

Secure Group Key Establishment Revisited

Jens-Matthias Bohli,
Institut für Algorithmen und Kognitive Systeme,
Universität Karlsruhe (TH),
76128 Karlsruhe, Germany
bohli@ira.uka.de

María Isabel González Vasco
Departamento de Matemática Aplicada,
Universidad Rey Juan Carlos,
c/ Tulipán, s/n, 28933 Madrid, Spain
mariaisabel.vasco@urjc.es

Rainer Steinwandt
Department of Mathematical Sciences,
Florida Atlantic University,
777 Glades Road, Boca Raton, FL 33431, USA
rsteinwa@fau.edu

Abstract

We examine the popular proof models for group key establishment of Bresson et al. [9, 8] and point out missing security properties addressing malicious protocol participants. We show that established group key establishment schemes from CRYPTO 2003 and ASIACRYPT 2004 do not fully meet these new requirements. Next to giving a formal definition of these extended security properties, we prove a variant of the explored proposal from ASIACRYPT 2004 secure in this stricter sense. Our proof builds on the Computational Diffie Hellman (CDH) assumption and the random oracle model.

1 Introduction

Group key establishments allow $n \geq 2$ principals to agree upon a common secret key. It turns out that the design of such schemes faces some qualitatively new challenges that in this form do not arise in the two-party case. An excellent introduction and survey of the subject is given by Boyd and Mathuria [6]. To allow a rigorous security analysis, a framework for modelling group key establishments has been developed [8, 9, 20] going back on [3, 4, 1, 12, 24, 2]. A recent overview of those indistinguishability-based models is given in [14].

An issue that compared to the two-party case becomes much more relevant is the protocol behavior in the presence of malicious participants: For groups with $n \gg 2$ participants, assuming

all participants to strictly follow the protocol specification can be a rather strong assumption. Thus the question of security guarantees in the presence of malicious insiders naturally arises. Unfortunately, so far for the quite popular security models [8] and [9] extensions to malicious participants have hardly been explored and analyzed. In this type of model, security analysis typically restricts to the case of honest participants (see, e. g., [7, 20, 15]).

Our contribution. We examine two group key establishment protocols put forward in [20, 21] and point out insider attacks that are not covered by the model [9] in which they are proven secure. We give a formal definition of some security notions motivated by these attacks. We put forward a notion of session integrity that can be seen as a correctness guarantee in the presence of malicious insiders. Further on, we suggest a way to formalize an agreement property in the style of [9]. In a sense, these new notions round off the model [9] and cover most known attacks that can be carried out in the presence of malicious insiders. In the model we consider, a long term key is associated with a principal U_i , i. e., it is identical for all protocol instances $\Pi_i^{s_i}$ run by U_i . Once U_i 's long term key gets compromised through a corruption, we consider U_i as dishonest and do not try to establish security guarantees for individual protocol instances of U_i . In particular, we do not discuss key-compromise impersonation attacks where knowledge of a principal U_i 's long term secret is exploited to impersonate principals $U_j \neq U_i$ towards U_i .

To give a flavor of how to design secure protocols in the new strict sense that we suggest, in Section 4.1 a “hidden” security feature in a proposal from [20] is proven. Finally, in Section 4.2 we present a modification of [21] and prove it secure in our model—using the Computational Diffie Hellman (CDH) assumption as well as the random oracle model.

Related work. While we are not aware of a formal treatment of malicious insiders in the frameworks of [8, 9], the issue of malicious insiders in group key establishment protocols has been addressed by several authors already. In particular, Saeednia and Safavi-Naini put forward several security classes for group key establishments [23], and their class D.2 imposes that it *is infeasible for any coalition of malicious insiders to break the authenticity of the conference key without no insider detecting the fraud*. Also Cheng et al.'s list of attacks in [13] mentions insider attacks, and the work of Tzeng [26], for instance, shows that it is feasible to derive protocols with well-specified security guarantees even if a subset of the protocol participants acts maliciously.

Frameworks guaranteeing universal composability provide another approach to model key establishment protocols [25] where insider attacks or failures are already considered [11, 18]. Katz and Shin [18] in particular point out that their *definition of insider impersonation attacks is stronger than the numerous varieties of insider attacks considered in [23, 13]* and present a protocol compiler to obtain protocols that are secure in this model. On the other hand, no efficient two-round protocol as [21] is available in the UC framework and the protocol we give in Section 4.2 cannot be obtained by the protocol compiler of Katz and Shin. The formulation of key agreement in this setting is also unclear [16]. The definition of agreement in [18] differs from the one we give below, and in particular does not quantify the influence maliciously acting protocol participants have on the session key. Another significant difference between the approach in [18] and the one below is the role of the session identifier. Unlike [18] we do not assume a session identifier to be available from a protocol-external context, but take it as a goal of the group key establishment to come up with a session identifier that can serve as non-secret identifier for the established key.

2 Security Model and Security Goals

As indicated already, our basic security model is the proof model [9] in the way it is used by Katz and Yung in [20]. Before we motivate and describe our extensions we give a short summary of the model:

Participants. We model the (potential) protocol participants as a finite set \mathcal{U} of fixed size with each $U_i \in \mathcal{U}$ being a probabilistic polynomial time (ppt) Turing machine. Each protocol participant $U_i \in \mathcal{P} (\subseteq \mathcal{U})$ may execute a polynomial number of protocol instances in parallel. We will refer to instance s_i of principal U_i as *instance* $\Pi_i^{s_i}$ ($i \in \mathbb{N}$). Each such instance may be taken for a process executed by U_i and has assigned seven variables $\text{state}_i^{s_i}$, $\text{sid}_i^{s_i}$, $\text{pid}_i^{s_i}$, $\text{sk}_i^{s_i}$, $\text{term}_i^{s_i}$, $\text{used}_i^{s_i}$ and $\text{acc}_i^{s_i}$:

$\text{used}_i^{s_i}$ indicates whether this instance is or has been used for a protocol run. The $\text{used}_i^{s_i}$ flag can only be set through a protocol message received by the instance due to a call to the Send oracle (see below);

$\text{state}_i^{s_i}$ keeps the state information during the protocol execution;

$\text{term}_i^{s_i}$ shows if the execution has terminated;

$\text{sid}_i^{s_i}$ denotes a (non-secret) session identifier that can serve as identifier for the session key $\text{sk}_i^{s_i}$;

$\text{pid}_i^{s_i}$ stores the set of identities of those principals that $\Pi_i^{s_i}$ aims at establishing a key with— including U_i himself;

$\text{acc}_i^{s_i}$ indicates if the protocol instance was successful, i. e., the principal accepted the session key;

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

For more details on the usage of the variables see [2]. We suppose that an instance $\Pi_i^{s_i}$ must accept the session key constructed at the end of the corresponding protocol instance if no deviation from the protocol specification occurs.

Communication network. We assume that arbitrary point-to-point connections among the principals are available. As connections are potentially under adversarial control (cf. the adversarial model below) the network is non-private and it is fully asynchronous.

Adversarial model. The adversary \mathcal{A} is modelled as a ppt Turing machine and considered to be active: \mathcal{A} has full control of the communication network and may delay, eavesdrop, suppress, alter and insert messages at will. To make the adversary's capabilities explicit, the subsequently listed oracles are used that can be executed by \mathcal{A} .

Send(U_i, s_i, M) This oracle serves two purposes:

- If $\text{used}_i^{s_i} = \text{true}$, the message message M is sent to the instance $\Pi_i^{s_i}$. If $\Pi_i^{s_i}$ sends a message in the protocol right after receiving M , then the Send oracle returns this message to the adversary.

- If $\text{used}_i^{s_i} = \text{false}$, the message M has to be of the form $M = \{U_1, \dots, U_n\}$ with $U_1, \dots, U_n \in \mathcal{U}$. In this way the adversary can initialize a protocol run among principals U_1, \dots, U_n . After such a query, $\Pi_i^{s_i}$'s $\text{pid}_i^{s_i}$ -value is initialized to $\{U_1, \dots, U_n\} \cup \{U_i\}$, the $\text{used}_i^{s_i}$ -flag is set and $\Pi_i^{s_i}$ processes the first step of the protocol. This means that in this session, U_i aims at establishing a common key with the principals specified in M .

$\text{Reveal}(U_i, s_i)$ yields the session key $\text{sk}_i^{s_i}$ provided that it is defined, i.e., if $\text{acc}_i^{s_i} = \text{true}$ and $\text{sk}_i^{s_i} \neq \text{NULL}$. Otherwise the distinguished NULL-value is returned.

$\text{Corrupt}(U_i)$ reveals the long term secret key SK_i of U_i to the adversary. Given a concrete protocol run, involving instances $\Pi_i^{s_i}$ of principals U_1, \dots, U_n , we say that principal $U_{i_0} \in \{U_1, \dots, U_n\}$ is *honest* if and only if no query of the form $\text{Corrupt}(U_{i_0})$ has ever been made by the adversary.

$\text{Test}(U_i, s_i)$ Only one query of this form is allowed for the adversary \mathcal{A} . Provided that $\text{sk}_i^{s_i}$ is defined, (i.e. $\text{acc}_i^{s_i} = \text{true}$ and $\text{sk}_i^{s_i} \neq \text{NULL}$), \mathcal{A} can execute this oracle query at any time when being activated. Then with probability 1/2 the session key $\text{sk}_i^{s_i}$ and with probability 1/2 a uniformly chosen random session key is returned.

As the session identifier $\text{sid}_i^{s_i}$ is intended to serve as public identifier for the session key $\text{sk}_i^{s_i}$, we also grant the adversary access to any session identifiers of her choice.

Initialization. Before the actual key establishment protocol is executed for the first time, an initialization phase takes place where for each principal $U_i \in \mathcal{P}$ a public key/secret key pair (SK_i, PK_i) is generated¹, SK_i is revealed to U_i only, and PK_i is given to all principals and the adversary.

Correctness. This property basically expresses that the protocol will establish a good key without adversarial interference and allows us to exclude “useless” protocols. We take a group key establishment protocol for *correct* if in the absence of attacks a common key along with a common identifier is established:

Definition 1. A group key establishment protocol \mathbf{P} is referred to as correct if upon honest delivery of all messages and no **Corrupt**-queries being made, a single execution of the protocol for establishing a key among U_1, \dots, U_n involves n instances $\Pi_1^{s_1}, \dots, \Pi_n^{s_n}$ and ensures that with overwhelming probability all instances:

- accept, i.e., $\text{acc}_1^{s_1} = \dots = \text{acc}_n^{s_n} = \text{true}$.
- obtain a common session identifier $\text{sid}_1^{s_1} = \dots = \text{sid}_n^{s_n}$ which is globally unique.
- have accepted the same session key

$$\text{sk}_1^{s_1} = \dots = \text{sk}_n^{s_n} \neq \text{NULL}$$

associated with the common session identifier $\text{sid}_1^{s_1}$.

- know their partners

$$\text{pid}_1^{s_1} = \text{pid}_2^{s_2} = \dots = \text{pid}_n^{s_n} = \{U_1, \dots, U_n\}.$$

¹For the sake of simplicity we assume these key pairs to be generated by a trusted party, i.e., we do not consider malicious parties who try to generate incorrect key pairs. Also, we do not consider scenarios where only low-entropy secrets, like passwords, are available for authentication.

Partnering. To detail the security definition, we will have to specify under which conditions a Test-query may be executed. To do so we fix the following notion of partnering.

Definition 2. *Two instances $\Pi_i^{s_i}, \Pi_j^{s_j}$ are partnered if $\text{sid}_i^{s_i} = \text{sid}_j^{s_j}$, $\text{pid}_i^{s_i} = \text{pid}_j^{s_j}$ and $\text{acc}_i^{s_i} = \text{acc}_j^{s_j} = \text{true}$.*

Freshness. A Test-query should only be allowed to those instances holding a key that is not for trivial reasons known to the adversary. An instance $\Pi_i^{s_i}$ is called *fresh* if none of the following two conditions hold:

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

The idea here 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. While the second condition seems pretty natural, imposing the first condition might look too restrictive. It is adopted from [5] and aims at precluding (insider) attacks where, once a subset of the principals has computed the session key, this subset is corrupted and the last outgoing messages are altered and correctly signed so that some honest protocol participants end up with a different session identifier but identical session key. Then the honest participants are not partnered with the corrupted ones, and breaking the security of the protocol becomes trivial. In the next section, we will discuss such a scenario for a protocol of Katz and Yung.

Security. The security definition of [9] can be summarized as follows. As a function of the security parameter k we define the advantage $\text{Adv}_{\mathcal{A}}(k)$ of a ppt adversary \mathcal{A} in attacking protocol P as

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

where Succ is the probability that the adversary queries Test on a fresh instance $\Pi_i^{s_i}$ and guesses correctly the bit b used by the Test oracle.

Definition 3. *We call the group key establishment protocol P secure if for any ppt adversary \mathcal{A} who does not violate the freshness of the Test instance the function $\text{Adv}_{\mathcal{A}} = \text{Adv}_{\mathcal{A}}(k)$ is negligible.*

3 Extended Security Properties

Unfortunately, established protocols that are proven secure in the above model can be vulnerable to annoyingly simple attacks if one considers a slightly broader scenario. In this section we explore the protocol of Katz and Yung [20] that goes back to Burmester and Desmedt [10] and a very efficient protocol from Kim, Lee and Lee [21]. We present new attacks on these protocols, but we stress that these attacks were not considered in the security model where they are proven secure: Hence our discussion does not invalidate the security proofs given by the authors. Nevertheless we think such vulnerabilities are relevant and should indeed be prevented.

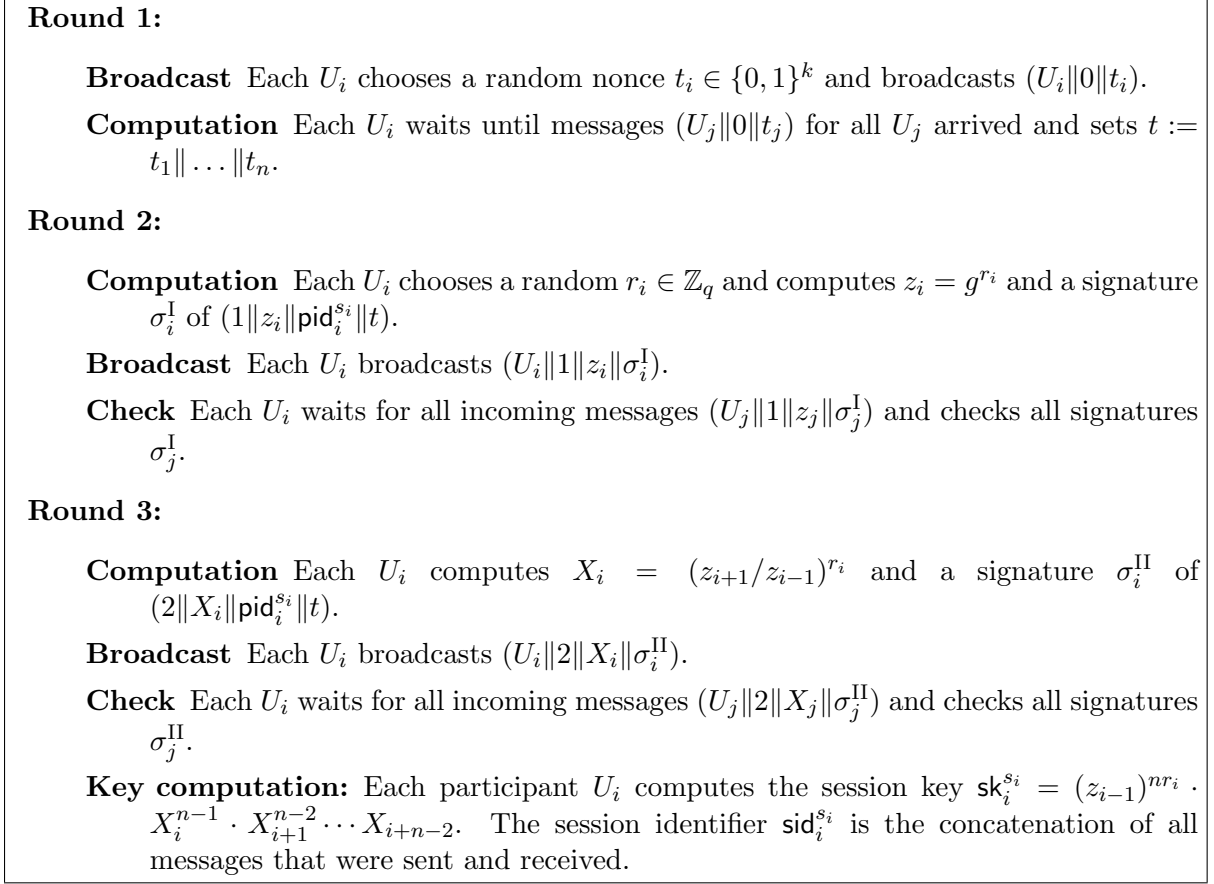


Figure 1: A group key establishment protocol from CRYPTO 2003 [20].

3.1 Attacks on a Proposal of Katz and Yung

At CRYPTO 2003, Katz and Yung put forward a three round group key agreement [20] building on the protocol of [10]. In an initialization phase a finite cyclic group \mathbb{G} of prime order q and a generator g of \mathbb{G} is chosen such that the Decisional Diffie Hellman (DDH) assumption holds. We summarize the fundamentals of the protocol for establishing a key among $\{U_1, \dots, U_n\}$, where indices are to be taken in a cycle. A detailed overview of the exchanged messages is given in Figure 1. Arbitrary point to point connections among participants are available, and a *broadcast* is understood as simultaneous point to point delivery of messages to all intended recipients. The participants exchange nonces in the first round to get a unique session. In the following, the participants broadcast $z_i = g^{r_i}$ and compute a Diffie-Hellman key with each of their neighbors. In the third round, the participants compute the quotient of the keys shared with their two neighbors $X_i = (z_{i+1}/z_{i-1})^{r_i}$ and broadcast this value. It is now possible for all participants to compute the key $\text{sk}_i^{s_i} = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \dots \cdot X_{i+n-2}$.

Using a model close to the one outlined in Section 2, in [20] this protocol is shown to be secure. At this, it is assumed that the signature scheme used is not only secure against existential forgeries under adaptive chosen message attacks, but with overwhelming probability also prevents an attacker from producing a different signature for an already signed message (*strong unforgeability*).

Violating the integrity of a session. Let us assume the adversarial goal is now to prevent a certain session unnoticeably from succeeding, forcing some involved principals to obliviously compute a different session key with the same session identifier. (In Definition 4 we will formalize resilience against this type of attack as *integrity*.) Say $n > 3$ and $\text{ord}(g)$ are coprime, then the adversary \mathcal{A} can mount the following (insider) attack, here shown on an example with four participants U_1, \dots, U_4 :

1. \mathcal{A} corrupts U_1 and U_3 and impersonates them for the protocol execution, in the first two rounds following the protocol description honestly.
2. In the 3rd protocol round, \mathcal{A} computes X_1, X_3 as specified, but then sets $\widetilde{X}_1 := X_3, \widetilde{X}_3 := X_1$. Now \mathcal{A} signs $(U_1 \| 2 \| \widetilde{X}_1 \| t)$ with U_1 's signing key, signs $(U_3 \| 2 \| \widetilde{X}_3 \| t)$ with U_3 's signing key and then broadcasts the messages $(U_1 \| 2 \| \widetilde{X}_1 \| \sigma_1^I)$ and $(U_3 \| 2 \| \widetilde{X}_3 \| \sigma_3^I)$. In other words, \mathcal{A} swaps the X_i -contributions of U_1 and U_3 .

Now all protocol participants compute the same session identifier, all of them receive the same messages, but with overwhelming probability the (honest) participants U_2 and U_4 will have derived different session keys: With the notation from Figure 1 a simple computation shows that the quotient of U_2 's and U_4 's session keys is $X_3^n \cdot (z_2/z_4)^{r_{3n}} = 1_{\mathbb{G}}$ without and $X_1^n \cdot (z_2/z_4)^{r_{3n}}$ with U_1 and U_3 swapping their X_i -contributions in the 3rd protocol round. Thus, in the latter case the keys derived by U_2 and U_4 coincide with negligible probability only. This actually violates the *correctness* of [20] where matching keys are required for all instances that accepted in the same session. As this is a non-trivial attack we restricted the correctness explicitly to honest executions and will introduce a broadened requirement—named *integrity*—in Section 3.3.

Moreover it is now easy to see that in the above scheme not every participant contributes to the session key. In fact, the key can be completely determined by an adversary corrupting two neighboring principals U_i, U_{i+1} . In Section 4.1 we prove that corrupting only one principal does not suffice for successfully attacking this scheme in a similar fashion.

3.2 Attacks on a Proposal of Kim, Lee and Lee

At ASIACRYPT 2004, Kim, Lee and Lee presented an efficient authenticated group key agreement protocol [21], that is claimed to take precautions against “illegal members or system faults”. No formal definition or security proof for this is provided, however, and below we will see that the protocol does not meet strong security guarantees as one malicious participant is sufficient to violate integrity and to mount an impersonation attack.

Figure 2 outlines Kim, Lee and Lee’s proposal for establishing a key among $\{U_1, \dots, U_n\}$, where again indices are to be taken in a cycle. Similarly as in the proposal of Katz and Yung, during an initialization phase a cyclic group \mathbb{G} of prime order q along with a generator g is chosen such that the CDH assumption holds; the hash function $H(\cdot)$ is modelled as random oracle and again *broadcast* is understood as simultaneous point to point delivery of messages. The protocol begins with the participants broadcasting $y_i = g^{x_i}$, again to establish Diffie-Hellman keys t_i^L, t_i^R with their two neighbored participants. In the second round the participants broadcast the XOR sum $T_i = t_i^L \oplus t_i^R$ of their two keys to allow all participants to compute all shared keys. Moreover they broadcast a nonce k_i as contribution to the session key, though one participant broadcasts his nonce encrypted $k_n \oplus t_n^R$. Now all participants can compute the nonces and the session key $\text{sk}_i^{S_i} = H(k_1 \| \dots \| k_n \| 0)$.

Round 1:

Computation Each U_i chooses $k_i \in \{0, 1\}^k$, $x_i \in \mathbb{Z}_q^*$ and computes $y_i = g^{x_i}$, only U_n computes additionally $H(k_n || 0)$. Each U_i except U_n sets $M_i^I = y_i$ and U_n sets $M_n^I = H(k_n || 0) || y_n$. Each U_i computes a signature σ_i^I on $M_i^I || \text{pid}_i^{s_i} || 0$.

Broadcast Each U_i broadcasts $(M_i^I || \sigma_i^I)$.

Check Each U_i checks all signatures σ_j^I of incoming messages $(M_j^I || \sigma_j^I)$.

Round 2:

Computation Each U_i computes $t_i^L = H(y_{i-1}^{x_{i-1}} || \text{pid}_i^{s_i} || 0)$, $t_i^R = H(y_{i+1}^{x_{i+1}} || \text{pid}_i^{s_i} || 0)$ and $T_i = t_i^L \oplus t_i^R$, only U_n computes additionally $k_n \oplus t_n^R$. The participants U_1, \dots, U_{n-1} set $M_i^{II} = k_i || T_i$, U_n sets $M_n^{II} = k_n \oplus t_n^R || T_n$ and each U_i computes a signature σ_i^{II} of $M_i^{II} || \text{pid}_i^{s_i} || 0$.

Broadcast Each U_i broadcasts $(M_i^{II} || \sigma_i^{II})$.

Check Firstly, each U_i checks all signatures σ_j^{II} of incoming messages. Then each U_i checks if $T_1 \oplus \dots \oplus T_n = 0$, computes t_n^R to obtain k_n from U_n 's message and checks the commitment $H(k_n || 0)$ for k_n .

Key computation: Each participant U_i computes the session key $\text{sk}_i^{s_i} = H(k_1 || \dots || k_n || 0)$.

Figure 2: A group key establishment protocol from ASIACRYPT 2004 [21].

Attacks on the security and integrity. In [21] it is not specified how to generate the session identifier $\text{sid}_i^{s_i}$, and it turns out that the standard method of concatenating all messages an instance sent and received is not enough to prove it secure: For $n > 3$, an adversary \mathcal{A} could proceed as follows to provoke a situation where U_1 and U_3 end up with different session identifiers (hence not being partnered) but identical session key $\text{sk}_1^{s_1} = \text{sk}_3^{s_3}$:

1. \mathcal{A} executes a complete protocol run and eavesdrops the message $(M_1^I \parallel \sigma_1^I)$ broadcast by U_1 in Round 1.
2. \mathcal{A} initiates another protocol execution, but in Round 1 replaces the message sent from U_1 to U_3 with the old $(M_1^I \parallel \sigma_1^I)$ -value which was eavesdropped in the previous protocol run.

Because of U_3 not being a neighbor of U_1 , this message substitution does not affect the computation of the session key, but with overwhelming probability U_3 ends up with a session identifier different from the session identifier computed by U_1 and the respective instances of U_1 and U_3 will not be partnered. Therefore, the key of U_1 can be revealed but U_3 remains fresh. Also the attack from Section 3.1 aiming at a different session key under the same session identifier applies here in an analogous way.

To avoid this kind of “trivial” problems, subsequently we assume the session identifier $\text{sid}_i^{s_i}$ to be derived as

$$\text{sid}_i^{s_i} = H(k_1 \parallel \dots \parallel k_{n-1} \parallel H(k_n \parallel 0)),$$

so that identical session identifiers with overwhelming probability correspond with identical session keys. However, this session identifier still allows a single protocol run ending up with different session identifiers. This can be provoked by simply having a malicious participant U_1 in Round 2 sending different k_1 -values to the other protocol participants (instead of broadcasting one k_1 -value).

An impersonation attack Independent from the choice of the session identifier and potentially more severe is the following impersonation attack. For $n > 2$ participants an adversary \mathcal{A} can impersonate participants as follows:

1. First, she gets herself a protocol transcript of a successful key establishment among principals U_1, \dots, U_n . Next, \mathcal{A} reveals U_1 's long term secret by querying $\text{Corrupt}(U_1)$.
2. Now \mathcal{A} initializes unused instances of U_3, \dots, U_n with

$$\text{pid}_j^{s_j} = \{U_1, \dots, U_n\}.$$

3. In Round 1 she replays the message that U_2 sent in the previously eavesdropped key establishment and participates honestly for U_1 .
4. In Round 2, \mathcal{A} again replays U_2 's message from the eavesdropped protocol run. On behalf of U_1 the adversary computes

$$T_1 := T_2 \oplus \dots \oplus T_n$$

and broadcasts the signed message $(M_1^{\text{II}} \parallel \sigma_1^{\text{II}})$ with $M_1^{\text{II}} = k_1 \parallel T_1$.

Now all participants can compute the session key and will accept it as common secret key among U_1, \dots, U_n although the honest principal U_2 never took part in the session.

3.3 Definition of Extended Security Goals

The model in [12] goes further in its definition of security than the model in [9]: Building on the notion of a matching session, a protocol is called secure if besides the usual negligible advantage in guessing the session key it also holds, that a matching session results in the participants accepting the same key. In a group key establishment protocol it is more appropriate to identify matching sessions via a session identifier. So in analogy to the two-party case, a matching session identifier should result in matching session keys. In the case of $n \gg 2$ protocol participants, there is significant potential for insider attacks, however, and malicious insiders indeed matter in group key establishment protocols: Even if several principals are dishonest, there can still remain numerous honest protocol participants.

Granted, in the presence of malicious participants the adversary always learns the session key, for honest participants the situation can still differ. For some applications it could even be more relevant to prevent the case in which honest principals share mismatching keys than the case where the correct key is shared with unintended principals. For instance, if the keys serve as access control passwords for shared data, then the above attacks result in situations in which principals assume others to have access rights which they may actually not have. We therefore propose the following notions to extend the security of group key establishments.

Session integrity. Motivated by the security definition of [3] and [12] we introduce an integrity property also for group key establishments to prevent sessions to mix up under adversarial influence. This property will basically be an extension of the correctness to active adversaries and malicious insiders.

We have seen an attack in Section 3.1 on the protocol of Katz and Yung where a malicious participant could run a protocol where two honest participants ended up with the same session identifier but different session keys. Such a situation possibly invalidates vital assumptions on the application level without the participants having a chance to detect it.

Recall also that the notion of correctness of a key establishment prevents partners from accepting when they have different $\text{pid}_i^{s_i}$ -values—i.e. they aim at establishing the key with different sets of users. We want to keep this property also in the presence of malicious insiders. For instance, assume a group key establishment, where a malicious U_1 could convince U_2 to have partners $\{U_1, U_2, U_3\}$ and U_3 to have partners $\{U_1, U_2, U_3, U_4\}$ when indeed only U_1, U_2 and U_3 have a common key. Then the honest principals U_2 and U_3 will not agree whether the subsequent application is confidential with respect to U_4 or not. To avoid such situations, in our definition of integrity we impose that a matching session identifier should also result in a matching partner identifier.

Definition 4. *We say a correct group key establishment protocol fulfills integrity if with overwhelming probability all instances of honest principals that have accepted with the same session identifier $\text{sid}_j^{s_j}$ hold identical session keys $\text{sk}_j^{s_j}$ and associate this key with the same principals $\text{pid}_j^{s_j}$.*

Strong entity authentication. Entity authentication is a relevant issue for key establishment even excluding the possibility of corrupted protocol participants. It is considered in the model [3] and in the models for password-based key establishment following [2]. Again malicious insiders are significantly stronger in violating this property, as seen in the attack scenarios in Section 3.2. An approach to define entity authentication formally was made in [17]. For our security model dealing with group key establishment we rephrase this definition as follows.

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

For the two-party case, the above definition is close to the mutual authentication requirements in [17], but instead of imposing exchanged messages seen by instances $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ to be equal, we require equality of the session identifiers.

Key agreement. Clearly, key freshness can never be guaranteed in the presence of malicious participants if some incomplete subset of principals is able to predetermine the key. However, if the key establishment is *contributory*, that is, if all parties must be involved in the construction of the key, we can at least provide some freshness guarantees. This kind of contributory key establishment protocols is usually referred to as *key agreement protocols*; however, some caution has to be taken here, as different notions of key agreement exist and not all of them suit our purposes. Before defining the type of key agreement we have in mind, it is worth to motivate this type of requirement:

- A protocol participant could be a program embedded in an environment that prevents protocol-external communication. In such a situation controlling (parts) of the session key may still be feasible, even if communicating a learned value is not.
- Even partial control over the session key can be useful, if on the application level only parts of the established session key are actually used for a specific purpose, say symmetric encryption. Another part of the established session key could be used for a purpose that is less relevant for the adversary.

The notion of key agreement we use is motivated by the discussion in [22] and imposes a quantitative restriction on the influence principals have on the derived session key. To express this security requirement, we split the adversarial action into two parts $\mathcal{A}_1, \mathcal{A}_2$. This separation is only for the ease of explanation, and we allow \mathcal{A}_1 and \mathcal{A}_2 to freely exchange state information a . In a precomputation phase, \mathcal{A}_1 tries to identify a favorable subset κ of the key space \mathcal{K} and a protocol participant U_i that seems likely to accept a key from κ . For instance, κ could be the set of all potential session keys with the most significant half of the bits being 0. The algorithm \mathcal{A}_2 controls the malicious participants during the actual protocol run and tries to establish a session key that is contained in κ .

Definition 6. Let $\tau \in \{1, \dots, |\mathcal{P}|\}$, P a key establishment protocol, and for a fixed pair of ppt algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ consider the following game:

1. The initialization phase of P establishing the long term keys is executed.
2. Having access to the public keys, the **Send** and **Reveal** oracle and being allowed up to $\tau - 1$ calls to the **Corrupt** oracle, \mathcal{A}_1 outputs a quadruple (i, s_i, χ_κ, a) with state information a and such that
 - U_i is honest with $\text{used}_i^{s_i} = \text{false}$;
 - χ_κ is a boolean-valued ppt algorithm with $\kappa := \{sk \in \mathcal{K} : \chi_\kappa(sk) = \text{true}\}$ such that $|\kappa|/|\mathcal{K}|$ is negligible in the security parameter.

3. Upon input of the state information a , \mathcal{A}_2 tries to make $\Pi_i^{s_i}$ accept a session key $sk_i^{s_i} \in \kappa$; for this, \mathcal{A}_2 has access to the **Send** and **Reveal** oracle, but may call the **Corrupt** oracle only with an argument $\neq U_i$ and as long as the total number of **Corrupt**-queries of \mathcal{A}_1 and \mathcal{A}_2 is $\leq t - 1$.

If there is no such pair $(\mathcal{A}_1, \mathcal{A}_2)$ with \mathcal{A}_2 succeeding with non-negligible probability, then we refer to \mathcal{P} as being τ -contributory. Moreover, by a key agreement we mean a $|\mathcal{P}|$ -contributory key establishment.

Summarizing, we take a group key establishment protocol for secure if it is correct, a proper subset of dishonest principals cannot predetermine the key, and it provides the “usual” confidentiality guarantees, integrity, and strong entity authentication:

Definition 7. We say a group key establishment protocol is secure against τ malicious participants if it is a correct $(\tau + 1)$ -contributory protocol in the sense of Definition 1 and Definition 6, secure in the sense of Definition 3, and assuming at most τ principals are dishonest, it offers integrity in the sense of Definition 4 and provides strong entity authentication to all participating instances in the sense of Definition 5. A group key establishment secure against $|\mathcal{P}| - 1$ malicious participants is referred to as a secure group key agreement.

4 Secure Authenticated Group Key Agreement

4.1 Looking back to Katz and Yung

To illustrate our extended model we show that the protocol of Katz and Yung is *partially* secure in this sense. The generation of the session identifier has to be modified, though. We moreover assume that all participants check for $\prod_i X_i = 1$ before accepting the key.

Proposition 4.1. Suppose that in Katz and Yung’s protocol described in Figure 1 all participants check for $\prod_i X_i = 1$ before accepting the key. Then, with session identifier $\text{sid}_i^{s_i} = \text{pid}_i^{s_i} || t$ (in this point diverging from [20]), we obtain a key establishment protocol secure against one malicious participant.

Proof. For correctness and security according to Definition 3 the proof of [20] applies. In the sequel, we assume only one participant in the key establishment is allowed to act maliciously.

We begin by defining two events, **Forge** and **Repeat**; the proof will follow from the fact that these events happen with negligible probability only.

Let us denote by **Forge** the event that the adversary succeeds in forging an authenticated message $M_{U_i} || \sigma_{U_i}$ for participant U_i without having queried **Corrupt**(U_i) and where M_{U_i} was not output by any of U_i ’s instances. An adversary \mathcal{A} that can reach **Forge** can be used for forging a signature for a given public key: This key is assigned to one of the n principals and \mathcal{A} succeeds in the intended forgery with probability $\geq \frac{1}{n} \cdot P(\text{Forge})$. Thus, using \mathcal{A} as black box we can derive an attacker defeating the existential unforgeability of the underlying signature scheme S with probability

$$P(\text{Forge}) \leq n \cdot \text{Adv}_S^{\text{cma}}.$$

Here $\text{Adv}_S^{\text{cma}}$ denotes the advantage of the adversary in violating the strong unforgeability under adaptive chosen message attack of the signature scheme, which is negligible by assumption. Thus, the event **Forge** occurs with negligible probability only.

Let **Repeat** be the event that an uncorrupted participant chooses a nonce t_i that was previously used by an instance of any user. Denoting by q_s a polynomial bound for the number of queries to the **Send** oracle, there are at most q_s used instances that may have chosen a nonce t_i and thus **Repeat** happens with a probability

$$P(\text{Repeat}) \leq \frac{q_s^2}{2^k},$$

again negligible in k .

Integrity. Let us suppose an adversary \mathcal{A} aims at violating integrity as defined in Definition 4, however, she is only able to make one corrupt call to, say, principal U_c . Recall that the adversarial goal is to make two honest principals that accept a fixed session **sid** hold either different session keys or **pid** values. The adversary cannot achieve the latter, as once the session identifier is fixed its **sid** contains **pid**, shared thus to all honest principals in that session.

Let us see why \mathcal{A} cannot violate integrity either, by forcing honest principals that have accepted with the same **sid** to share different keys. Let U_a, U_b be any two honest users, whose instances $\Pi_a^{s_a}$ and $\Pi_b^{s_b}$ accept and hold a common session identifier **sid**. Let U_i be any honest user that participated (including U_a and U_b). Unless the event **Forge** takes place, $\Pi_a^{s_a}$ and $\Pi_i^{s_b}$ received a signed message generated by a U_i instance. If the nonce t_i , which was included in the signature, is unique, only *one* message M_i^{II} respectively M_i^{III} signed by U_i and including the nonce t_i exists. The nonces are unique as long as the event **Repeat** does not happen. Thus, $\Pi_a^{s_a}$ and $\Pi_b^{s_b}$ hold the same value X_i for user U_i . The same argument holds for all honest users U_i , i.e. all users except U_c . Hence, $\Pi_a^{s_a}$ and $\Pi_b^{s_b}$ hold the same set of values $\{X_1, \dots, X_n\} \setminus \{X_c\}$. However, X_c is implicitly fixed by the values X_i of the honest participants as $X_c = (\prod_{i \neq c} X_i)^{-1}$. Summarizing, the adversary can only violate the integrity of the above scheme with a probability bounded from above by $P(\text{Forge}) + P(\text{Repeat})$.

Entity authentication. Let us see, why entity authentication is provided for an honest user U_i whose instance $\Pi_i^{s_i}$ has accepted. The concatenation of the nonces t_i computed in the first round is fresh as long as one honest instance is involved. Let $U_j \in \text{pid}_i^{s_i}$ be any honest user participating in the protocol. User U_j computes a signature over the message $(1||z_j||\text{pid}_j^{s_j}||t)$. It is assured that in absence of **Forge** at the end of the second round $\Pi_i^{s_i}$ and one instance $\Pi_j^{s_j}$ of U_j indeed hold the same $\text{pid}_i^{s_i} = \text{pid}_j^{s_j}$ and the same value t . Therefore they also compute the same session identifier **sid** = **pid**|| t . Summarizing, the adversary cannot break the strong entity authentication with probability that is above $P(\text{Forge})$.

2-Contributory. Note that the adversary cannot influence the key by a dedicated choice of one principal’s “random” selections in the first two rounds. Obviously, the random nonce in the first round does not influence the key. In the second round the adversary chooses values for a Diffie-Hellman key exchange. Assumed it is not allowed to choose the exponent $r_i = 0$ the probability that the resulting key is included in a predefined negligible fraction of the key space specified is negligible. For $n \geq 3$ this is also true, even allowing exponent 0. \square

4.2 A Secure Two-Round Protocol

As shown in Section 3.2, the proposal of Kim, Lee and Lee in Figure 2 does not offer the discussed security guarantees. In Figure 3 we present—with the notation from Section 3.2—a variant of the

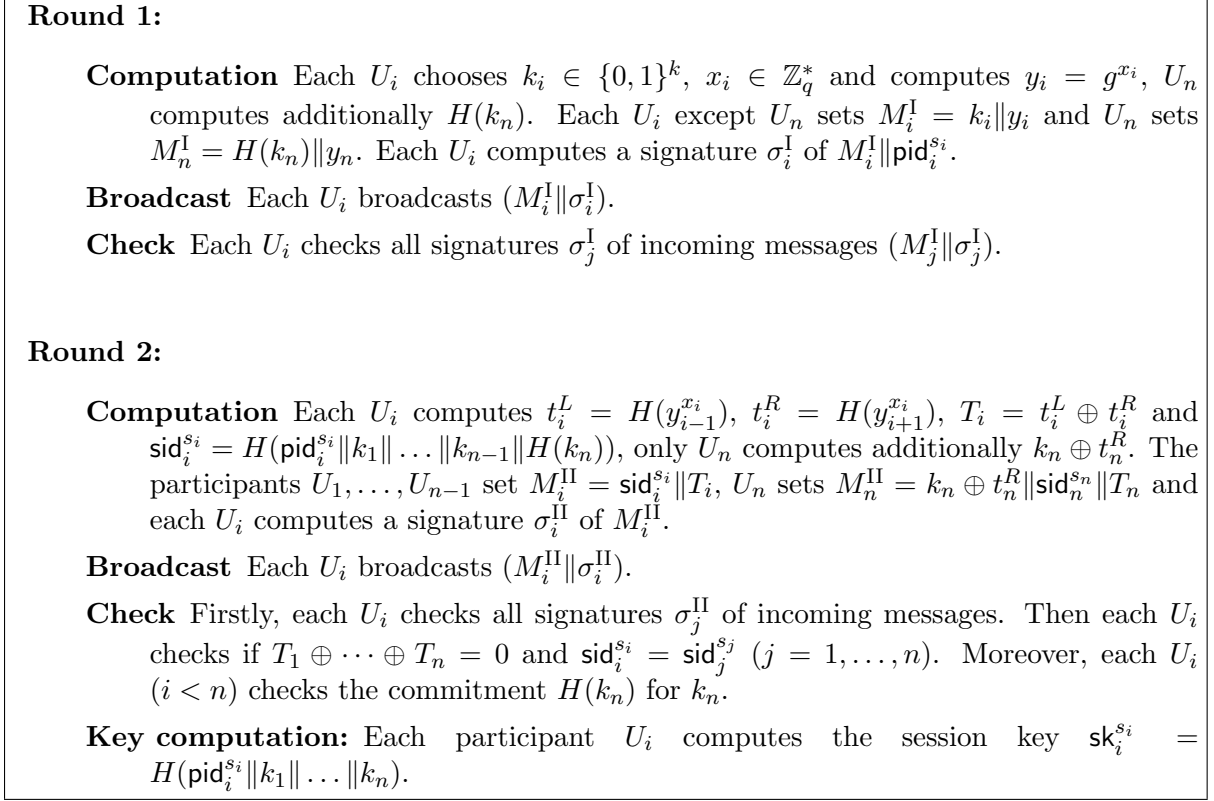


Figure 3: A secure group key agreement protocol.

protocol that again consists of two rounds, but in the presence of malicious participants offers the security guarantees from Definition 7. We changed the protocol so that all participants U_i except U_n send their contribution k_i to the session key already in the first round. Thus the session key is fixed by the messages of the first round. This allows the participants in the second round to send a confirmation of the key material, namely the value $H(\text{pid}_i^{s_i} \| k_1 \| \dots \| k_{n-1} \| H(k_n))$, to certify that all of them will compute the same session key. Therewith, the attacks from Section 3.2 are effectively defeated.

The protocol allows U_n a rushing attack, waiting in the first round until k_1, \dots, k_{n-1} are known and then choosing k_n depending on these. This attack is also possible in the protocol of Kim et al., however in the second round and for all participants except U_n . One may argue that stronger security requirements on the key *agreement* property of the protocol should be imposed, so that the adversary cannot predetermine any bit of the session key (cf. [22]). Transforming the above protocol accordingly is possible for the price of slightly increasing the computational effort of the involved parties: Instead of broadcasting the k_i -values in Round 1, in the first round only commitments $H(k_i)$ are sent and the k_i values are transmitted in the second round. However, the present protocol fulfills the requirements according to Definition 6.

Proposition 4.2. *Suppose that the CDH assumption holds for (\mathbb{G}, g) , $H(\cdot)$ is a random oracle and the underlying signature scheme is existentially unforgeable under adaptive chosen message attacks. Then the protocol in Figure 3 is a secure group key agreement in the sense of Definition 7.*

Proof. Let q_s and q_{ro} be polynomial bounds for the number of the adversary's queries to the Send respectively the random oracle. We begin by defining three events that will occur in several places throughout the proof, and we give bounds for the probability of these events that are negligible in k .

Forge is the event that the adversary succeeds in forging an authenticated message $M_{U_i} \parallel \sigma_{U_i}$ for participant U_i without having queried $\text{Corrupt}(U_i)$ and where M_{U_i} was not output by any of U_i 's instances. An adversary \mathcal{A} that can reach **Forge** can be used for forging a signature for a given public key: This key is assigned to one of the n principals and \mathcal{A} succeeds in the intended forgery with probability $\geq \frac{1}{n} \cdot P(\text{Forge})$. Thus, using \mathcal{A} as black box we can derive an attacker defeating the existential unforgeability of the underlying signature scheme S with probability

$$\begin{aligned} \text{Adv}_S^{\text{cma}} &\geq \frac{1}{n} \cdot P(\text{Forge}) \\ \iff P(\text{Forge}) &\leq n \cdot \text{Adv}_S^{\text{cma}}. \end{aligned}$$

Here $\text{Adv}_S^{\text{cma}}$ denotes the advantage of the adversary in violating the existential unforgeability under an adaptive chosen message attack of the signature scheme, which is negligible by assumption. Thus, the event **Forge** occurs with negligible probability only.

Collision is the event that the random oracle produces a collision. A **Send** query causes at most 3 random oracle calls. Thus, the total number of random oracle queries is bounded by $3q_s + q_{ro}$ and the probability that a collision of the random oracle occurs is

$$P(\text{Collision}) \leq \frac{(3q_s + q_{ro})^2}{2^k},$$

which is negligible in k .

Repeat is the event that an uncorrupted participant chooses a nonce k_i that was previously used by an oracle of some principal. There are at most q_s used instances that may have chosen a nonce k_i and thus **Repeat** happens with a probability

$$P(\text{Repeat}) \leq \frac{q_s^2}{2^k},$$

again negligible in k .

Security. To prove the security according to Definition 3, we consider a sequence of games. In these games we let the adversary \mathcal{A} interact with a simulator, that in Game 0 offers the original protocol environment to \mathcal{A} , and subsequently we change the simulator's behavior in several small steps without affecting \mathcal{A} 's success probability significantly. Keeping track of the changes between subsequent games, in the last game we will be able to derive the desired negligible upper bound on $\text{Adv}_{\mathcal{A}}$.

Game 0: In this game the protocol participants' instances are faithfully simulated for the adversary, i. e., the adversary's situation is the same as in the real model.

$$\text{Adv}_{\mathcal{A}}^{\text{Game 0}} = \text{Adv}_{\mathcal{A}}.$$

Game 1: This game is aborted if one of the events **Forge**, **Collision** or **Repeat** occurs. Otherwise the

game is identical with Game 0 and the adversary cannot detect the difference. Thus, for adversary \mathcal{A} 's advantage we have

$$\begin{aligned} |\text{Adv}_{\mathcal{A}}^{\text{Game 1}} - \text{Adv}_{\mathcal{A}}^{\text{Game 0}}| &\leq P(\text{Forge}) + P(\text{Collision}) \\ &+ P(\text{Repeat}). \end{aligned}$$

Game 2: This game differs from Game 1 in the simulator's response in Round 2. If the simulator has to output the message of an instance $\Pi_i^{s_i}$ and none of the participants $U_\ell \in \text{pid}_i^{s_i}$ is corrupted, then the simulator chooses random values from $\{0, 1\}^k$ for $t_i^L = t_{i-1}^R$ and $t_i^R = t_{i+1}^L$ instead of querying the random oracle. To keep consistency, the same values have to be used in the neighbored instances subsequently². By the random oracle assumption, the adversary can only detect the difference by querying the random oracle for $y_{i-1}^{x_i} = y_i^{x_{i-1}}$.

An adversary \mathcal{A} that distinguishes Game 1 and Game 2 can be used as black box to solve a CDH instance. Two instances $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ are selected by randomly choosing two different users $U_i, U_j \in \mathcal{P}$ plus two numbers $s_i, s_j \in \{1, \dots, q_s\}$. Game 2 only differs from Game 1, if at least one session is set up of uncorrupted users. To distinguish the games, the adversary has to query the random oracle with at least one Diffie-Hellman key, established between neighbors in a session with uncorrupted participants. These randomly chosen instances will be those neighbored participants with probability at least $1/(n \cdot q_s)^2$.

A given CDH instance (g^a, g^b) is then assigned to $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ such that these instances will use g^a respectively g^b as their message for the first round.

If at some point now $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ do not qualify any longer to be neighbored participants in an session with only uncorrupted users, the simulation is aborted (as noted, this happens with probability bounded by $1/(n \cdot q_s)^2$).

Then a random index $z \in \{1, \dots, q_{ro}\}$ is chosen and the adversary's z -th query to the random oracle is taken for the answer to the CDH challenge. The answer to the CDH challenge is correct if \mathcal{A} distinguished the games with the chosen instances and also z was determined correctly. So we have

$$|\text{Adv}_{\mathcal{A}}^{\text{Game 2}} - \text{Adv}_{\mathcal{A}}^{\text{Game 1}}| \leq \text{Succ}_{(\mathbb{G}, g)}^{\text{CDH}} \cdot q_{ro} \cdot n^2 \cdot q_s^2,$$

where $\text{Succ}_{(\mathbb{G}, g)}$ is an — under the CDH assumption — negligible upper bound for the success probability of the above algorithm to solve CDH.

Game 3: In this game the simulator changes the computation of the session key. Having received all messages of Round 2 for an instance $\Pi_i^{s_i}$, the simulator checks if all $U_j \in \text{pid}_i^{s_i}$ are uncorrupted. If so, then the simulator chooses a session key $\text{sk}_i^{s_i} \in \{0, 1\}^k$ at random instead of querying the random oracle. For consistency the simulator will assign the same key to all partnered instances.

The only way for the adversary to detect the difference is by querying the random oracle for $H(\text{pid}_i^{s_i} \| k_1 \| \dots \| k_n)$. However, k_n is information-theoretically unknown to the adversary. Thus, the adversary can only guess a random value for k_n and query the random oracle at most q_{ro} times. This results in:

$$|\text{Adv}_{\mathcal{A}}^{\text{Game 3}} - \text{Adv}_{\mathcal{A}}^{\text{Game 2}}| \leq \frac{q_{ro}}{2^k}.$$

²In the first round, the participants include the pid in the signature and thus know afterwards, that they obtained the same value pid . Thus, being neighbored is well defined.

None of the partners of the adversary's `Test`-instance are allowed to be corrupted or to be revealed (see Definition 3). Thereby, those instances were affected in Game 3 and use a random value as session key. Therefore, the adversary has only a probability of $\frac{1}{2}$ for guessing the bit of `Test`, yielding

$$\text{Adv}_{\mathcal{A}}^{\text{Game 3}} = 0.$$

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

$$\begin{aligned} \text{Adv}_{\mathcal{A}} \leq & P(\text{Forge}) + P(\text{Collision}) + P(\text{Repeat}) + \\ & \text{Succ}_{(\mathbb{G},g)}^{\text{CDH}} \cdot q_{ro} \cdot n^2 \cdot q_s^2 + \frac{q_{ro}}{2^k}. \end{aligned}$$

Integrity. Let `NoIntegrity` be the event that some instance violates the condition imposed in Definition 4. To determine the probability of `NoIntegrity` let U_i and U_j be any two honest principals whose instances $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ accept (`acc = true`) with a matching session identifier $\text{sid} := \text{sid}_i^{s_i} = \text{sid}_j^{s_j}$. The session identifier `sid` is unique if uncorrupted principals contributed fresh nonces k_i (unless `Repeat`) and the random oracle is collision-free (unless `Collision`). Moreover the messages of uncorrupted principals cannot be forged (unless `Forge`) by the adversary. Thus $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ must have received each other's message $\text{sid}_i^{s_i} \| T_i \| \sigma_i^{\text{II}}$ resp. $\text{sid}_j^{s_j} \| T_j \| \sigma_j^{\text{II}}$, where necessarily $\text{sid} := \text{sid}_i^{s_i} = \text{sid}_j^{s_j}$ matched due to the check phase.

The construction of `sid` assures that $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ hold the same $\text{pid} := \text{pid}_i^{s_i} = \text{pid}_j^{s_j}$ (obtained in the respective instance's initialization) and know the same values k_1, \dots, k_{n-1} and $H(k_n)$. Again by collision-freeness of the random oracle, $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ have received the same k_n and therewith compute the same session key $\text{sk}_i^{s_i} = \text{sk}_j^{s_j}$. Thus, putting things together we obtain the desired negligible upper bound

$$P(\text{NoIntegrity}) \leq P(\text{Collision}) + P(\text{Repeat}) + P(\text{Forge}).$$

Entity authentication. Denote by `EntAuthFail` be the event that strong entity authentication fails. We consider entity authentication in Game 1. Let U_i be any principal with an instance $\Pi_i^{s_i}$ that has accepted. It is easy to see that entity authentication is provided to $\Pi_i^{s_i}$: Since $\Pi_i^{s_i}$ has accepted, in Round 2 it received messages including the session identifier `sid` from all principals $U_\ell \in \text{pid}_i^{s_i}$ (unless `Forge`). As above, in absence of `Collision`, `Repeat` and `Forge`, the session identifier is unique and the message cannot be replayed from a past session. Thus every honest partner holds the same session identifier `sid` and for the reasons stated above also the partner identifiers $\text{pid}_i^{s_i}$ and $\text{pid}_j^{s_j}$ match. Therewith entity authentication is violated with a probability

$$P(\text{EntAuthFail}) \leq P(\text{Collision}) + P(\text{Repeat}) + P(\text{Forge}).$$

Key Agreement. Let `NotAgreement` be the event that the adversary wins the experiment in Definition 6. The values relevant for deriving the session key are only the values k_i that participant U_i chooses in the first round. Unless `Repeat` occurs, an honest participant chooses a fresh value k_i . The adversary can learn the input k_i of the honest participant only during the protocol execution and can query the random oracle at most q_{ro} times for his inputs to influence the session key. The

random oracle’s answer is uniformly distributed over the key space, therefore only with negligible probability in the set κ of definition 6. In total we have:

$$P(\text{NotAgreement}) \leq P(\text{Repeat}) + q_{ro} \cdot \text{negl}(k),$$

which is clearly negligible in k .

Correctness of the protocol in Figure 3 is straightforward, and hence the proposition follows. \square

5 Conclusion

Building on established models for analyzing group key establishment protocols, the tools suggested in this paper offer a possibility to explore security properties of group key establishment protocols in the presence of malicious participants. The introduced framework in particular allows to show that a protocol proposed by Katz and Yung in [20] offers security guarantees against a single malicious participant “for free”, whereas a proposal of Kim, Lee and Lee [21] fails to do so. However, as shown in the last section, security against malicious participants is achievable in two rounds: without sacrificing efficiency, the discussed proposal of [21] can be modified to offer rather strong security guarantees even in the presence of malicious participants.

Acknowledgements

We are indebted to Dominique Unruh and the anonymous reviewers for their insightful comments and remarks.

References

- [1] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Proceedings of STOC 98*, pages 419–428. ACM, 1998.
- [2] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT’00*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, 2000.
- [3] Mihir Bellare and Phillip Rogaway. Entity 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, 1993.
- [4] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution — the three party case. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing, STOC’95*, pages 57–66. ACM Press, 1995.
- [5] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. Burmester-Desmedt Tree-Based Key Transport Revisited: Provable Security. Cryptology ePrint Archive: Report 2005/360, 2005. At the time of writing available electronically at <http://eprint.iacr.org/2005/360>.

- [6] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2004.
- [7] Colin Boyd and Juan Manuel González Nieto. Round-optimal Contributory Conference Key Agreement. In Yvo Desmedt, editor, *Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2003.
- [8] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–309. Springer, 2001.
- [9] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In Pierangela Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS-8)*, pages 255–264. ACM, 2001.
- [10] 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.
- [11] Christian Cachin and Reto Strobil. Asynchronous Group Key Exchange with Failures. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 357–366. ACM Press, 2004.
- [12] 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.
- [13] Zhaohui Cheng, Luminita Vasiu, and Richard Comley. Pairing-Based One-Round Tripartite Key Agreement Protocols. Cryptology ePrint Archive: Report 2004/079, 2004. At the time of writing available electronically at <http://eprint.iacr.org/2004/079>.
- [14] Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 585–604. Springer, 2005.
- [15] Kim-Kwang Raymond Choo, Colin Boyd, Yvonne Hitchcock, and Greg Maitland. On Session Identifiers in Provably Secure Protocols: The Bellare-Rogaway Three-Party Key Distribution Protocol Revisited. In Carlo Blundo and Stelvio Cimato, editors, *Fourth Conference on Security in Communication Networks - SCN 2004 Proceedings*, volume 3352 of *Lecture Notes in Computer Science*, pages 351–366. Springer, 2005.
- [16] Dennis Hofheinz, Jörn Müller-Quade, and Rainer Steinwandt. Initiator-Resilient Universally Composable Key Exchange. In Einar Snekkenes and Dieter Gollmann, editors, *Computer Security, Proceedings of ESORICS 2003*, volume 2808 of *Lecture Notes in Computer Science*, pages 61–84. Springer, 2003.

- [17] Shaoquan Jiang and Guang Gong. Password Based Key Exchange with Mutual Authentication. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography: 11th International Workshop, SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 267–279. Springer, 2004.
- [18] Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. Cryptology ePrint Archive: Report 2005/163, 2005. At the time of writing available electronically at <http://eprint.iacr.org/2006/163>. Full version of [19].
- [19] Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *12th ACM Conference on Computer and Communications Security*, pages 180–189. ACM Press, 2005.
- [20] 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.
- [21] Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In Pil Joong Lee, editor, *Advances in Cryptology — ASIACRYPT’04*, volume 3329 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 2004.
- [22] Chris J. Mitchell, Mike Ward, and Piers Wilson. Key control in key agreement protocols. *IEE Electronics Letters*, 34(10):980–981, 1998.
- [23] Shahrokh Saeednia and Rei Safavi-Naini. Efficient Identity-Based Conference Key Distribution Protocols. In Colin Boyd and Ed Dawson, editors, *Information Security and Privacy, Third Australasian Conference, ACISP’98*, volume 1438 of *Lecture Notes in Computer Science*, pages 320–331. Springer, 1998.
- [24] Victor Shoup. On Formal Models for Secure Key Exchange. Cryptology ePrint Archive: Report 1999/012, 1999. At the time of writing available electronically at <http://eprint.iacr.org/1999/012>.
- [25] Michael Steiner. *Secure Group Key Agreement*. PhD thesis, Universität des Saarlandes, 2002. At the time of writing available at http://www.semper.org/sirene/publ/Stein_02.thesis-final.pdf.
- [26] Wen-Guey Tzeng. A Practical and Secure Fault-Tolerant Conference-Key Agreement Protocol. In Hideki Imai and Yuliang Zheng, editors, *Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.