# Secure Group Key Establishment Revisited

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

[1] Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe, 76128 Karlsruhe, Germany;
`bohli@ira.uka.de`

[2] Área de Matemática Aplicada, Universidad Rey Juan Carlos, c/ Tulipán, s/n, 28933 Madrid, Spain;
`migonzalez@escet.urjc.es`

[3] Dept. 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. [BCPQ01,BCP01] and point out missing security properties that are present in some models for two-party key establishment. These properties are actually of more importance in group key establishments due to the possibility of malicious insiders. 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.

**Keywords:** Group Key Establishment, Provable Security, Malicious Insiders

## 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 which 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 [BM04]. To allow a rigorous security analysis, a framework for modelling group key establishments has been developed [BCP01,BCPQ01,KY03] going back on [BR93,BR95,BCK98,CK01,Sho99,BPR00]. A recent overview of those indistinguishability-based models is given in [CBH05].

The models [BR93] and [CK01] use the notion of *matching conversation* respectively *matching session* in order to reflect the situation in which two participants have received each others messages. This allows to identify each protocol session and make requirements on its integrity explicit. In contrast, the models for group key establishment distinguish sessions by so called *session identifiers*, what is more practical since in the group case all the parties involved do not necessarily receive the same messages. However, integrity properties are hence overseen, which is rather unfortunate considering malicious participants are more probable and devastating as the number of participants grows.

The work of Tzeng [Tze00], for instance, shows that it is clearly feasible to derive protocols with well-specified security guarantees even if a subset of the protocol participants acts maliciously. Unfortunately, so far for the quite popular security models [BCP01] and [BCPQ01] extensions to malicious participants have hardly been explored and analyzed. In this type of model analyses typically restrict to the case of honest participants (see, e. g., [BN03,KY03,CBHM05]).

Frameworks guaranteeing universal composability provide another approach to model key establishment protocols [Ste02] where insider attacks or failures are already considered [CS04,KS05]. However, no efficient two-round protocol as [KLL04] is available in those models and also the formulation of key agreement in those models is unclear [HMQS03]. Summarizing, more research on the suitability of those models for the formulation of key establishment protocols has to be done.

*Our contribution.* We examine two group key establishment protocols put forward in [KY03,KLL04] and point out attacks that are not covered by the model [BCPQ01] in which they are proven secure. We give a formal definition of some security notions motivated by these attacks. To our knowledge, these new notions round off the model [BCPQ01] and cover all known attacks that can be carried out in the presence of malicious insiders. To give a flavor of how to design secure protocols in this new strict sense, we prove a "hidden" security feature in a proposal from [KY03]. Finally we present a modification of [KLL04] that we can prove to be secure in our model—using the Computational Diffie Hellman (CDH) assumption as well as the random oracle model.

## 2 Security Model and Security Goals

As indicated already, our basic security model is the proof model [BCPQ01] in the way it is used by Katz and Yung in [KY03]. 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 $U_i$ being a probabilistic polynomial time (ppt) Turing machine. Each protocol participant $U_i \in \mathcal{P}$ may execute a polynomial number of protocol instances in parallel. We will refer to instance $s_i$ of principal $U_i$ as *oracle* $\Pi_i^{s_i}$ $(i \in \mathbb{N})$. Each such oracle may be taken for a process executed by $U_i$ and has assigned seven variables $\mathsf{state}_i^{s_i}$, $\mathsf{sid}_i^{s_i}$, $\mathsf{pid}_i^{s_i}$, $\mathsf{sk}_i^{s_i}$, $\mathsf{term}_i^{s_i}$, $\mathsf{used}_i^{s_i}$ and $\mathsf{acc}_i^{s_i}$:

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

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

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

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

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

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

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

For more details on the usage of the variables see [BPR00]. We suppose that an oracle $\Pi_i^{s_i}$ 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 arbitrary point-to-point connections among the principals to be available. As connections are potentially under adversarial control (cf. the adversarial model below) the network is non-private and fully asynchronous.

*Adversarial model.* For a passive adversary $\mathcal{A}$ all messages sent by protocol participants are sent as specified in the protocol description, but may be eavesdropped by $\mathcal{A}$. An active adversary $\mathcal{A}$ has full control of the communication network and may delay, suppress and insert messages at will. To make the adversary's capabilities explicit, the subsequently listed oracles are used: An active adversary is a ppt Turing machine which may execute any of these, whereas a passive adversary is a ppt Turing machine which is only given access to the $\mathsf{Execute}$, $\mathsf{Reveal}$ and $\mathsf{Test}$ oracles.

$\mathsf{Execute}(\{U_1, U_2, \ldots, U_r\})$ This executes the protocol among unused instances $\Pi_i^{s_i}$ of the specified parties and returns a transcript of the protocol run (listing all messages sent during the protocol execution among the oracles $\Pi_i^{s_i}$).

$\mathsf{Send}(U_i, s_i, M)$ This sends the message $M$ to the instance $\Pi_i^{s_i}$ and outputs the reply generated by this instance. If the adversary calls this oracle with an unused instance $\Pi_i^{s_i}$ and $M = \{U_1, \ldots, U_r\}$, then $\Pi_i^{s_i}$'s $\mathsf{pid}_i^{s_i}$-value is initialized to the value $\mathsf{pid}_i^{s_i} := M$ and the $\mathsf{used}_i^{s_i}$-flag is set. If the oracle $\Pi_i^{s_i}$ sends a message in the protocol right after receiving $M$, then $\mathsf{Send}$ returns this message to the adversary.

$\mathsf{Reveal}(U_i, s_i)$ yields the session key $\mathsf{sk}_i^{s_i}$ and the session identifier $\mathsf{sid}_i^{s_i}$.

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

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

*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 generated[1], $SK_i$ is revealed to $U_i$ only, and $PK_i$ is given to all principals.

*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 presence of a passive adversary indeed a common key along with a common identifier is established:

**Definition 1.** *A group key establishment protocol $\mathcal{P}$ is called* correct *if in the presence of a passive adversary a single execution of the protocol for establishing a key among $U_1, \ldots, U_r$ involves $r$ oracles $\Pi_1^{s_1}, \ldots, \Pi_r^{s_r}$ and ensures that with overwhelming probability all oracles:*

- *accept, i. e.,* $\mathsf{acc}_1^{s_1} = \cdots = \mathsf{acc}_r^{s_r} = \mathsf{true}$.
- *obtain a common session identifier* $\mathsf{sid}_1^{s_1} = \cdots = \mathsf{sid}_r^{s_r}$ *which is globally unique.*
- *have accepted the same session key* $\mathsf{sk}_1^{s_1} = \cdots = \mathsf{sk}_r^{s_r} \neq \text{NULL}$ *associated with the common session identifier* $\mathsf{sid}_1^{s_1}$.
- *know their partners* $\mathsf{pid}_1^{s_1} = \mathsf{pid}_2^{s_2} = \cdots = \mathsf{pid}_r^{s_r}$ *and it is* $\mathsf{pid}_1^{s_1} = \{U_1, \ldots U_r\}$.

*Partnering.* For detailing the security definition, we will have to specify under which conditions a $\mathsf{Test}$-query may be executed. To do so we fix the following notion of partnering.

**Definition 2.** *Two oracles $\Pi_i^{s_i}$, $\Pi_j^{s_j}$ are* partnered *if* $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$, $\mathsf{acc}_i^{s_i} = \mathsf{acc}_j^{s_j} = \mathsf{true}$ *and both* $U_j \in \mathsf{pid}_i^{s_i}$ *and* $U_i \in \mathsf{pid}_j^{s_j}$.

*Freshness.* A $\mathsf{Test}$-query should only be allowed to those oracles 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 $U_j \in \mathsf{pid}_i^{s_i}$ a $\mathsf{Corrupt}(U_j)$ query was executed before a query of the form $\mathsf{Send}(U_k, s_k, *)$ has taken place where $U_k \in \mathsf{pid}_i^{s_i}$.
- $\mathcal{A}$ queried $\mathsf{Reveal}(U_j, s_j)$ with $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ being partnered.

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

[1] For 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.

*Security.* The security definition of [BCPQ01] can be summarized as follows. As a function of the security parameter $k$ we define the advantage $\mathsf{Adv}_{\mathcal{A}}(k)$ of a ppt adversary $\mathcal{A}$ in attacking protocol $\mathsf{P}$ as

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

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

**Definition 3.** *We call the group key establishment protocol* $\mathsf{P}$ $\mathsf{secure}$ *if for any ppt adversary* $\mathcal{A}$ *the function* $\mathsf{Adv}_{\mathcal{A}} = \mathsf{Adv}_{\mathcal{A}}(k)$ *is negligible.*

## 3 Extended Security Properties

Established protocols proven secure in the above model are however vulnerable to simple attacks if one considers a slightly broader scenario. In this section we explore the protocol of Katz and Yung [KY03] that goes back to Burmester and Desmedt [BD95] and a very efficient protocol from Kim et al. [KLL04]. 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.

### 3.1 Attacks on a Proposal of Katz and Yung

At CRYPTO 2003, Katz and Yung put forward a three round group key agreement [KY03] building on the protocol of [BD95]. 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 $\{U_1, \ldots, U_n\}$, where indices are to be taken in a cycle. An 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 their neighbors. In the third round, the participants compute the quotient of the key with the 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 $\mathsf{sk}_i^{s_i} = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i+n-2}$.

Using a model close to the one outlined in Section 2, in [KY03] 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.

**Round 1:**

**Broadcast** Each $U_i$ chooses a random nonce $t_i \in \{0,1\}^k$ and broadcasts $(U_i \| 0 \| t_i)$.

**Computation** Each $U_i$ waits until messages $(U_j \| 0 \| t_j)$ for all $U_j$ arrived and sets $t := t_1 \| \dots \| t_n$.

**Round 2:**

**Computation** Each $U_i$ chooses a random $r_i \in \mathbb{Z}_q$ and computes $z_i = g^{r_i}$ and a signature $\sigma_i^1$ of $(1 \| z_i \| \mathsf{pid}_i^{s_i} \| t)$.

**Broadcast** Each $U_i$ broadcasts $(U_i \| 1 \| z_i \| \sigma_i^1)$.

**Check** Each $U_i$ waits for all incoming messages $(U_j \| 1 \| z_j \| \sigma_j^1)$ and checks all signatures $\sigma_j^1$.

**Round 3:**

**Computation** Each $U_i$ computes $X_i = (z_{i+1}/z_{i-1})^{r_i}$ and a signature $\sigma_i^2$ of $(2 \| X_i \| \mathsf{pid}_i^{s_i} \| t)$.

**Broadcast** Each $U_i$ broadcasts $(U_i \| 2 \| X_i \| \sigma_i^2)$.

**Check** Each $U_i$ waits for all incoming messages $(U_j \| 2 \| X_j \| \sigma_j^2)$ and checks all signatures $\sigma_j^2$ and $X_1 \cdots X_n = 1$.

**Key computation:** Each participant $U_i$ computes the session key $\mathsf{sk}_i^{s_i} = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i+n-2}$. The session identifier $\mathsf{sid}_i^{s_i}$ is the concatenation of all messages that were sent and received.

**Fig. 1.** A group key establishment protocol from CRYPTO 2003 [KY03].

*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.

Say $n > 3$ and $\mathrm{ord}(g)$ are coprime, then the adversary $\mathcal{A}$ can mount the following attack:

1. $\mathcal{A}$ corrupts $U_1$ and $U_3$ (henceforth blocking any communication from and to these parties).
2. $\mathcal{A}$ swaps the contributions of $U_1$ and $U_3$ to the 3rd protocol round, i. e., $\mathcal{A}$ computes $X_1, X_3$ as specified, signs $(U_1 \| 2 \| X_3 \| t)$ with $U_1$'s signing key, signs $(U_3 \| 2 \| X_1 \| t)$'s with $U_3$'s signing key and then broadcasts $(U_1 \| 2 \| X_3 \| \sigma_1^2)$ and $(U_3 \| 2 \| X_1 \| \sigma_3^2)$.

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 $X_3^n \cdot (z_2/z_4)^{r_3 n} = 1_{\mathbb{G}}$ without and $X_1^n \cdot (z_2/z_4)^{r_3 n}$ 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.

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 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 [KLL04], which 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 bradcast 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 $\mathsf{sk}_i^{s_i} = H(k_1 \| \dots \| k_n \| 0)$.

**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 $M_i^1 = y_i$ and $U_n$ sets $M_n^1 = H(k_n \| 0) \| y_n$. Each $U_i$ computes a signature $\sigma_i^1$ on $M_i^1 \| \mathsf{pid}_i^{s_i} \| 0$.

**Broadcast** Each $U_i$ broadcasts $(M_i^1 \| \sigma_i^1)$.

**Check** Each $U_i$ checks all signatures $\sigma_j^1$ of incoming messages $(M_j^1 \| \sigma_j^1)$.

**Round 2:**

**Computation** Each $U_i$ computes $t_i^L = H(y_{i-1}^{x_i} \| \mathsf{pid}_i^{s_i} \| 0)$, $t_i^R = H(y_{i+1}^{x_i} \| \mathsf{pid}_i^{s_i} \| 0)$ 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 $M_i^2 = k_i \| T_i$, $U_n$ sets $M_n^2 = k_n \oplus t_n^R \| T_n$ and each $U_i$ computes a signature $\sigma_i^2$ of $M_i^2 \| \mathsf{pid}_i^{s_i} \| 0$.

**Broadcast** Each $U_i$ broadcasts $(M_i^2 \| \sigma_i^2)$.

**Check** Firstly, each $U_i$ checks all signatures $\sigma_j^2$ of incoming messages. Then each $U_i$ checks if $T_1 \oplus \cdots \oplus T_n = 0$, decrypts $k_n$ and checks the commitment $H(k_n \| 0)$ for $k_n$.

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

**Fig. 2.** A group key establishment protocol from ASIACRYPT 2004 [KLL04].

*Remark 1.* In [KLL04] it is not specified how to generate the session identifier $\mathsf{sid}_i^{s_i}$, and it turns out that the standard method of concatenating all messages an oracle sent and received is not enough to prove it secure: For $n > 3$, an active 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 $\mathsf{sk}_1^{s_1} = \mathsf{sk}_3^{s_3}$:

1. $\mathcal{A}$ executes a complete protocol run and eavesdrops the message $M_1^i \| \sigma_i^1$ 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 $M_1^i \| \sigma_i^1$-value 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 oracles of $U_1$ and $U_3$ will not be partnered. Therefore, the key of $U_1$ can be revealed but $U_3$ remains fresh.

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

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

so that identical session identifiers with overwhelming probability result in identical session keys.

*Attacks on the integrity of a session.* A protocol run ending up with different session identifiers 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).

Also, it is possible to violate the integrity of a session by the following impersonation attack. For $n > 2$ participants an active adversary $\mathcal{A}$ can impersonate participants as follows:

1. First, she gets herself a protocol transcript of a successful key establishment among principals $U_1, \dots, U_n$, e. g., by calling the $\mathsf{Execute}$ oracle. Next, $\mathcal{A}$ reveals $U_1$'s long term secret by querying $\mathsf{Corrupt}(U_1)$.
2. $\mathcal{A}$ initializes unused oracles of $U_3, \dots, U_n$ with $\mathsf{pid}_j^{s_j} = \{U_1, \dots, U_n\}$.
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 $M_1^2 := k_1 \| T_1 \| \sigma_1^2$.

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 models [BR93] and [CK01] go further in their definition of security than the model [BCPQ01]. The models know the notion of a matching session and 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. However, the models do not consider the influence corrupted principals now have on the uniqueness or integrity of the session identifier, which is as we have shown quite a relevant issue.

Granted, in the presence of malicious participants the adversary always learns the 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 or share the correct keys with unintended principals from the group (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 [BR93] and [CK01] we introduce an integrity property also for group key establishments to prevent sessions to mix up.

Besides the attacks we have seen in the last section, another example for two protocol executions that mix up is the *unknown key share attack* where a maliciously acting principal $U_1$ makes a protocol participant $U_2$ believe that $U_2$ established a session key with $U_1$, while indeed $U_2$ shares the key with $U_3 \neq U_1$. This kind of problem has first been pointed out by Diffie et al. [DOW92]. As explained in [BWJM97] the model of Bellare and Rogaway [BR93] prevents unknown key share attacks due to their notion of matching conversations, but unfortunately this is no longer true for the revised models [BPR00,BCPQ01] that base partnering on session identifiers, which are more suited for the group case.

**Definition 4.** *We say a correct group key establishment protocol fulfills* integrity *if with overwhelming probability all oracles of honest principals that have accepted with the same session identifier* $\mathsf{sid}_j^{s_j}$

– *hold identical session keys* $\mathsf{sk}_j^{s_j}$*, and*
– *hold a* $\mathsf{pid}_j^{s_j}$*-value encompassing the identities of all honest parties having accepted with session identifier* $\mathsf{sid}_j^{s_j}$*.*

*Strong entity authentication.* Entity authentication is a relevant issue for key establishment even excluding the possibility of corrupted participants. It is considered in the model [BR93] and in the models for password-based key establishment following [BPR00].

An approach to define entity authentication formally was made in [JG04]. For our security model dealing with group key establishment we rephrase this definition as follows.

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

*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. The notion of key agreement we use is motivated by the discussion in [MWW98] and imposes a quantitative restriction on the influence principals have on the derived session key.

**Definition 6.** *Let* $t \in \{1, \ldots, |\mathcal{P}|\}$, $\mathsf{P}$ *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* $\mathsf{P}$ *establishing the longterm keys is executed.*
2. *Having access to the public keys, the* Execute-, Send *and* Reveal-*oracle and being allowed up to* $t-1$ *calls to the* Corrupt *oracle,* $\mathcal{A}_1$ *outputs a quadruple* $(i, s_i, \chi_\kappa, a)$ *with state information* $a$ *and such that*
   - $U_i$ *is honest with* $\mathsf{used}_i^{s_i} = \mathsf{false}$;
   - $\chi_\kappa$ *is a boolean-valued ppt algorithm with* $\kappa := \{sk \in \mathcal{K} : \chi_\kappa(sk) = \mathsf{true}\}$ *such that* $|\kappa|$ *is polynomial 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* Execute-, 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* $\mathsf{P}$ *as being* $t$-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 $t$ malicious participants *if it is a correct* $(t+1)$*-contributory protocol in the sense of Definition 1 and Definition 6, secure in the sense of Definition 3, and assuming at most* $t$ *principals are dishonest, it offers integrity in the sense of Definition 4 and provides strong entity authentication to all participating oracles 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 1.** *Suppose that in the protocol of Katz and Yung described in Figure 1 all participants check for* $\prod_i X_i = 1$ *before accepting the key. Then, with session identifier* $\mathsf{sid}_i^{s_i} = \mathsf{pid}_i^{s_i}||t$ *(in this point diverging from [KY03]), we obtain a key establishment protocol secure in our model when we restrict to one* Corrupt *query.*

*Proof.* For correctness and security according to Definition 3 the proof of [KY03] applies. In the sequel, we assume only one participant in the key establishment is allowed to act maliciously.

*Integrity.* Let us suppose an adversary $\mathcal{A}$ aims at violating integrity as defined in Definition 4, however, she is only able to make a corrupt call to, say, principal $U_i$. The adversarial goal is to make two honest principals that accept a fixed session $\mathsf{sid}_j^{s_j}$ have either different session keys or hold an incorrect $\mathsf{pid}_j^{s_j}$ value. Though, once the session is fixed its $\mathsf{sid}_j^{s_j}$ contains $\mathsf{pid}_j^{s_j}$, shared thus to all honest principals.

Let us see why she cannot either violate integrity by forcing honest principals that have accepted to share different keys. Here are the concrete messages $\mathcal{A}$ can alter:

(i) the messages in the first round, especially since they are not authentified. Anyway sending an invalid message at this stage will result in blocking of a particular connection, and not have influence on accepting principals.

(ii) the value $z_i$ that $U_i$ broadcasts in the second round. The latter is actually only used by $U_{i-1}$ and $U_{i+1}$. Obviously, the protocol is still correct if $U_i$ sends different values $z_i$ and $z_i'$ to its neighbors, sending the same one is rather to save an exponentiation.

(iii) the value $X_i$ that $U_i$ broadcast in the third round. However the message is implicitly fixed by the values $X_j$ of the honest participants as $X_i = (\prod_{j \neq i} X_j)^{-1}$.

*Entity authentication.* The concatenation of the nonces $t$ computed in the first round is fresh as long as one honest oracle is involved. Since all participants compute the signature over the message $(1||z_i||\mathsf{pid}_i^{s_i}||t)$, it is assured that at the end of the second round all honest oracles have knowledge of the session identifier if it is chosen as $\mathsf{sid}_i^{s_i} = \mathsf{pid}_i^{s_i}||t$, and in particular they hold the same $\mathsf{pid}_i^{s_i}$.

*2-Contributory.* Note that the adversary cannot influence the key by a dedicated choice of one principal's "random" choices 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 the negligible fraction of the key space specified by the adversary is negligible; this is also true, even allowing exponent 0, for $n \geq 3$. $\square$

### 4.2 A Secure 2-Round Protocol

As shown above 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 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 $H(\mathsf{pid}_i^{s_i} \| k_1 \| \ldots \| k_{n-1} \| H(k_n))$, to certify that all of them will compute the same session key. Therewith, the attacks from Section 3.2 are effectively defeated.

**Round 1:**

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

**Broadcast** Each $U_i$ broadcasts $(M_i^1 \| \sigma_i^1)$.

**Check** Each $U_i$ checks all signatures $\sigma_j^1$ of incoming messages $(M_j^1 \| \sigma_j^1)$.

**Round 2:**

**Computation** Each $U_i$ computes $t_i^L = H(y_{i-1}^{x_i})$, $t_i^R = H(y_{i+1}^{x_i})$, $T_i = t_i^L \oplus t_i^R$ and $\mathsf{sid}_i^{s_i} = H(\mathsf{pid}_i^{s_i} \| k_1 \| \ldots \| k_{n-1} \| H(k_n))$, only $U_n$ computes additionally $k_n \oplus t_n^R$. The participants $U_1, \ldots, U_{n-1}$ set $M_i^2 = \mathsf{sid}_i^{s_i} \| T_i$, $U_n$ sets $M_n^2 = k_n \oplus t_n^R \| \mathsf{sid}_n^{s_n} \| T_n$ and each $U_i$ computes a signature $\sigma_i^2$ of $M_i^2$.

**Broadcast** Each $U_i$ broadcasts $(M_i^2 \| \sigma_i^2)$.

**Check** Firstly, each $U_i$ checks all signatures $\sigma_j^2$ of incoming messages. Then each $U_i$ checks if $T_1 \oplus \cdots \oplus T_n = 0$ and $\mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ $(j = 1, \ldots, n)$. Moreover, each $U_i$ $(i < n)$ checks the commitment $H(k_n)$ for $k_n$.

**Key computation:** Each participant $U_i$ computes the session key $\mathsf{sk}_i^{s_i} = H(\mathsf{pid}_i^{s_i} \| k_1 \| \ldots \| k_n)$.

**Fig. 3.** A secure group key agreement protocol.

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. [MWW98]). Transforming the above protocol accordingly is possible for the prize 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.

**Proposition 2.** *If the CDH assumption holds for $(\mathbb{G}, g)$ and $H(\cdot)$ is a random oracle, then the protocol in Figure 3 is a secure group key agreement in the sense of Definition 7.*

*Proof.* Let $q_{ex}, q_s$ and $q_{ro}$ be polynomial bounds for the number of the adversary's queries to the $\mathsf{Execute}$, the $\mathsf{Send}$ respectively the random oracle and $q_p$ a polynomial bound for the number of random oracle queries done by a principal.

Let $\mathsf{Forge}$ be the event that the adversary succeeds in forging an authenticated message $M_U||\sigma_U$ for one participant $U$ without having queried $\mathsf{Corrupt}(U)$. An adversary $\mathcal{A}$ that can reach $\mathsf{Forge}$ can be used for forging a signature for a given key. This key is assigned to one of the $n$ principals and thus $\mathcal{A}$ succeeds in the intended forgery with probability $\geq \frac{1}{n} \cdot P(\mathsf{Forge})$. Thus, using $\mathcal{A}$ as black box we can derive an attacker defeating the existential unforgeability of the underlying signature scheme $S$ with probability

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

Because of $\mathsf{Adv}_S^{\mathrm{cma}}$ being negligible by assumption, the event $\mathsf{Forge}$ occurs with negligible probability only. In summary, the events $\mathsf{Collision}$, $\mathsf{Repeat}$ and $\mathsf{Forge}$ all occur with a probability negligible in $k$.

Moreover, denote by $\mathsf{Collision}$ the event that the random oracle produces a collision. As the total number of random oracle queries is bounded by $n \cdot q_p + q_{ro}$, the probability that a collision of the random oracle occurs is

$$P(\mathsf{Collision}) \leq \frac{(n \cdot q_p + q_{ro})^2}{2^k}.$$

Finally, let $\mathsf{Repeat}$ be 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 $n \cdot q_{ex} + q_s$ used oracles that may have chosen a nonce $k_i$ and thus $\mathsf{Repeat}$ happens with a probability

$$P(\mathsf{Repeat}) \leq \frac{(n \cdot q_{ex} + q_s)^2}{2^k}.$$

*Security.* To prove the security according to Definition 3 we consider a sequence of games:

**Game 0:** In this game the protocol participants' oracles are faithfully simulated for the adversary, i. e., it behaves as in the real model.

**Game 1:** This game is aborted if the event $\mathsf{Forge}$ occurs. In this case the adversary loses.

Game 1 behaves like the real model if the events Forge does not occurs. Thus, for adversary $\mathcal{A}$'s advantage we have

$$\mathsf{Adv}_{\mathcal{A}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{Game1}} + P(\mathsf{Forge})$$

and it it sufficient to identify $\mathsf{Adv}_{\mathcal{A}}^{\text{Game1}}$ as being negligible. To this aim, we introduce

**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 oracle $\Pi_i^{s_i}$ and none of the neighbors $U_{i-1}$ or $U_{i+1}$ 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 consistent the same values have to be used in the neighbored instances subsequently. Then the simulator answers with the message including the value $T_i = t_i^L \oplus t_i^R$. If the adversary queries the random oracle with $y_{i-1}^{x_i} = y_i^{x_{i-1}}$, the game is aborted and the adversary loses.

By the random oracle assumption, the adversary can only detect the difference by querying the random oracle for determining these hash values. We denote by Random the event that $\mathcal{A}$ queries the random oracle with $y_{i-1}^{x_i} = y_i^{x_{i-1}}$ and it holds that $U_i$ and $U_{i-1}$ are uncorrupted. Since the event Forge is already excluded the messages from the first round were generated by the oracles and the exponents $x_i$ and $x_{i-1}$ cannot be known to the adversary.

To know the value $y_{i-1}^{x_i} = y_i^{x_{i-1}}$ shared between two oracles $\mathcal{A}$ needs to solve a CDH instance to learn $(y_{i-1}^{x_i}) = (y_i^{x_{i-1}})$. More precisely, using $\mathcal{A}$ as blackbox and guessing at random the oracles to which the CDH instance is assigned, from $\mathcal{A}$ we can derive an attacker against the CDH problem with success probability

$$\mathsf{Succ}_{(\mathbb{G},g)}^{\text{CDH}} \geq \frac{1}{(q_s + n \cdot q_{ex})^2} \cdot P(\mathsf{Random}).$$

Thus, from the CDH assumption we conclude that $P(\mathsf{Random})$ is negligible. Further on, the adversary has success in Game 2 exactly in the cases in which he succeeds in Game 1, unless the event Random occurs:

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Game 1}} \leq \mathsf{Adv}_{\mathcal{A}}^{\text{Game 2}} + P(\mathsf{Random}).$$

In the adversary's Test-session no protocol participant is allowed to be corrupted in Round 1 (see Definition 3). Thereby, all oracles use random values in Round 2 and no information about $k_n$ is transmitted. So the adversary in Game 2 can do no better than guessing the session key and has no advantage. Putting the probabilities together we recognize the adversary's advantage in the real model as negligible:

$$\mathsf{Adv}_{\mathcal{A}} \leq P(\mathsf{Forge}) + P(\mathsf{Random}).$$

*Integrity.* Let NoIntegrity be the event that some oracle 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 oracles $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ accept ($\mathsf{acc}_i^s = \mathsf{true}$)

with a matching session identifier $\mathsf{sid} := \mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$. The session identifier $\mathsf{sid}$ is unique if uncorrupted principals contributed fresh nonces $k_i$ (unless $\mathsf{Repeat}$) and the random oracle is collision free (unless $\mathsf{Collision}$). Moreover the messages of uncorrupted principals cannot be forged (unless $\mathsf{Forge}$) by the adversary. Thus $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ must have received each other's message $\mathsf{sid}_i^{s_i} \| T_i \| \sigma_i^2$ respectively $\mathsf{sid}_j^{s_j} \| T_j \| \sigma_j^2$, where necessarily $\mathsf{sid} := \mathsf{sid}_i^{s_i} = \mathsf{sid}_j^{s_j}$ matched due to the check phase.

The construction of $\mathsf{sid}$ assures that $\Pi_i^{s_i}$ and $\Pi_j^{s_j}$ hold the same $\mathsf{pid} := \mathsf{pid}_i^{s_i} = \mathsf{pid}_j^{s_j}$ (obtained in the respective oracle's initialization) and know the same values $k_1, \ldots, k_{n-1}$ 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 $\mathsf{sk}_i^{s_i} = \mathsf{sk}_j^{s_j}$. Thus, putting things together we obtain the desired negligible upper bound

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

*Entity authentication.* Let $\mathsf{EntAuthFail}$ 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 $\mathsf{sid}$ from all principals $U \in \mathsf{pid}_i^{s_i}$ (unless $\mathsf{Forge}$). As above, in absence of $\mathsf{Collision}$, $\mathsf{Repeat}$ and $\mathsf{Forge}$, the session identifier is unique and the message cannot be replayed from a past session. Thus every honest partner holds the same session identifier $\mathsf{sid}$ and for the reasons stated above also the partner identifiers $\mathsf{pid}_i^{s_i}$ and $\mathsf{pid}_j^{s_j}$ match. Therewith entity authentication is violated with a probability

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

*Key Agreement.* The values relevant for deriving the session key are only the values $k_i$ that participant $U_i$ chooses in the first round. An honest participant chooses a fresh value with probability $1 - P(\mathsf{Repeat})$. Thus a corrupted participant $U_n$, who can know the inputs of $U_1, \ldots, U_{n-1}$ can only choose between a polynomial set of keys, bounded by the number of random oracle queries $q_{ro}$.

Finally, correctness of the protocol in Figure 3 is straightforward, and hence the proposition follows.

□

## 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 [KY03] offers security guarantees against a single malicious participant "for free", whereas a proposal of Kim, Lee and Lee [KLL04] fails to do so. However, as shown in the last section, security against malicious participants is achievable in two rounds. Without sacrificing efficiency, the latter protocol can be modified to offer rather strong security guarantees even in the presence of malicious participants.

## References

[BCK98] 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.

[BCP01] 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.

[BCPQ01] 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.

[BD95] 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.

[BM04] Colin Boyd and Anish Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2004.

[BN03] 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.

[BPR00] 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.

[BR93] 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.

[BR95] 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.

[BWJM97] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, pages 30–45. Springer, 1997.

[CBH05] 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.

[CBHM05] 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.

[CK01] 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.

[CS04] Christian Cachin and Reto Strobl. 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.

[DOW92] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[HMQS03] 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.

[JG04] 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.

[KLL04] 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.

[KS05] Jonathan Katz and Ji Sun Shin. Modeling Insider Attacks on Group Key-Exchange Protocols, 2005. 12th ACM Conference on Computer and Communications Security.

[KY03] 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.

[MWW98] Chris J. Mitchell, Mike Ward, and Piers Wilson. Key control in key agreement protocols. *IEE Electronics Letters*, 34(10):980–981, 1998.

[Sho99] 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`.

[Ste02] 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/Stei_02.thesis-final.pdf`.

[Tze00] 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.