\leq\mu\leq r}$ such that for each $\mu = 0, \ldots, r-1$ the instances $\Pi_{\alpha_\mu}^{t_{\alpha_\mu}}$ and $\Pi_{\alpha_{\mu+1}}^{t_{\alpha_{\mu+1}}}$ have jointly executed 2-SAKE, hold two nonces r_{α_μ} and $r_{\alpha_{\mu+1}}$ linked to this execution[8] and, furthermore, they all hold the same pid (namely, $\mathsf{pid}_{\alpha_0} = \cdots = \mathsf{pid}_{\alpha_r}$). The adversary $\mathcal{A}$ can detect the difference to Game 2 if $\mathcal{A}$ replayed or fabricated a message that should have led to acceptance in Round 1 in that game. Since all messages broadcasted in Round 1 must contain the signed nonce r_i (as part of the signed sid_i) and we excluded already the events Forge and Repeat, games 2 and 3 are identical for $\mathcal{A}$. As a result

$$\mathsf{Adv}(\mathcal{A}, G_3) = \mathsf{Adv}(\mathcal{A}, G_2).$$

[8] Implicitly, the pair of nonces (r_i, r_j) complete the role of a session identifier for the corresponding 2-SAKE execution.

Game 4. In this game, we impose that a fresh instance $\Pi_i^{t_i}$ does not accept the session key in Round 1 whenever two instances $\Pi_j^{t_j}$ and $\Pi_{j+1}^{t_{j+1}}$ in the same session (as above) which have jointly executed 2-SAKE and hold matching nonces r_j and r_{j+1} linked to this execution hold however non-matching two party keys[9]. Let us denote this event by Coll.
Due to the modifications made in Game 3, in a fresh session every message must have been generated according to the specification of the protocol. Therefore the event Coll happens only if there are two instances $\Pi_j^{t_j}$ and $\Pi_{j+1}^{t_{j+1}}$ such that $\overrightarrow{K}_j \neq \overleftarrow{K}_{j+1}$ but $\overrightarrow{\rho}_j = \overleftarrow{\rho}_{j+1}$ (as otherwise the involved instances would not accept).
Let UH be the function chosen at the beginning of the protocol and denote by $\overrightarrow{\alpha}_i = \mathrm{UH}(U_i, U_{i+1}, \overrightarrow{K}_i, \mathsf{pid}_i)$ and $\overleftarrow{\alpha}_i = \mathrm{UH}(U_{i-1}, U_i, \overleftarrow{K}_i, \mathsf{pid}_i)$. Taking into account how $\overrightarrow{\rho}_i$, $\overleftarrow{\rho}_i$, $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ are defined, it is clear that the event Coll happens only if one of the two following events happen:

- Coll1, which is the event that during the security game there exist instances $\Pi_j^{t_j}$ and $\Pi_{j+1}^{t_{j+1}}$ such that $\overrightarrow{K}_j \neq \overleftarrow{K}_{j+1}$ but $\overrightarrow{\alpha}_j = \overleftarrow{\alpha}_{j+1}$.
- Coll2, which is the event that during the security game there exists instances $\Pi_j^{t_j}$ and $\Pi_{j+1}^{t_{j+1}}$ such that $\overrightarrow{K}_j \neq \overleftarrow{K}_{j+1}$, $\overrightarrow{\alpha}_j \neq \overleftarrow{\alpha}_{j+1}$ but $\overrightarrow{\rho}_j = \overleftarrow{\rho}_{j+1}$.

Because of the universal property of the family $\mathcal{UH}$, the probability of UH producing the same output with two different inputs is at most $1/2^{t_\ell}$.

In addition the number of possible pairs of nonces generated during the security game is upper bounded by $nq_e + \frac{q_s}{2}$. As a result,

$$P(\mathsf{Coll1}) \leq \frac{1}{2^{t_\ell}}(nq_e + \frac{q_s}{2}).$$

On the other hand, an adversary $\mathcal{A}$ which produces the event Coll2 can be used to construct an adversary against the collision resistance of the pseudo-random family $\mathcal{F}$. The reason is that, in case Coll2 happens, then two different indexes $\overrightarrow{\alpha}_j \neq \overleftarrow{\alpha}_{j+1}$ have been found such that $F_{\overrightarrow{\alpha}_j}(v) = F_{\overleftarrow{\alpha}_{j+1}}(v)$. As the function $F_{(.)}(v)$ is invoked at most $3nq_e + q_s$ times during the game, it holds that

$$P(\mathsf{Coll2}) \leq \mathsf{Adv}_{\mathrm{COLL\text{-}PRF}}(\ell, 3nq_e + q_s).$$

Putting everything together we have

$$|\mathsf{Adv}(\mathcal{A}, G_3) - \mathsf{Adv}(\mathcal{A}, G_4)|$$

is bounded by

$$P(\mathsf{Coll}) \leq P(\mathsf{Coll1}) + P(\mathsf{Coll2})$$

and thus by

$$\frac{1}{2^{t_\ell}}(nq_e + \frac{q_s}{2}) + \mathsf{Adv}_{\mathrm{COLL\text{-}PRF}}(\ell, 3nq_e + q_s).$$

Game 5. The simulation of the Send and Execute oracles is modified in the following way. For every $i = 1, \ldots, n$, whenever an instance $\Pi_i^{t_i}$ is still considered fresh at the end of Round 0, and the two party keys $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ are defined, they are replaced with random values chosen from the appropriate set. This replacement is done consistently, in the sense that, if $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ coincide with $\overleftarrow{K}_{i+1}$ and $\overrightarrow{K}_{i-1}$ respectively, they are replaced with matching random values.

[9] As 2-SAKE is only assumed to have implicit key confirmation, it is not excluded that two users enrolled in an execution end up with different – thus useless – keys

In order to bound the distance between G_3 and G_4 we will build, from an adversary $\mathcal{A}$ which is able to distinguish between these two games, another adversary $\mathcal{B}$ attacking the underlying 2-SAKE protocol such that

$$|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_5)| = 2 \cdot \mathsf{Adv}_{\mathcal{B}-\text{2-SAKE}}(k, q_s^{\mathcal{B}}, q_e^{\mathcal{B}}),$$

where $\mathsf{Adv}_{\mathcal{B}-\text{2-SAKE}}(k, q_s^{\mathcal{B}}, q_e^{\mathcal{B}})$ denotes the advantage of a probabilistic polynomial time adversary $\mathcal{B}$ attacking 2-SAKE and making at most $q_s^{\mathcal{B}}$ calls to the Send oracle and $q_e^{\mathcal{B}}$ calls to the Execute oracle.

To prove this bound, assume that $\mathcal{B}$, which runs $\mathcal{A}$ as an auxiliary algorithm, is given access to a simulation of 2-SAKE. Further, $\mathcal{B}$ executes the key generation algorithm of $\mathcal{S}$ for each user U_i, thus retrieving a pair of corresponding signing keys $(vk_i, sigk_i)$.

Now, whenever an instance $\Pi_i^{s_i}$ is used by $\mathcal{A}$ as the input of a query, $\mathcal{B}$ associates it with two different independent instances $\Pi_i^{2s_i-1}$ and $\Pi_i^{2s_i}$ of the same user in the 2-SAKE protocol.

Also a list $\mathcal{L}$, storing the returned random nonces r_i, is needed to answer the queries of $\mathcal{A}$. More precisely, the first time a random nonce r_i is required to answer a $\mathsf{RevealRand}_{\mathcal{A}}$, $\mathsf{Execute}_{\mathcal{A}}$ or $\mathsf{Send}_{\mathcal{A}}$ query involving instance Π_i^s, a random value r_i is chosen u.a.r. from the appropriate set and (U_i, s, r_i) is stored in $\mathcal{L}$. Whenever this value is needed again to answer a query, it is extracted from $\mathcal{L}$. Similarly, $\mathcal{B}$ maintains a list for the signing nonces $Sig\mathcal{L}$, where he stores appropiately generated randomness involved in any of the signatures that might be involved in the simulation.

Now let us describe how the answers to the queries of $\mathcal{A}$ are constructed:

- Whenever a query $\mathsf{Corrupt}_{\mathcal{A}}(U_i)$ is made, $\mathcal{B}$ queries $\mathsf{Corrupt}_{\mathcal{B}}(U_i)$ to retrieve $2sk_i$ and provides $(2sk_i, sigk_i)$ as answer to $\mathcal{A}$.

- To answer a query $\mathsf{RevealRand}_{\mathcal{A}}(U_i, s_i)$, $\mathcal{B}$ executes $\mathsf{RevealRand}_{\mathcal{B}}$ with the two associated instances $\Pi_i^{2s_i-1}$ and $\Pi_i^{2s_i}$, obtaining $\overrightarrow{r}_i$ and $\overleftarrow{r}_i$. Then chooses r_i u.a.r. from the appropriate set (or extracts it from $\mathcal{L}$ if needed) and similarly generates signing nonces for $\mathcal{S}$, r_i^1 and r_i^2 or retrieves them from the $Sig\mathcal{L}$ list.

- To answer an $\mathsf{Execute}_{\mathcal{A}}$ query, $\mathcal{B}$ queries $\mathsf{Execute}_{\mathcal{B}}$ with the corresponding pairs of instances to construct a transcript for Round 0. Then makes $\mathsf{Test}_{\mathcal{B}}$ queries to obtain the keys $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ for every user and constructs the rest of the transcript for Round 1 and 2 as it would be done in a real execution of the protocol, taking random values from $\mathcal{L}$ and $Sig\mathcal{L}$ as needed.

- To answer a $\mathsf{Send}_{\mathcal{A}}$ query for Round 0, a query $\mathsf{Send}_{\mathcal{B}}$ is executed by $\mathcal{B}$ with the associated instances and the same responses are returned. If the $\mathsf{Send}_{\mathcal{A}}$ query is for Round 1, first $\mathcal{B}$ sets the values of $\overrightarrow{K}_i$ and $\overleftarrow{K}_i$ by querying one of the oracles $\mathsf{Test}_{\mathcal{B}}$ or $\mathsf{Reveal}_{\mathcal{B}}$ (depending on whether the involved instance is fresh according to our definition, and thus allows to query $\mathsf{Test}_{\mathcal{B}}$, or not). The rest of the answer is generated as in the description of the $\mathsf{Execute}_{\mathcal{A}}$ answer.

- A $\mathsf{Reveal}_{\mathcal{A}}$ query is answered in a similar way as a $\mathsf{Send}_{\mathcal{A}}$ or $\mathsf{Execute}_{\mathcal{A}}$ query.

- Finally, to answer allowed $\mathsf{Test}_{\mathcal{A}}$ queries, a bit $b' \in \{0, 1\}$ is chosen by $\mathcal{B}$ at the beginning of the simulation. If $b' = 1$ a random group key is returned while if $b' = 0$ an actual key, constructed consistently with the rest of the simulation, is returned.

At some point $\mathcal{A}$ will output a bit b'' as a guess for b' which will determine the output b of $\mathcal{B}$ for the 2-SAKE challenge. Namely, $\mathcal{B}$ outputs $b = 0$ if and only if $b' = b''$. Taking into account that the view of $\mathcal{A}$ is identical to G_4 if the answers of $\mathsf{Test}_{\mathcal{B}}$ are real keys and to G_5 if the answers of $\mathsf{Test}_{\mathcal{B}}$ are random ones, and counting $q_s^{\mathcal{B}} \leq q_s$ and $q_e^{\mathcal{B}} \leq nq_e$, we have that

$$|\mathsf{Adv}(\mathcal{A}, G_4) - \mathsf{Adv}(\mathcal{A}, G_5)|$$

is bounded by

$$2 \cdot \mathsf{Adv}_{\text{2-SAKE}}(k, q_s, nq_e).$$

Game 6. In this game, for every $i = 1, \ldots, n$, the value $sk_i := F_{\mathrm{UH}(K)}(v)$ is replaced with a random value chosen from $\{0,1\}^{\kappa_\ell}$. As by now $K := (K_1, \ldots, K_n, \mathsf{sid}_i)$, and all the K_i have been chosen u.a.r. in this game, the output of $F_{\mathrm{UH}(K)}$ is, due to the pseudorandomness property of $\mathcal{F}$, distinguishable from a random value only with negligible probability in ℓ (which is polynomial in k). More precisely we have,

$$|\mathsf{Adv}(\mathcal{A}, G_6) - \mathsf{Adv}(\mathcal{A}, G_5)| \leq \mathsf{Adv}_{\mathrm{PRF}}(\ell, 3nq_e + q_s).$$

In addition, the advantage of the adversary in G_6 equals 0, as the secret keys are chosen u.a.r. in $\{0,1\}^{\kappa_\ell}$. This concludes the proof.

□

3.4 MA-security

In this section we show our protocol satisfies MA-security with insider KCIR. For the shake of simplicity, we have chosen to formulate the security statement only in terms of the security parameter and not use the number of oracle calls. Further, we provide only a proof sketch, as a detailed one would repeat many of the arguments already specified above in the proof of Theorem 3.4.

Theorem 3.5 *Assuming $\mathcal{S}$ is an* `EUF-CMRA` *signature scheme, the protocol from Figure 1 satisfies MA-security with insider KCIR.*

PROOF'S SKETCH. This result may be obtained by "game hopping", letting the adversary $\mathcal{A}_{ma}$ interact with a simulator:

Game 0. In this game, the simulator faithfully simulates all protocol participants' instances for the adversary $\mathcal{A}_{\mathsf{ma}}$, i. e., the adversary's situation is the same as in the real model:

$$\mathsf{Adv}_{\mathcal{A}_{\mathsf{ma}}}^{\mathrm{Game}\ 0}(k) = \mathsf{Adv}_{\mathcal{A}_{\mathsf{ma}}}(k) = \mathsf{Succ}_{\mathcal{A}}^{\mathsf{ma}}(k).$$

Game 1. This game is aborted if the events Forge or Repeat, as described in the previous proof, occur. Otherwise, the game is identical to Game 0 and the adversary cannot detect the difference. The distance between the success probabilities of G_0 and G_1 is bounded by $P(\mathsf{Forge}) + P(\mathsf{Repeat})$, which is negligible in the security parameter k.

Let $\Pi_i^{s_i}$ be an uncorrupted instance that has accepted. Notice that once these events have been eliminated, all the honest parties in $\mathsf{pid}_i^{s_i}$ compute the same key:

Let $U_j \in \mathsf{pid}_i^{s_i}$ a user that is not corrupted at the time $\Pi_i^{s_i}$ accepts. If the events Forge and Repeat do not occur, since $\Pi_i^{s_i}$ has checked succesfully the equality of the session and partner identifiers, there is an instance $\Pi_j^{s_j}$ such that $(\mathsf{sid}_i^{s_i}, \mathsf{pid}_i^{s_i}) = (\mathsf{sid}_j^{s_j}, \mathsf{pid}_j^{s_j})$. Therefore, for the adversary to win the MA game, the session keys accepted should be different. However, if the instance accepted, in particular U_j successfully checked the confirmation strings $\overleftarrow{\rho}_{j+1}$ and $\overrightarrow{\rho}_{j-1}$ for his left and right two-party keys respectively. This means K_j and K_{j+1} have been computed correctly, except with negligible probability, since F is chosen from $\mathcal{F}$, a collision-resistant pseudorandom function family. Then, from the values X_l sent by the other users in $\mathsf{pid}_i^{s_i}$, and the rest of the confirmation strings, one can be sure, with overwhelming probability, that all K_l computed by U_i and U_j are equal. Therefore, both users compute the same session key.

Putting the probabilities together we recognize the adversary's advantage in the real model as negligible.

□

Scheme	Ref	Rounds	Model	Assumption
HMQV	[34]	2	ROM	GDH + KEA1
NAXOS	[35]	1	ROM	GDH (o PDH)
Cremers-Felz	[17]	1	ROM	Gap-CDH
Fujioka et al.	[21]	2	Std.	Ring-LWE, DBDH, DDH
Bergsma et al.	[5]	1	Std.	Factoring

Table 1: Examples of recent 2-SAKE protocols

Scheme	Ref	Assumption
Boneh-Boyen	[7]	Strong DH for bilinear groups
Camenisch-Lysyanskaya	[12]	Strong RSA
Fischlin	[19]	Strong RSA
Hofheinz-Kiltz	[28]	Strong RSA / q-Strong DH for bilinear groups
Hohenberger-Waters	[30]	RSA

Table 2: Examples of `EUF-CMRA` signature schemes

4 Concrete Implementations

In this section we propose several options to instantiate our compiler. Some possible choices for the two-party 2-SAKE scheme and the `EUF-CMRA` signature scheme are enumerated in Tables 1 and 2.

It is worth pointing out that every signature scheme in Table 2 is secure in the standard model. In addition, [7, 12, 19] provide strong `EUF-CMA`; however this property is not needed for the security of our proposal, thus [28, 30] are also suitable choices.

When a one-round 2-SAKE secure in the standard model, such as [5], is combined with any signature scheme in Table 2, the result of applying our compiler is a two-round group key exchange protocol which is strongly secure and MA-secure in our model. To the best of our knowledge, this is the first two-round group key establishment protocol with security proofs in a model considering randomness leakage in the attacked session, which does not make use of random oracles. In addition, if the choice for the signature scheme is [30], the security of the compiled two-round protocol depends only on the well-known RSA assumption.

M.I. González Vasco and Angel L. Pérez del Pozo are partially supported by research project MTM2013-41426-R, and A. Suárez Corona is supported by MTM2013-45588-C3-1-P, both funded by the Spanish MINECO.

References

[1] Michel Abdalla, Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. (Password) authenticated key establishment: From 2-party to group. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2007.

[2] 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, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2005.

[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] Florian Bergsma, Tibor Jager, and Jörg Schwenk. One-round key exchange with strong security: An efficient and generic construction in the standard model. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 477–494. Springer, 2015.

[6] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. Secure group key establishment revisited. *Int. J. Inf. Sec.*, 6(4):243–254, 2007.

[7] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[8] Timo Brecher, Emmanuel Bresson, and Mark Manulis. Fully robust tree-diffie-hellman group key exchange. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings*, volume 5888 of *Lecture Notes in Computer Science*, pages 478–497. Springer, 2009.

[9] Emmanuel Bresson and Mark Manulis. Securing group key exchange against strong corruptions. In Masayuki Abe and Virgil D. Gligor, editors, *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 249–260. ACM, 2008.

[10] Emmanuel Bresson, Mark Manulis, and Jörg Schwenk. On security models and compilers for group key exchange protocols. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *Advances in Information and Computer Security, Second International Workshop on Security, IWSEC 2007, Nara, Japan, October 29-31, 2007, Proceedings*, volume 4752 of *Lecture Notes in Computer Science*, pages 292–307. Springer, 2007.

[11] 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.

[12] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002.

[13] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[14] Cheng Chen, Yanfei Guo, and Rui Zhang. Group key exchange resilient to leakage of ephemeral secret keys with strong contributiveness. In Sabrina De Capitani di Vimercati and Chris Mitchell, editors, *Public Key Infrastructures, Services and Applications - 9th European Workshop, EuroPKI 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers*, volume 7868 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2012.

[15] Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

[16] Cas J. F. Cremers. Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS authenticated key exchange protocol. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, volume 5536 of *Lecture Notes in Computer Science*, pages 20–33, 2009.

[17] Cas J. F. Cremers and Michele Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. *IACR Cryptology ePrint Archive*, 2011:300, 2011.

[18] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[19] Marc Fischlin. The cramer-shoup strong-rsa signature scheme revisited. *IACR Cryptology ePrint Archive*, 2002:17, 2002.

[20] Atsushi Fujioka, Mark Manulis, Koutarou Suzuki, and Berkant Ustaoglu. Sufficient condition for ephemeral key-leakage resilient tripartite key exchange. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings*, volume 7372 of *Lecture Notes in Computer Science*, pages 15–28. Springer, 2012.

[21] Atsushi Fujioka, Koutarou Suzuki, Keita Xagawa, and Kazuki Yoneyama. Strongly secure authenticated key exchange from factoring, codes, and lattices. *Des. Codes Cryptography*, 76(3):469–504, 2015.

[22] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure hash-and-sign signatures without the random oracle. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 123–139. Springer, 1999.

[23] Maria Isabel González-Vasco, Ángel L. Pérez del Pozo, and Adriana Suárez-Corona. Thwarting randomness reveals in group key agreement. In Jesús Vigo-Aguiar, editor, *Proceedings of the 16th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2016*, volume 2, pages 606–614, 2016.

[24] M. Choudary Gorantla, Colin Boyd, Juan Manuel González Nieto, and Mark Manulis. Generic one round group key exchange in the standard model. In Dong Hoon Lee and Seokhie Hong, editors, *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, volume 5984 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2009.

[25] M. Choudary Gorantla, Colin Boyd, Juan Manuel González Nieto, and Mark Manulis. Modeling key compromise impersonation attacks on group key exchange protocols. *ACM Trans. Inf. Syst. Secur.*, 14(4):28, 2011.

[26] M.Choudary Gorantla, Colin Boyd, and JuanManuel Gonzlez Nieto. Modeling key compromise impersonation attacks on group key exchange protocols. In Stanisaw Jarecki and Gene Tsudik, editors, *Public Key Cryptography PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 105–123. Springer Berlin Heidelberg, 2009.

[27] Christoph G. Günther. An identity-based key-exchange protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer, 1989.

[28] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. *J. Cryptology*, 25(3):484–527, 2012.

[29] Susan Hohenberger and Brent Waters. Realizing hash-and-sign signatures under standard assumptions. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 333–350. Springer, 2009.

[30] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 654–670. Springer, 2009.

[31] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[32] Jonathan Katz and Ji Sun Shin. Modeling insider attacks on group key-exchange protocols. In Vijay Atluri, Catherine A. Meadows, and Ari Juels, editors, *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 180–189. ACM, 2005.

[33] Jonathan Katz and Moti Yung. Scalable Protocols for Authenticated Group Key Exchange. In Dan Boneh, editor, *Advances in Cryptology — CRYPTO'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.

[34] Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[35] Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2007.

[36] Mark Manulis, Koutarou Suzuki, and Berkant Ustaoglu. Modeling leakage of ephemeral secrets in tripartite/group key exchange. *IEICE Transactions*, 96-A(1):101–110, 2013.

[37] David Naccache, David Pointcheval, and Jacques Stern. Twin signatures: an alternative to the hash-and-sign paradigm. In Michael K. Reiter and Pierangela Samarati, editors, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001.*, pages 20–27. ACM, 2001.

[38] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 33–43. ACM, 1989.

[39] Sven Schäge. Twin signature schemes, revisited. In Josef Pieprzyk and Fangguo Zhang, editors, *Provable Security, Third International Conference, ProvSec 2009, Guangzhou, China, November 11-13, 2009. Proceedings*, volume 5848 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2009.

[40] Sven Schäge. Strong security from probabilistic signature schemes. In Marc Fischlin, Johannes A. Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*, pages 84–101. Springer, 2012.

[41] Yuh-Min Tseng, Tung-Tso Tsai, and Sen-Shan Huang. Enhancement on strongly secure group key agreement. *Security and Communication Networks*, 8(2):126–135, 2015. SCN-13-0353.R1.

[42] Jianjie Zhao, Dawu Gu, and M. Choudary Gorantla. Stronger security model of group key agreement. In Bruce S. N. Cheung, Lucas Chi Kwong Hui, Ravi S. Sandhu, and Duncan S. Wong, editors, *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, pages 435–440. ACM, 2011.

[43] Hua-Fei Zhu. New digital signature scheme attaining immunity to adaptive chosen message attack. *Chinese Journal of Electronics*, 10(4):484–486, 2001.