with indistinguishability of keys computed in *earlier* sessions assuming that users can be corrupted at some later stage, and recently it has been updated to include outsider key compromise impersonation attacks [19]. Another requirement called *Mutual Authentication (MA)* has been formalized and studied for the honest users setting in [8, 9].

Insider Security In 2005, Katz and Shin formalized[4] the notion of security against insider impersonation attacks and key agreement in [23], and Choo et al. defined resistance against unknown-key share attacks [18]. Another goal — *contributiveness* — has been identified in [3, 10, 12, 19] and is also related to non-malleability [21] or key control [31, 34]: briefly speaking, it prevents the adversary from "fixing" the value of the group key computed by honest users; this property states the main difference between key exchange and key transport. In particular, it also prevents key-replication attacks [28].

Robustness Executions of GKE protocols that do not provide robustness are aborted if some deviation from the given protocol specification is detected by the users, e.g. when users are not able to send or receive messages or if some necessary verification steps fail. Amir et al. [1] were the first to consider robustness in GKE protocols; they merged the non-robust GKE protocol by Steiner et al. [38] with an underlying *group communication system (GCS)* [17]. However their protocol must be restarted in case of failures. Assuming authenticated channels, Cachin and Strobl [14] proposed and formally proved (in the framework of Reactive Simulatability [33]) an asynchronous GKE protocol by combining the GKE protocol by Burmester and Desmedt [5] with an additional k*-resilient consensus protocol* [13, 16]. Their protocol can tolerate only up to $n-2k$ corruptions to remain forward secure (which is an upper-bound for the asynchronous setting). Desmedt et al. [21] considered an unauthenticated reliable broadcast setting and designed a provably secure scheme immune to outsider and insider attacks based on *verifiable secret sharing (VSS)*. They also explained how to use the authentication compiler by Katz and Yung [24] to sort out invalid messages and tolerate failures. Jarecki et al. [22] followed by Kim and Tsudik [27] used reliable broadcast/multicast setting to design robust GKE protocols proven secure against outsider adversaries. They also defined *full-robustness*: since a GKE protocol requires at least two users, the optimal criterion for robustness is the ability to tolerate up to $n-2$ (out of n) failed users. We remark that solutions proposed in [14, 21] are not fully robust.

Corruptions and Opening Attacks Beside formal definitions of outsider and insider security, existing models for GKE protocols differ in the adversarial abilities to corrupt users: *weak corruptions* [9, 21, 24] allow the adversary to obtain users' long-lived keys, but not their internal states, whereas *strong corruptions* [8, 10, 12, 23, 29, 36, 38] reveal both secret types at the same time. The latter gain more and more on attention due to the significant advances in the field of malware and side-channel attacks used to recover information stored locally within hardware and software. Corruptions allow to model the requirement of *(strong) forward secrecy* [8, 9, 11, 12] whose goal is to ensure AKE-security "in the future". Strong corruptions have been recently refined with so-called *opening attacks* [12] that provide higher flexibility: they allow the adversary to get users' ephemeral secrets without necessarily obtaining their long-lived keys. The formal advantage of this refinement is that it can exclude subsequent impersonation attacks on the "opened" users who are then treated as *honest* (rather than *corrupted*). As an illustrative example, AKE-security can be extended to capture the leakage of users' internal states prior to the protocol execution; thus, GKE protocols which pre-compute their ephemeral secrets off-line (for better efficiency) may become insecure.

[4] First security definitions for insider attacks in non-robust GKE protocols were formalized in [23]. Later, [10, 12] merged these definitions as part of MA-security and formalized contributiveness. Then, [19] updated MA-security with insider key compromise impersonation.

1.2 Organization of the Paper

In Section 2 we present our *tree replication* technique. Then in Sections 3 and 4, we present the protocols R-TDH1 and IR-TDH1, respectively. The security model is described in Section 5; interestingly, it can be viewed as an extension of [12, 19] towards consideration of (full) robustness, and at the same time as an extension of [21, 22] towards consideration of strong corruptions and opening attacks. In Section 6 we compare security and efficiency of R-TDH1 and IR-TDH1 with some earlier robust GKE protocols.

2 The Tree-Diffie-Hellman Protocol and Our Tree Replication Technique

First, we recall the basic steps of the non-robust protocol TDH1 (see [12, 26] for more details). Then, we introduce at a high-level our tree replication technique that achieves full robustness.

2.1 Overview of Basic TDH1

Preliminaries The protocol makes use of a *linear* binary tree T_n: a full binary tree with one leaf at each level, except for the deepest one with two leaves, i.e. each node in T_n has a *label* $\langle l, v\rangle$, where $l \in [0, n-1]$ denotes its level and $v \in [0, 1]$ its *position* within that level. For each node $\langle l, v\rangle$, there are two associated values: a secret value, denoted $x_{l,v}$, and a public one, denoted $y_{l,v}$ and computed as $y_{l,v} = g^{x_{l,v}}$. Moreover, each secret value associated to an internal node is the Diffie-Hellman function of the public values associated to its children. In other words, for any l we have:

$$x_{l,0} = \mathsf{DH}(y_{l+1,0}, y_{l+1,1}) = g^{x_{l+1,0}x_{l+1,1}}$$

In order to be able to "chain" such operations, all operations are performed in a cyclic group $\mathbb{G}$ with generator g in which the classical Decisional Diffie-Hellman (DDH) assumption is assumed to be hard and for which there exists an *efficient*, bijective mapping from $\mathbb{G}$ to $\mathbb{Z}_{|\mathbb{G}|}$ (which is not the discrete logarithm!); this bijection is used to consider multiple-decker exponentiations: the result of an exponentiation (an element of $\mathbb{G}$) can be in turn re-interpreted as an exponent in $\mathbb{Z}_{|\mathbb{G}|}$. A suitable group $\mathbb{G}$ of prime order q generated by a quadratic residue g modulo a large safe prime number $p = 2q + 1$ has been described in [12, 25, 26]. In this group if some exponent x is uniform and random in $\mathbb{Z}_q$ then so is g^x in $\mathbb{G}$ and $\mathbb{G} = \mathbb{Z}_q$ (as sets).

Protocol Steps The protocol is based on the following (intuitive) trick: users are associated to the leaf nodes. Each user U_i knows the secret value associated to its own node, and can reveal the corresponding public value y_i. Using these values, it is easy to check that the users associated to the deepest leaves have enough information to compute all values (both secret and public) associated to the internal nodes. And once all public values are available (that is, including public values associated to internal nodes), it is clear that every user can inductively compute the secret value associated to the root.

- Round 1. User U_i associated to leaf $\langle l, v\rangle$ chooses a secret $x_{l,v}$ and broadcasts $y_{l,v} := g^{x_{l,v}}$;
- Round 2. The goal of the protocol is to let each U_i compute the secret value $x_{0,0}$ associated to the root. Therefore, U_1 assigned to $\langle n-1, 0\rangle$ computes a set X_1 of secret values $x_{l,0}$ in its path up to the root $\langle 0, 0\rangle$. Each $x_{l,0}$ can be seen as the output of the Diffie-Hellman function of $y_{l+1,0}$ and $y_{l+1,1}$. Note that for each internal node $\langle l, 0\rangle$ user U_1 knows the public value $y_{l+1,1}$ broadcasted by some other user and the secret value $x_{l+1,0}$ by induction at level $l+1$. Having computed the set X_1 user U_1 broadcasts each $y_{l,0} = g^{x_{l,0}}$. We emphasize, however, that $y_{0,0}$ is never made public —it is not used in the protocol;
- Group Key Derivation. Once user U_1 finishes, all other users $U_{i\neq 1}$ are also able to compute the secret values $x_{l,0}$ in their paths up to the root. Hence, every user finally learns $x_{0,0}$ and uses it to derive the group key.

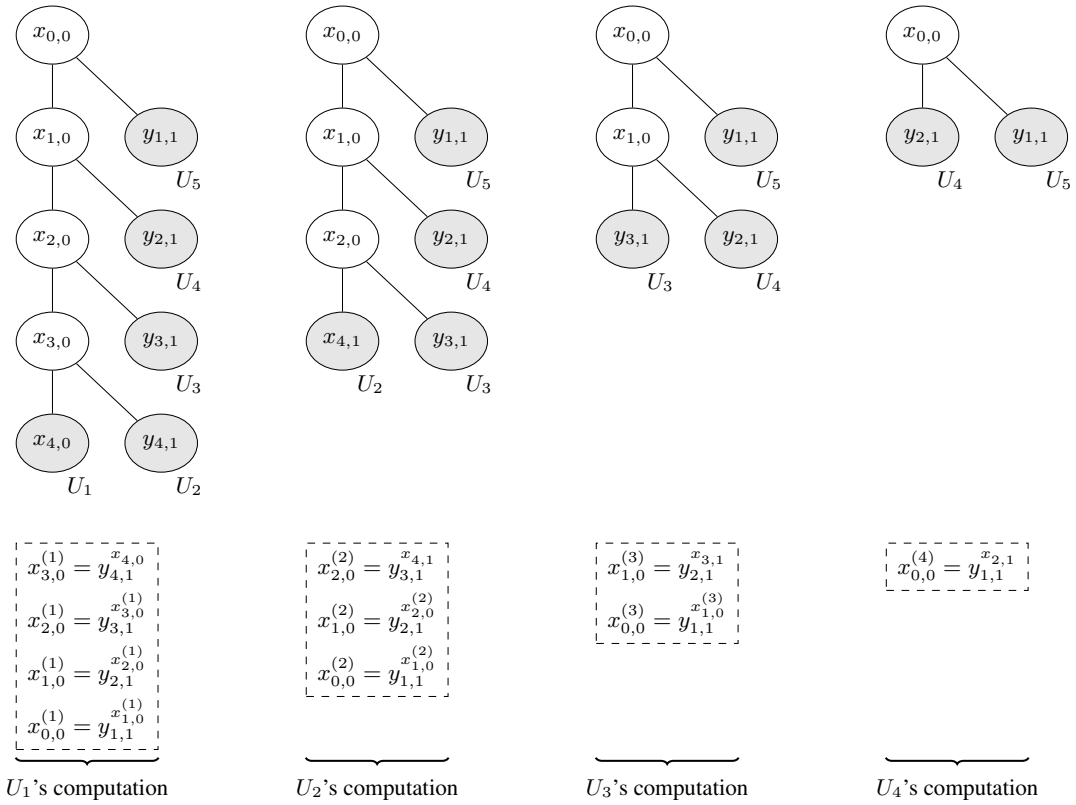

Fig. 1. Tree Replication Technique. Representation of each user's computation in the second protocol round. One of the trees will be used commonly by all active users to derive the group key.

2.2 The Tree Replication Technique

The original TDH1 protocol is not robust, in particular if the second round message is not delivered then all parties have to abort. In order to achieve robustness, we enhance TDH1 such that, even if some users fail (they halt and/or are not able to continue), the remaining users are still able to compute a common tree structure and to compute a common root secret. This feature is achieved through what we call the *tree replication* technique.

Intuitively, it means that every user is going to compute its own key tree structure, and act as if it would be in the position of U_1. Then, after some users failed, the "deepest" common structure will be used by all users to compute the root secret. At a high level the modification is as follows.

- Round 1. Each user U_i chooses its secret exponent $x_i \in_R \mathbb{G}$ and broadcasts its public value $y_i := g^{x_i}$. After this round all active users (i.e. those who do not fail) will receive sent public values of other active users. Based on this information, they are assigned to tree leaves;
- Round 2. Each user U_i computes its own set X_i as visualized in Figure 1. Each set X_i is composed of secret values $x_{l,0}^{(i)}$, where the superscript "(i)" indicates a quantity which is computed by U_i only; when we write $x_{l,0}^{(i)}$ we mean that user U_i computes his own value of the variable named "$x_{l,0}$". Users U_i also broadcasts the public values $y_{l,0}^{(i)}$ corresponding to the exponents $x_{l,0}^{(i)}$ in X_i (and with the obvious exclusion of $y_{0,0}^{(i)}$);

– Group Key Derivation. For the computation of the secret root, all users choose the message broadcasted by the lowest-index alive user. Of importance is that all users who are still active choose the same broadcast message and compute the same secret value for the root $\langle 0, 0 \rangle$.

Remarks Unlike in TDH1, the lowest-indexed user is not necessarily U_1: it can be U_2 if U_1 has failed and so on. Note also that X_n and X_{n-1} are empty sets: if U_{n-1} and U_n are the only remaining users, the protocol reduces to two-party Diffie-Hellman. Yet, U_n and U_{n-1} must still broadcast their "liveness" messages in the second round.

3 A Fully Robust Protocol with Strong *Outsider* Security: R-TDH1

Here we specify R-TDH1 that achieves full robustness while preserving the constant number of rounds and strong outsider security of TDH1 [12]. In what follows we assume that each user U_i has a long-lived key $LL_i = (sk_i, pk_i)$ generated by $\Sigma.\mathtt{Gen}(1^\kappa)$, where $\Sigma = (\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ is an existentially unforgeable signature scheme. By $F := \left\{\left\{f_k\right\}_{k\in\{0,1\}^\kappa}\right\}_{\kappa\in\mathbb{N}}$ we denote a pseudo-random function ensemble. Our descriptions are provided from the perspective of one session with the initial set of participants $(U_1, \ldots, U_n)$.

Formal Description of R-TDH1 We assume that each U_i is initialized with a *partner id* $\mathtt{pid}_i$ encompassing the identities of all users participating in that session. In the beginning of each round each U_i will update own $\mathtt{pid}_i$ by removing users that are no longer active based on the messages it received. We assume that if at some stage $\mathtt{pid}_i = \{U_i\}$ then U_i erases every secret information from it internal state $\mathtt{state}_i$ and terminates without accepting.

Round 1. Each user U_i does the following:
- $r_i \in_R \{0,1\}^\kappa$; broadcast $U_i|1|r_i$.

Round 2. Each user U_i does the following:
- Remove from $\mathtt{pid}_i$ every U_j with missing messages;
- $\mathtt{nonces}_i \leftarrow r_1|\ldots|r_{n'}$;
- $x_i \in_R \mathbb{G}$; $y_i \leftarrow g^{x_i}$;
- $\sigma_i \leftarrow \Sigma.\mathtt{Sign}(sk_i, 2|y_i|\mathtt{nonces}_i|\mathtt{pid}_i)$;
- Broadcast $U_i|2|y_i|\sigma_i$.

Round 3. Each user U_i does the following:
- Remove from $\mathtt{pid}_i$ every U_j with missing or invalid messages;
- Remove nonces of failed oracles from $\mathtt{nonces}_i$;
- Assigned remaining oracles to the leaves of T_n, where $n = |\mathtt{pid}_i|$;
- $Y \leftarrow \{y_j\}_{1\le j\le n}$;
- $x^{(i)}_{n-i,0} \leftarrow x_i$ (renaming);
- For $l = n - i - 1$ downto 0, iteratively compute a set X_i made of values: $x^{(i)}_{l,0} = y^{x^{(i)}_{l+1,0}}_{l+1,1}$;
- For $l = n - i - 1$ downto 1, compute a set $\hat{Y}_i$ made of values: $y^{(i)}_{l,0} = g^{x^{(i)}_{l,0}}$;
- $\sigma_i \leftarrow \Sigma.\mathtt{Sign}(sk_i, 3|M|Y|\mathtt{nonces}_i|\mathtt{pid}_i)$ where $M = \hat{Y}_i$ if $i < n - 1$ and $M =$'$\mathtt{alive}$' if $i \ge n - 1$;
- Broadcast $U_i|3|M|\sigma_i$.

Group Key Derivation. Each user U_i does the following:
- Remove from $\mathtt{pid}_i$ every U_j with missing messages or invalid signatures;
- Update $\mathtt{nonces}_i$ by removing the nonces of failed oracles;
- Determine the *lowest-indexed* oracle U_γ;

- $x_{n-i,0} \leftarrow \begin{cases} y_{\gamma}^{x_i} & \text{if } i = \gamma + 1 \\ y_{n-i+1,0}^{(\gamma)\ x_i} & \text{if } i > \gamma + 1 \end{cases}$
- For $l = n - i - 1$ downto 0: $x_{l,0} = y_{l+1,1}^{x_{l+1,0}}$
- $k_i \leftarrow f_{x_{0,0}^{(\gamma)}}(v)$;
- Erase every ephemeral secret information from $\texttt{state}_i$ and accept with k_i.

4 Securing R-TDH1 against Strong Insider Attacks

In this section we describe IR-TDH1, an extension of R-TDH1 which provides security against outsider *and* insider attacks. To do so, we use a special NIZK proof for the *equality of the double discrete logarithm and a (single) discrete logarithm* from [2, 37].

4.1 NIZK Proof $\texttt{Lg}^2\texttt{EqLg}$

Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a cryptographic hash function for some ℓ. Let $g, y, \tilde{y}_1, \tilde{y}_2$ be public elements of $\mathbb{G}$. A Non-Interactive Zero-Knowledge Proof for the statement $\log_g(\tilde{y}_1) = \log_y(\log_g(\tilde{y}_2))$ is denoted $\texttt{Lg}^2\texttt{EqLg}[(x) : \tilde{y}_1 = g^x \wedge \tilde{y}_2 = g^{y^x}]$ and can be constructed as follows using the witness x: for $i \in [1, \ell]$, pick at random α_i in $\mathbb{G}$ and compute $t_{1,i} := g^{\alpha_i}$ and $t_{2,i} := g^{y^{\alpha_i}}$; then output $z := (c, s_1, \ldots, s_\ell)$ with

$$c := H(g|y|\tilde{y}_1|\tilde{y}_2|t_{1,1}|\ldots|t_{1,\ell}|t_{2,1}|\ldots|t_{2,\ell})$$
$$s_i := \begin{cases} \alpha_i & \text{if } c[i] = 0 \quad (c[i] \text{ is the } i\text{-th bit of } c) \\ \alpha_i - x & \text{otherwise.} \end{cases}$$

To verify z one simply checks whether $c \stackrel{?}{=} H(g|y|\tilde{y}_1|\tilde{y}_2|\bar{t}_{1,1}|\ldots|\bar{t}_{1,\ell}|\bar{t}_{2,1}|\ldots|\bar{t}_{2,\ell})$ where

$$\bar{t}_{1,i} := \begin{cases} g^{s_i} & \text{if } c[i] = 0 \\ \tilde{y}_1 g^{s_i} & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{t}_{2,i} := \begin{cases} g^{y^{s_i}} & \text{if } c[i] = 0 \\ \tilde{y}_2^{y^{s_i}} & \text{otherwise.} \end{cases}$$

Using the Random Oracle Model (ROM) [4] one can show that $\texttt{Lg}^2\texttt{EqLg}$ is secure; we denote by $\mathsf{Adv}^{\mathtt{zk}}_{\texttt{Lg}^2\texttt{EqLg}}(\kappa)$ the maximum advantage of distinguishing a real proof from a simulated one, and by $\mathsf{Succ}^{\mathtt{snd}}_{\texttt{Lg}^2\texttt{EqLg}}(\kappa)$ the probability of computing a valid proof for a false statement (*soundness*).

4.2 Description of IR-TDH1

Our protocol is an extension of R-TDH1. We add NIZK proofs in order to prevent corrupted users from sending bad values. This increases the costs of the protocol. Briefly speaking, the protocol is modified in the third round, as follows. In addition to computing X_i (a set of values), each user computes a set Z_i of NIZK proofs $\{z_l^{(i)}\}_{n-i-1 \geq l \geq 1}$:

- for $1 \leq l \leq n - i - 2$, proof $z_l^{(i)}$ proves that

$$x_{l+1,0}^{(i)} = \log_g\left(y_{l+1,0}^{(i)}\right) = \log_{y_{l+1,1}}\left(\log_g\left(y_{l,0}^{(i)}\right)\right)$$

- for $l = n - i - 1$, proof $z_l^{(i)}$ proves that

$$\begin{cases} x_{n-1,0} = \log_g(y_{n-1,0}) = \log_{y_{n-1,1}}\left(\log_g\left(y_{n-2,0}^{(1)}\right)\right) & \text{if } i = 1 \\ x_{n-i+1,1} = \log_g(y_{n-i+1,1}) = \log_{y_{n-i,1}}\left(\log_g\left(y_{n-i-1,0}^{(i)}\right)\right) & \text{if } i > 1 \end{cases}$$

We note that Z_{n-1} and Z_n computed by U_{n-1} and U_n are empty. The remaining of the protocol is identical to R-TDH1, however, in the Random Oracle Model, the key derivation is simplified as: $k_i := H'(x_{0,0}^{(\gamma)}|\texttt{nonces}_i|\texttt{pid}_i)$, where $x_{0,0}^{(\gamma)}$ is the common secret computed by active users.

5 Security of R-TDH1 and IR-TDH1

5.1 Security Model

Protocol Participants and Execution Model In order to capture multiple sessions, we model each user U through different *instance oracles* Π_U^s. Then, the *session identifier* is of the form $\texttt{sid} := U_{i_1}|s_{i_1}|\ldots|U_{i_n}|s_{i_n}$. We say that Π_U^s and $\Pi_{U'}^t$ are *partners* if there exists $\texttt{sid}$ containing $U|s$ and $U'|t$ as substrings. For each Π_U^s we also define its *partner id* $\texttt{pid}_U^s$ and the *internal state* $\texttt{state}_U^s$ as mentioned in the description of the protocols. Once invoked Π_U^s turns into the *processing* stage where it communicates and updates $\texttt{pid}_U^s$ by removing user identities of oracles that it treats as failed. As long as $|\texttt{pid}_U^s| > 1$, the oracle is *active* (it continues the execution); at some point, it *accepts* with a *session key* k_U^s and terminates successfully. Otherwise ($|\texttt{pid}_U^s| \leq 1$ or no acceptance), it terminates with a failure.

Communication For the security of R-TDH1 and IR-TDH1 we consider a reliable broadcast channel without authentication and any ordering guarantees; this is similar to [21] and less restrictive than [22]. The protocol execution is organized in rounds, which are delimited by a local timer δ (within a round, events are asynchronous): that is participants expect to receive round messages before their timer expires. At the beginning of each round, the *adversary* $\mathcal{A}$ learns each round message to be broadcast. It can then block (refuse to deliver) some of these messages. Additionally, it can inject its own messages. Thus at some point, $\mathcal{A}$ will have a set of messages which it "puts" on the broadcast channel before the timer expires. We model network failures by considering user U as *disconnected* if no expected message containing U as sender's identity is put on the channel. Reliability of the broadcast channel means that all messages put on it are delivered to *all participants that are still connected* in that round. The actual delivery order is determined by the adversary. At the end of the round, each oracle Π_U^s updates its partner id based on the previously received messages: users from whom no message has been delivered are removed. Reliability of the channel implies consistency of updated partner ids.

Adversarial Queries The adversary $\mathcal{A}$ is modeled as a PPT (*probabilistic polynomial-time Turing machine*), it is assumed to mount its attacks through the following queries:

Initialize$(\mathcal{S})$: for each user in the set $\mathcal{S}$ a new oracle Π_U^s is initialized and the resulting session id $\texttt{sid}$ is given to $\mathcal{A}$.

Invoke$(\texttt{sid}, \mathcal{S}')$, assuming that $\texttt{sid}$ is a valid session id and $\mathcal{S}'$ is a set of initialized oracles ($\mathcal{S}' \subseteq \mathcal{S}$ where $\mathcal{S}$ led to the construction of $\texttt{sid}$). In response, for each $U \in \mathcal{S}'$ the oracle Π_U^s turns into the processing stage and learns $\texttt{pid}_U^s = \mathcal{S}$. Then, $\mathcal{A}$ is given the first protocol message m computed by each Π_U^s and the round timer δ is started. The separation between *Initialize* and *Invoke* allows opening attacks against honest oracles prior to the first protocol round. It also allows $\mathcal{A}$ to decide which of these oracles should proceed with the execution. We require that the *Invoke* query can be invoked only once with a given argument.

Broadcast$(\texttt{sid}, m)$: In this query the message m is supposed to contain the identity of its sender U. The *Broadcast* queries for the current round are collected in the order of their occurrence; at the end of the round, message m is delivered to all connected oracles Π_U^s in $\texttt{sid}$ (i.e., oracles such that U is part of m for some collected *Broadcast*$(\texttt{sid}, m)$ query). $\mathcal{A}$ can also provide several messages m that include the same sender U. It is the task of the protocol to determine which of these messages should be processed or dropped. At the end of the round, $\mathcal{A}$ receives messages to be sent by the connected oracles in the next round, and each oracle Π_U^s updates its set $\texttt{pid}_U^s$ according to the protocol specification.

Corrupt(U): $\mathcal{A}$ obtains LL_U. This allows impersonation attacks, in which $\mathcal{A}$ can "talk" on the network pretending to be U.

AddUser(U, Λ), where Λ contains the registration information and a long-lived key LL_U: in response, a new user with that long-lived key is added to $\mathcal{U}$. This query (which is missing in [12]) allows $\mathcal{A}$ to register new users whose behavior it will fully control.

RevealState(Π_U^s): $\mathcal{A}$ obtains ephemeral secrets stored in $\texttt{state}_U^s$ (which may also be empty, if erased). This query models opening attacks [12].

RevealKey(Π_U^s): $\mathcal{A}$ obtains k_U^s (only if Π_U^s has already accepted).

Terminology We say U is *corrupted* or *malicious* if LL_U is known to $\mathcal{A}$, either via *Corrupt*(U) or *AddUser*(U, Λ); if no such queries have been asked then U is *honest*. This terminology also refers to the oracles of U. However, an opening attack is not sufficient to make Π_U^s malicious.

Definition of Robust Group Key Exchange We can now formally specify what a (fully) robust GKE protocol is.

Definition 1 (Robust GKE Protocol). *A* robust group key exchange (RGKE) *protocol* P *consists of a key generation algorithm* KeyGen*, and a protocol* Setup*:*

- P.KeyGen(1^κ)*: On input a security parameter* 1^κ*, each user is given* LL_U*.*
- P.Setup($\mathcal{S}$)*: On input a set* $\mathcal{S} \subseteq \mathcal{U}$ *a new oracle* Π_U^s *is created for each* $U \in \mathcal{S}$*. A probabilistic interactive protocol is executed between these oracles such that at the end all active oracles (those that have not failed and are still connected) accept with the session group key and terminate.*

P *is* correct *if all active oracles that are honest accept the same session group key.* P *is* fully robust *if it can tolerate all oracles dishonest except two.*

Strong Outsider Security The *outsider* security of GKE protocols can be expressed through AKE-security. In this section we define its strong version revising the one from [12] to address robustness.

We use the classical query *Test*(Π_U^s) to model AKE-security: in response, a bit b is privately flipped and $\mathcal{A}$ is given k_U^s if $b = 1$ or a random string if $b = 0$. The difficulty is in defining how to use this query; the notion of freshness aims at excluding trivial and meaningless attacks. We provide the following four conditions: condition (a) excludes prevents $\mathcal{A}$ from introducing new users; condition (b) allows $\mathcal{A}$ to corrupt some user of the attacked session but $\mathcal{A}$ must remain passive on behalf of that user's oracle until the session is complete, this models outsider key compromise impersonation attacks [19] for robust protocols; condition (c) follows from [12] and allows $\mathcal{A}$ to inspect internal states of participants before and after the attacked session but not during it; condition (d) prevents $\mathcal{A}$ from obtaining the key directly via the *RevealKey* query.

Definition 2. (Oracle Freshness for RGKE) *In a session* sid *of* P *an oracle* Π_U^s *that has accepted is* fresh *if* ***all*** *of the following holds:*

(a) no U' *included in* sid *has been added by* $\mathcal{A}$ *via a corresponding* AddUser *query,*
(b) if some U' *(incl.* $U' = U$*) from* sid *has been asked* Corrupt(U') *prior to the acceptance of* Π_U^s *then any message* m *with sender's identity* U' *asked via a* Broadcast(sid, m) *query must have been produced by the corresponding oracle* $\Pi_{U'}^t$ *partnered with* Π_U^s*,*
(c) neither Π_U^s *nor any of its partners has been asked for a query* RevealState *before they terminated,*
(d) neither Π_U^s *nor any of its partners is asked for a query* RevealKey *after having accepted and terminated.*

Definition 3. (Strong AKE-Security for RGKE) *Let* P *be a correct RGKE protocol and* b *a uniformly chosen bit. Consider an adversary* $\mathcal{A}$ *against the AKE-security of* P*. We define the adversarial game* $\mathsf{Game}^{\mathsf{ake}-b}_{\mathcal{A},\mathrm{P}}(\kappa)$ *as follows:*

- $\mathcal{A}$ *interacts via queries;*
- *at some point* $\mathcal{A}$ *asks a* Test *query to an oracle* Π_U^s *which is (and remains)* ***fresh****;*

– *$\mathcal{A}$ continues interacting via queries;*
– *when $\mathcal{A}$ terminates, it outputs a bit, which is set as the output of the game.*

$$\textit{We define:} \qquad \mathsf{Adv}^{\mathsf{ake}}_{\mathcal{A},\mathtt{P}}(\kappa) := \left|2\Pr[\mathsf{Game}^{\mathsf{ake}-b}_{\mathcal{A},\mathtt{P}}(\kappa) = b] - 1\right|$$

and denote with $\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{P}}(\kappa)$ the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that a RGKE protocol P *provides* strong AKE-security *if this advantage is negligible.*

Strong Insider Security We now revisit the strong *insider* security definitions, that is MA-security and contributiveness, from [12] to address robustness.

In the next definition, condition (a) models robustness since it requires that every honest, non-failed participant accepts, provided there exists other participants that have not failed as well. In condition (b) we model mutual authentication in the sense that no user accepts the group key until it is assured of the active participation of the other users; this takes into account insider key compromise impersonation attacks [19] as $\mathcal{A}$ can obtain the long-lived key of a user, as long as it remains passive with respect to that user's oracle. Finally, condition (c) models key confirmation and requires that session group keys accepted by any two participants are identical.

Definition 4. (Strong MA-Security for RGKE) *Let* P *a correct RGKE protocol and $\mathcal{A}$ an adversary who is allowed to query* Initialize, Invoke, Broadcast, AddUser, Corrupt, RevealKey *and* RevealState. *We denote this interaction as $\mathsf{Game}^{\mathsf{ma}}_{\mathcal{A},\mathtt{P}}(\kappa)$. We say that $\mathcal{A}$* wins *in some session* sid *if at the end of that session* ***one*** *of the following conditions is satisfied:*

(a) there is an honest oracle Π^s_U (with $U|s$ part of sid*) which terminated without having accepted some key but for which other partners exist (i.e., $|\mathtt{pid}^s_U| > 1$),*
(b) there are two partnered oracles Π^s_U and $\Pi^t_{U'}$ such that Π^s_U has accepted and $U' \in \mathtt{pid}^s_U$ is uncorrupted but $\Pi^t_{U'}$ has not been invoked via Invoke$(\mathtt{sid}, \cdot)$,
(c) there are two honest partnered oracles Π^s_U and $\Pi^t_{U'}$ which have accepted and $k^s_U \neq k^t_{U'}$.

The maximum probability of this event is denoted $\mathsf{Succ}^{\mathsf{ma}}_{\mathtt{P}}(\kappa)$; we say that a RGKE protocol P *provides* strong MA-security *if this probability is negligible.*

The following requirement of strong contributiveness resists key control attacks by which a malicious subset of (at most $n-1$) users aims to predetermine the resulting value of the group key [34]. This is in contrast to the *non-malleability* property [21], which ensures uniform distribution of group keys in the presence of malicious participants but in a weaker model in which no opening attacks exist (as discussed in [12]).

Definition 5. (Strong Contributiveness for RGKE) *Let* P *be a correct RGKE protocol and $\mathcal{A}$ an adversary operating in two stages (*prepare *and* attack*) and having access to the queries* Initialize, Invoke, Broadcast, AddUser, Corrupt, RevealKey *and* RevealState. *We define the following game $\mathsf{Game}^{\mathsf{con}}_{\mathcal{A},\mathtt{P}}(\kappa)$:*

– *$\mathcal{A}$(*prepare*) interacts via queries and outputs some $\tilde{k} \in \{0,1\}^\kappa$, and some state information ζ;*
– *A set Ψ is built, consisting of all session ids* sid *for which a query* Invoke$(\mathtt{sid}, \mathcal{S}')$ *has been asked during the* prepare *stage;*
– *$\mathcal{A}(\mathsf{attack}, \zeta)$ continues interacting via queries and outputs some oracle identifier $U|s$.*

The adversary $\mathcal{A}$ wins in $\mathsf{Game}^{\mathsf{con}}_{\mathcal{A},\mathtt{P}}(\kappa)$ if ***all*** *of the following holds:*

(a) Π^s_U is honest, has terminated accepting $\tilde{k}$, and there is no $\mathtt{sid} \in \Psi$ *which contains its identifier $U|s$.*
(b) There are at most $n-1$ corrupted oracles that are partnered with Π^s_U.

$$\textit{We define:} \qquad \mathsf{Succ}^{\mathsf{con}}_{\mathcal{A},\mathtt{P}}(\kappa) := \Pr[\mathcal{A} \textit{ wins in } \mathsf{Game}^{\mathsf{con}}_{\mathcal{A},\mathtt{P}}(\kappa)]$$

and denote with $\mathsf{Succ}^{\mathsf{con}}_{\mathtt{P}}(\kappa)$ *the maximum probability of this event over all PPT adversaries* $\mathcal{A}$*; we say* P *provides* strong contributiveness *if this probability is negligible in* κ*.*

Since Ψ contains identifiers of sessions that have been invoked during the prepare stage, the requirement that no $\mathtt{sid} \in \Psi$ should contain $U|s$ excludes a trivial attack by which $\mathcal{A}$ chooses $\tilde{k}$ as a key computed in some session invoked during the prepare stage.

5.2 Security Results

The following theorems show that R-TDH1 is secure against strong outsiders and that IR-TDH1 is additionally secure against strong insider attacks.

In our proofs, similar to [8, 12], we assume that $\Sigma.\mathtt{Sign}$ is executed under the same protection mechanism as sk_i so that any randomness used to compute the signature will not be revealed in response to a *RevealState* query.

As the underlying number-theoretic assumption we use the well-known Square-Exponent Decisional Diffie-Hellman (SEDDH) assumption [22, 35], i.e. the following probability is assumed to be negligible:

$$\mathsf{Adv}^{\mathtt{SEDDH}}_{\mathbb{G}}(\kappa) =^{\max}_{\mathcal{A}'} \Big| \Pr_{a}\big[\mathcal{A}'(g, g^a, g^{a^2}) = 1\big] - \Pr_{a,b}\big[\mathcal{A}'(g, g^a, g^b) = 1\big]\Big|.$$

Theorem 1. *If* Σ *is existentially unforgeable under chosen message attacks, if* F *is pseudo-random, and* $\mathbb{G}$ *is* SEDDH*-hard then* R-TDH1 *provides strong AKE-security, and*

$$\mathsf{Adv}^{\mathsf{ake}}_{\mathtt{R\text{-}TDH1}}(\kappa) \leq \frac{Nq_s^2}{2^{\kappa-1}} + 2N^2\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa) + q_s N^2 \mathsf{Adv}^{\mathtt{SEDDH}}_{\mathbb{G}}(\kappa) + 2q_s \mathsf{Adv}^{\mathsf{prf}}_{F}(\kappa).$$

Proof. We define a sequence of games: $\mathbf{G}_i$, $i = 0, \ldots, 5$ (whereby $\mathbf{G}_4$ is a sequence of $n-1$ hybrid games where n is the number of invoked oracles in the attacked session) with the adversary $\mathcal{A}$ against the strong AKE-security of R-TDH1. In each game $\mathsf{Win}^{\mathsf{ake}}_i$ denotes the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in Game $\mathbf{G}_i$.

Game $\mathbf{G}_0$. This is the real game $\mathsf{Game}^{\mathsf{ake}-b}_{\mathcal{A},\mathtt{R\text{-}TDH1}}(\kappa)$ where a *simulator* Δ simulates the execution of the protocol and answers all queries of $\mathcal{A}$.

Recall that the *Test*$(U_i|s)$ query is asked to a fresh oracle $\Pi^s_{U_i}$ which has accepted, and that $\mathcal{A}$ then receives either a random string or a session group key $k^s_{U_i}$. Our definition of the oracle freshness restricts $\mathcal{A}$ from legal participation in the attacked session and from opening oracles that are partnered with $\Pi^s_{U_i}$ until these oracles terminate (having erased ephemeral secrets as required in R-TDH1).

Game $\mathbf{G}_1$. This game is identical to Game $\mathbf{G}_0$ except that Δ aborts and b' is set at random if any two honest oracles identified by $U|s$ and $U|s'$ that have been invoked for two different sessions choose the same nonce in Round 1. Since there are at most N users and at most q_s sessions we get $|\Pr[\mathsf{Win}^{\mathsf{ake}}_1] - \Pr[\mathsf{Win}^{\mathsf{ake}}_0]| \leq Nq_\mathsf{s}^2/2^\kappa$. This game ensures the uniqueness of nonces computed by each honest oracle Π^s_U in Round 2 over all invoked sessions.

Game $\mathbf{G}_2$. In this game the only exception is that Δ aborts and b' is set at random if $\mathcal{A}$ asks a query of the form *Broadcast*(sid,$U|t|m|\sigma$) such that $t \in \{2, 3\}$, the session id sid contains $U_i|s$ and there is an oracle of sender U which is partnered with $\Pi^s_{U_i}$ and is still treated as active during the t-th round, and σ is a valid signature on m, that has not been previously output by that oracle of U prior to a query *Corrupt*(U).

In other words the simulation fails if $\mathcal{A}$ outputs a successful forgery of the signature. A classical reductionist argument (e.g. [19]) can be used to construct a forger algorithm against Σ such that $|\Pr[\mathsf{Win}^{\mathsf{ake}}_2] - \Pr[\mathsf{Win}^{\mathsf{ake}}_1]| \leq N^2\mathsf{Succ}^{\mathtt{euf-cma}}_{\Sigma}(\kappa)$.

Since the concatenation $\texttt{nonces}|\texttt{pid}$ is part of every signed protocol message sent by the oracles this game prevents successful replay attacks.

Game $\mathbf{G}_3$. In this game we add the following rule: Δ chooses $q^* \in [1, q_\mathsf{s}]$ and aborts if the *Test* query does not occur in the q^*-th session. Let Q be the event that this guess for q^* is correct and $\Pr[\mathsf{Q}] = 1/q_\mathsf{s}$. Then, similar to the AKE-security proof of TDH1 in [12] we get $\Pr[\mathsf{Win}_2^{\mathsf{ake}}] = q_\mathsf{s}\left(\Pr[\mathsf{Win}_3^{\mathsf{ake}}] - \frac{1}{2}\right) + \frac{1}{2}$.

Game $\mathbf{G}_{4,j}$ for $j = 1, \ldots, n-1$. Each Game $\mathbf{G}_{4,j}$ is composed of two Sub-Games $\mathbf{G}_{4,j,1}$ and $\mathbf{G}_{4,j,2}$.

Sub-Game $\mathbf{G}_{4,j,1}$. In this (sub-)game Δ is given a tuple from the real SEDDH distribution, i.e., $(g, A = g^a, B = g^{a^2})$ for some unknown $a \in \mathbb{G}$, and embeds it into the simulation of the q^*-th session as follows. In Round 2 for each of n' active oracles Π_i the simulator defines the public value $y_i := A^{\alpha_i}$ for some random $\alpha_i \in_R \mathbb{G}$. In Round 3 for every remaining active oracle Π_i assigned to the leaf node $\langle n-1, 0\rangle$ (if $i = 1$) or $\langle n-i+1, 1\rangle$ (if $i > 1$), the iterative computation of the values $x_{c,0}^{(i)}$ and $y_{c,0}^{(i)}$, for $n-i-1 \geq c \geq 0$ is modified according to the following three rules (see also Fig. 2 for an example):

Rule 1: For $c = n-i-1$ downto $n-i-j+1$: the computation of $x_{c,0}^{(i)}$ is ignored and $y_{c,0}^{(i)}$ is defined to $A^{\alpha_{c,0}^{(i)}}$ for a randomly chosen $\alpha_{c,0}^{(i)}$; note that the rule is vacuous for $j = 1$.

Rule 2: For $c = n-i-j$, define $x_{c,0}^{(i)} := B^{\alpha_{c+1,0}^{(i)}\alpha_{c+1,1}}$ and $y_{c,0}^{(i)} = g^{x_{c,0}^{(i)}}$, where

$$\alpha_{c+1,0}^{(i)} = \begin{cases} \alpha_i & \text{if } j = 1 \text{ (value chosen in Round 2)} \\ \alpha_{c+1,0}^{(i)} & \text{as chosen in Rule 1, if } j > 1 \end{cases}$$

$$\alpha_{c+1,0}^{(i)} = \alpha_{i+1} \text{ (value chosen in Round 2)}$$

Rule 3: For $c = n-i-j-1$ downto 0, the computation is done normally: $x_{c,0}^{(i)} = {y_{c+1,1}^{(i)}}^{x_{c+1,0}^{(i)}}$ and $y_{c,0}^{(i)} = g^{x_{c,0}^{(i)}}$

Sub-Game $\mathbf{G}_{4,j,2}$. In this game Δ is given a tuple from the random SEDDH distribution, i.e., $(g, A = g^a, B = g^b)$ for some unknown $a, b \in \mathbb{G}$. Since the simulator performs the same steps as defined for the Sub-Game $\mathbf{G}_{4,j,1}$ the only difference between them is that in $\mathbf{G}_{4,j,1}$ $B = g^{a^2}$ and in $\mathbf{G}_{4,j,2}$ $B = g^b$. In each Game $\mathbf{G}_{4,j}$ the value B is embedded exactly $n-j$ times (that is once per set X_i for $n-j \geq i \geq 1$ using the re-randomization exponent $\alpha_{c+1,0}^{(i)}\alpha_{c+1,1}$ whose factor $\alpha_{c+1,0}^{(i)}$ is different for each i). Since $n \leq N$ the probability difference between $\mathbf{G}_{4,j,2}$ and $\mathbf{G}_{4,j,1}$ can be upper-bounded by $(N-j)\mathsf{Adv}_{\mathbb{G}}^{\texttt{SEDDH}}(\kappa)$.

Further we stress that by construction in Sub-Game $\mathbf{G}_{4,1,1}$ (the first sub-game in the sequence) the distribution of secret values in each X_i is identical to $\mathbf{G}_3$. And since j is a running variable from 1 to $n-1$ and $n \leq N$ we can upper-bound the probability difference between Game $\mathbf{G}_{4,n-1}$ (which ends with Sub-Game $\mathbf{G}_{4,n-1,2}$) and $\mathbf{G}_3$ as follows:

$$|\Pr[\mathsf{Win}_{4,n-1}^{\mathsf{ake}}] - \Pr[\mathsf{Win}_3^{\mathsf{ake}}]| \leq \sum_{j=1}^{N-1}(N-j)\mathsf{Adv}_{\mathbb{G}}^{\texttt{SEDDH}}(\kappa).$$

The consequence of $\mathbf{G}_{4,n-1}$ is that among different sets $X_i = \{x_{c,0}^{(i)}\}_{l \geq c \geq 0}$ computed by Δ in the q^*-th session all values $x_{c,0}^{(i)}$ are random and independent in $\mathbb{G} = \mathbb{Z}_q$ (the equality is due to the construction of $\mathbb{G}$ [12]). In particular, this implies that the value $x_{0,0}^{(\gamma)}$ used by every active oracle Π_i to derive the group key k_i in the q^*-th session is uniformly distributed in $\{0,1\}^\kappa$ (since κ is the length of q).

Game $\mathbf{G}_5$. In this game Δ replaces in the q^*-th session f by a truly random function. Hence, k_i computed by every active oracle Π_i including the one for which the *Test* query is asked is uniformly distributed, and $|\Pr[\mathsf{Win}_5^{\mathsf{ake}}] - \Pr[\mathsf{Win}_{4,n-1}^{\mathsf{ake}}]| \leq \mathsf{Adv}_F^{\mathsf{prf}}(\kappa)$. And since k_i is uniform: $\Pr[\mathsf{Win}_5^{\mathsf{ake}}] =$

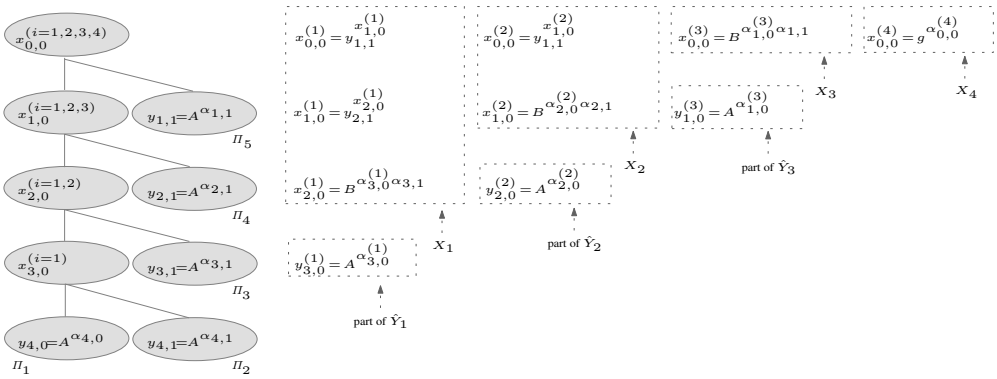

Fig. 2. Snapshot of $\mathbf{G}_{4,2,1}$ and $\mathbf{G}_{4,2,2}$ with oracles Π_i, $1 \leq i \leq 5$. In $\mathbf{G}_{4,2,1}$: $(A = g^a, B = g^{a^2})$. In $\mathbf{G}_{4,2,2}$: $(A = g^a, B = g^b)$. Left side: Δ embeds A into y_{l_i,v_i}. Right side: Δ follows the defined rules, i.e., Rule 1: it defines randomized $y_{3,0}^{(1)}$, $y_{2,0}^{(2)}$, and $y_{1,0}^{(3)}$ leaving corresponding $x_{3,0}^{(1)}$, $x_{2,0}^{(2)}$, and $x_{1,0}^{(3)}$ undefined, uses $\alpha_{0,0}^{(4)}$ to define $x_{0,0}^{(4)}$ (note that $x_{0,0}^{(4)}$ is already randomized at the end of Game $\mathbf{G}_{4,1,2}$); Rule 2: embeds B in each second value of each X_i $(i = 1, 2, 3)$; Rule 3: computes all subsequent values within X_i as specified in R-TDH1.

$1/2$. Combining the previous equations, we obtain the desired inequality for $\mathsf{Adv}^{\mathsf{ake}}_{\text{R-TDH1}}(\kappa)$. negligible advantage

Theorem 2. *If Σ is existentially unforgeable under chosen message attacks,* $\mathtt{Lg^2EqLg}$ *is zero-knowledge, and $\mathbb{G}$ is* SEDDH*-hard then* IR-TDH1 *provides strong AKE-security in ROM, and*

$$\mathsf{Adv}^{\mathsf{ake}}_{\text{IR-TDH1}}(\kappa) \leq \frac{Nq_s^2}{2^{\kappa-1}} + 2N\mathsf{Succ}^{\mathsf{euf-cma}}_{\Sigma}(\kappa) + q_s N^2 \left(\mathsf{Adv}^{\mathsf{zk}}_{\mathtt{Lg^2EqLg}}(\kappa) + \mathsf{Adv}^{\mathtt{SEDDH}}_{\mathbb{G}}(\kappa)\right).$$

Proof (Sketch). This proof is identical to the proof of Theorem 1 except that we need to show that $\mathtt{Lg^2EqLg}$ proofs computed by every honest oracle Π_i within Z_i do not reveal any additional information to the outsider adversary that asks the *Test* query. For this we need to plug in an additional game prior to Sub-Game $\mathbf{G}_{4,1,1}$ (the first sub-game of $\mathbf{G}_{4,1}$) in which Δ simulates the $\mathtt{Lg^2EqLg}$ proofs $\{z_l^{(i)}\}_l$ in Z_i computed by each active Π_i in Round 3. It is clear that the simulation of $\mathtt{Lg^2EqLg}$ proofs can be done via the classical technique of programmable random oracles. Hence, we omit the details. Assuming that n oracles remain active in Round 3 the number of simulated $\mathtt{Lg^2EqLg}$ proofs within each Z_j is $n-j-1$ (remember, oracles Π_{n-1} and Π_n do not compute any proofs). Since $j \leq n-2$ and $n \leq N$ we can upper-bound the probability difference between $\mathbf{G}_{4,1,1}$ and this game by $\sum_{j=1}^{N-2}(N-j-1)\mathsf{Adv}^{\mathsf{zk}}_{\mathtt{Lg^2EqLg}}(\kappa)$.

Since the group key derivation in IR-TDH1 is performed through the random oracle H' we can omit Game $\mathbf{G}_5$. The session group key k_i computed by Π_i to which the *Test* query is asked is already uniform at the end of Game $\mathbf{G}_{4,n-1}$. It is easy to see that the combination of probability upper-bounds for the difference of sequence games gives the desired inequality for $\mathsf{Adv}^{\mathsf{ake}}_{\text{IR-TDH1}}(\kappa)$.

Theorem 3. *If Σ is existentially unforgeable under chosen message attacks and if* $\mathtt{Lg^2EqLg}$ *is sound then* IR-TDH1 *provides strong MA-security in ROM, and*

$$\mathsf{Succ}^{\mathsf{ma}}_{\text{IR-TDH1}}(\kappa) \leq \frac{Nq_s^2}{2^{\kappa}} + N^2\mathsf{Succ}^{\mathsf{euf-cma}}_{\Sigma}(\kappa) + \frac{q_s N^2}{2}\mathsf{Succ}^{\mathsf{snd}}_{\mathtt{Lg^2EqLg}}(\kappa).$$

Proof. In the following games, event $\mathsf{Win}^{\mathsf{ma}}_i$ means that $\mathcal{A}$ wins in $\mathbf{G}_i$.

Game $\mathbf{G}_0$. This is the real game $\mathsf{Game}^{\mathsf{ma}}_{\texttt{IR-TDH1}}(\kappa)$ played between Δ and $\mathcal{A}$. Recall that $\mathcal{A}$ wins if at some point there is a session $\texttt{sid}$ for which the last protocol round is finished and one of its honest oracles $\Pi^s_{U_i}$ that does not fail after this round: (a) does not accept a group key although other active partners exist, or (b) accepts a group key without being assured that all other honest oracles that remain active after this last round have been invoked, or (c) accepts a different group key.

We observe that, by construction of $\texttt{IR-TDH1}$, every honest oracle $\Pi^s_{U_i}$ which has been invoked to participate in some session $\texttt{sid}$ and during the group key derivation phase holds $\texttt{pid}^s_{U_i}$ consisting of at least two identities (from which one is U_i) will compute the session group key $k^s_{U_i}$, and, thus accept. Hence, the probability that condition (a) ever occurs is 0. Therefore, in the following we focus on conditions (b) and (c).

Game $\mathbf{G}_1$. In this game, as in Game $\mathbf{G}_1$ from the proof of Theorem 1, Δ aborts if any two honest oracles identified by $U|s$ and $U|s'$ that have been invoked for two different sessions choose the same nonce in Round 1. Thus, $|\Pr[\mathsf{Win}^{\mathsf{ma}}_1] - \Pr[\mathsf{Win}^{\mathsf{ma}}_0]| \leq Nq_{\mathsf{s}}^2/2^{\kappa}$.

Game $\mathbf{G}_2$. In this game, as in Game $\mathbf{G}_2$ from the proof of Theorem 1, we eliminate signature forgeries in queries of the form $\mathit{Broadcast}(\texttt{sid}, U|t|m|\sigma)$ with $t \in \{2,3\}$ and get $|\Pr[\mathsf{Win}^{\mathsf{ma}}_2] - \Pr[\mathsf{Win}^{\mathsf{ma}}_1]| \leq N^2\mathsf{Succ}^{\texttt{euf-cma}}_{\Sigma}(\kappa)$. As a consequence of Game $\mathbf{G}_1$ we also eliminate successful replay attacks.

If an oracle accepts a key, then it is clear that it must have received correctly signed messages from its partners; having excluded forgeries, it means that these partners have actually been invoked. Thus, we exclude condition (b).

Further, each active and honest $\Pi^s_{U_i}$ must have received the same Round 3 message $U_j|3|M|\sigma'_j$ for all $U_j \in \texttt{pid}^s_{U_i}$ where M is either $\hat{Y}_j|Z_j$ or '$\texttt{alive}$' (depending on the assignments in the tree) and each $\Pi^s_{U_i}$ holds the same set Y constructed from the received Round 2 messages (which is also signed by σ'_j).

Game $\mathbf{G}_3$. In this game Δ aborts if on behalf of any two partnered honest oracles $\Pi^s_{U_i}$ and $\Pi^t_{U_j}$ that are active during the group key derivation phase Δ computes two different values for $x^{(\gamma)}_{0,0}$ which should be used by $\Pi^s_{U_i}$ and $\Pi^t_{U_j}$ to derive the session group key.

Assume that this failure event occurs and let $U_\gamma|3|\hat{Y}_\gamma|Z_\gamma|\sigma'_\gamma$ be the Round 3 message received by $\Pi^s_{U_i}$ and $\Pi^t_{U_j}$. In line with the notations used in $\texttt{IR-TDH1}$ upon the construction of T_n we denote $\Pi^s_{U_i}$ as Π_i (assigned to the leaf node $\langle n-i+1,1\rangle$) and $\Pi^s_{U_j}$ as Π_j (assigned to the leaf node $\langle n-j+1,1\rangle$) and assume w.l.o.g. that $i<j$. We know that $\gamma < i$ since Π_γ has the lowest index.

Since oracles Π_i and Π_j receive the same Round 3 message (due to the broadcast channel) and since in previous games we have excluded any impersonation attacks on honest oracles, we conclude that if both oracles compute different values for $x^{(\gamma)}_{0,0}$ then Π_γ must be malicious.

Now we focus on the computation of $x^{(\gamma)}_{0,0}$ by Π_j (still assuming that $j>i$). The first value computed by Π_j in the key derivation phase using its secret exponent $x_{n-j+1,1}$ and $y^{(\gamma)}_{n-j+1,0} \in \hat{Y}_\gamma$ is $x^{(\gamma)}_{n-j,0}$. Also Π_i computes this value, however, using the secret exponent $x^{(\gamma)}_{n-j+1,0}$ and $y_{n-j+1,1} \in Y_i$. Since the computation of $x^{(\gamma)}_{0,0}$ is deterministic and both oracles Π_i and Π_j use identical sets Y (and $Y_i \subset Y$) we follow that Π_i and Π_j compute different values for $x^{(\gamma)}_{0,0}$ only if they compute different values for $x^{(\gamma)}_{n-j,0}$.

Since Π_j honestly uses its secret exponent $x_{n-j+1,1}$ its computed value for $x^{(\gamma)}_{n-j,0}$ is different from that computed by Π_i only if $y^{(\gamma)}_{n-j+1,0} \in \hat{Y}_\gamma$ used by Π_j does not have the form $g^{x^{(\gamma)}_{n-j+1,0}}$ where $x^{(\gamma)}_{n-j+1,0}$ is the exponent used by Π_i. In turn, Π_i assigned to $\langle n-i+1,1\rangle$ computes $x^{(\gamma)}_{n-j+1,0}$ through iteration starting with the computation of $x^{(\gamma)}_{n-i,0}$ for which it honestly uses its secret exponent $x_{n-i+1,1}$.

By construction, if $y^{(\gamma)}_{n-j+1,0} \in \hat{Y}_\gamma$ used by Π_j does not have the required form $g^{x^{(\gamma)}_{n-j+1,0}}$, then the set Z_γ contains at least one forged proof. Moreover this proof can be discovered by Δ as follows. Δ uses $x_{n-i+1,1}$, $\hat{Y}_\gamma$, and Y to iteratively compute each $x^{(\gamma)}_{l,0}$ for $n-i \geq l \geq n-j+1$ and the

corresponding $y_{l,0}^{(\gamma)} := g^{x_{l,0}^{(\gamma)}}$. Then, Δ sequentially checks whether each computed $y_{l,0}^{(\gamma)}$ is the same as the one included in $\hat{Y}_\gamma$ until it finds the first one which is different. At least one such value must exist; otherwise, oracles would have computed identical values for $x_{n-j,0}^{(\gamma)}$. When Δ finds the first such value $y_{l,0}^{(\gamma)}$ the corresponding $\mathtt{Lg^2EgLg}$ proof $z_l^{(\gamma)}$ must be a forgery since it claims that $y_{l,0}^{(\gamma)} = g^{y_{l+1,1}^{x_{l+1,0}^{(\gamma)}}}$, a false statement.

Thus, if the described failure event occurs then Δ is able to output a forged $\mathtt{Lg^2EgLg}$ proof. Since for each oracle Π_j with $j \leq n-2$ there are exactly $(n-j-1)$ $\mathtt{Lg^2EgLg}$ proofs we can upper-bound

$$|\Pr[\mathsf{Win}_3^{\mathsf{ma}}] - \Pr[\mathsf{Win}_2^{\mathsf{ma}}]| \leq q_{\mathsf{s}} \sum_{j=1}^{N-2} (N-j-1)\mathsf{Succ}_{\mathtt{Lg^2EqLg}}^{\mathtt{snd}}(\kappa).$$

As a consequence of this game any two honest oracles $\Pi_{U_i}^s$ and $\Pi_{U_j}^t$ that are partnered with respect to some $\mathtt{sid}$ and remain active during the key derivation phase compute identical values for $x_{0,0}^{(\gamma)}$ and thus, accept with identical session group keys $k_{U_i}^s = k_{U_j}^s$. This excludes condition (c). Hence, $\Pr[\mathsf{Win}_3^{\mathsf{ma}}] = 0$. Combining the previous equations we obtain the desired inequality for $\mathsf{Succ}_{\mathtt{IR\text{-}TDH1}}^{\mathsf{ma}}(\kappa)$.

Theorem 4. $\mathtt{IR\text{-}TDH1}$ *provides strong contributiveness in ROM, and*

$$\mathsf{Succ}_{\mathtt{IR\text{-}TDH1}}^{\mathsf{con}}(\kappa) \leq \frac{Nq_{\mathsf{s}}^2 + 2Nq_{\mathsf{s}} + q_{H'}^2}{2^\kappa}.$$

Proof. In the following games, event $\mathsf{Win}_i^{\mathsf{con}}$ means that $\mathcal{A}$ wins in $\mathbf{G}_i$.

Game $\mathbf{G}_0$. This is the real game $\mathsf{Game}_{\mathcal{A},\mathtt{IR\text{-}TDH1}}^{\mathsf{con}}(\kappa)$, in which the honest players are simulated by Δ. Recall that $\mathcal{A}$ wins if after the stage $\mathsf{prepare}$ it returned $\tilde{k}$ and if in the stage attack the honest oracle $\Pi_{U_i}^s$ accepts $\tilde{k}$.

Game $\mathbf{G}_1$. In this game, similar to the previous proofs, Δ aborts if any two honest oracles identified by $U|s$ and $U|s'$ that have been invoked for two different sessions choose the same nonce in Round 1. Thus, $|\Pr[\mathsf{Win}_1^{\mathsf{con}}] - \Pr[\mathsf{Win}_0^{\mathsf{con}}]| \leq Nq_{\mathsf{s}}^2/2^\kappa$. This implies that in every session in which U_i participates through some oracle $\Pi_{U_i}^s$ which remains active during the group key derivation phase the concatenation of random nonces $\mathtt{nonces}_i$ held by $\Pi_{U_i}^s$ contains a fresh nonce r_i. Since the concatenation preserves the lexicographic order of user identities in $\mathtt{pid}_{U_i}^s$ the concatenation $\mathtt{nonces}_i|\mathtt{pid}_{U_i}^s$ used in addition to $x_{0,0}^{(\gamma)}$ as input for H' to derive $k_{U_i}^s$ is unique for each session.

Game $\mathbf{G}_2$. In this game Δ aborts if $\mathcal{A}(\mathsf{prepare})$ returned some $\tilde{k}$ which it did not receive from Δ in response to some query to the random oracle H' but which is computed by Δ on behalf of some honest $\Pi_{U_i}^s$ as $k_{U_i}^s$ (as output of H') invoked during the attack stage. Since H' is modeled as a random oracle the probability that this event occurs for any honest oracle and any invoked session is given by the probability for the random guess of the output of H', so that $|\Pr[\mathsf{Win}_2^{\mathsf{con}}] - \Pr[\mathsf{Win}_1^{\mathsf{con}}]| \leq Nq_{\mathsf{s}}/2^\kappa$.

Therefore, $\mathcal{A}$ wins in this game only if it queried H' on some input m during the $\mathsf{prepare}$ stage and received $\tilde{k}$ in response, which is then accepted by $\Pi_{U_i}^s$ as $k_{U_i}^s$.

Case $m = x_{0,0}^{(\gamma)}|\mathtt{nonces}_i|\mathtt{pid}_{U_i}^s$: The probability of $\mathcal{A}$ to win is given by the guess of r_i, i.e. $Nq_{\mathsf{s}}/2^\kappa$ (due to Game $\mathbf{G}_1$).

Case $m \neq x_{0,0}^{(\gamma)}|\mathtt{nonces}_i|\mathtt{pid}_{U_i}^s$: The probability of $\mathcal{A}$ to win is upper-bounded by $q_{H'}^2/2^\kappa$ by the birthday paradox.

Thus, $\Pr[\mathsf{Win}_2^{\mathsf{con}}] = (Nq_{\mathsf{s}} + q_{H'}^2)/2^\kappa$. Combining previous equations gives us the desired inequality for $\mathsf{Succ}_{\mathtt{IR\text{-}TDH1}}^{\mathsf{con}}(\kappa)$.

6 Comparison with Prior Work

At a glance, Table 1 describes how R-TDH1 and IR-TDH1 fit into the current state of the art of provably secure RGKE protocols in terms of security, robustness and complexity: (i) we indicate whether AKE-security, MA-security and contributiveness (CON) is achieved, and for which strength of corruptions, (ii) we indicate the maximum of users that may fail without disrupting the protocol execution (fully robust protocols have robustness of $n-2$), and (iii) we compare the broadcast complexity and the total number of operations *per* user.

Table 1. Security, Robustness, and Complexity of R/IR-TDH1 and other *Robust* GKE Protocols

	Out-/Insider Security				Robustness	Complexity		
RGKE Prot.	AKE	MA	CON	Model	max Faults ($k \leq$)	Rounds	Broadcast	Ops
adopted [14]	strong	-	-	STD	$n-2$	2	$O(n^2)$	$O(n)$
[21]	weak	weak	weak	STD	$n/2-1$	7	$O(nk)$	$O(n)$
BD-RGKA [22]	weak	-	-	STD	$n-2$	2	$O(n^3)$	$O(n^2)$
RGKA [22]	weak	-	-	STD	$n-2$	2	$O(n^2)$	$O(n)$
t-RGKA [22]	weak	-	-	STD	$2t-1$	2	$O(nt)$	$O(t)$
RGKA$'$ [22]	weak	-	-	STD	$n-2$	$O(\delta)$	$O(n \log n)$	$O(n)$
R-TDH1	strong	-	-	STD	$n-2$	3	$O(n^2)$	$O(n)$
IR-TDH1	strong	strong	strong	ROM	$n-2$	3	$O(n^2 l)$	$O(nl)$
TDH1 [12]	strong	strong	strong	STD	0	3	$O(n)$	$O(n)$

The out-/insider security entries reflect the formally proven properties of the protocols, though it might be possible to amend the protocols from [14, 21, 22] to achieve strong outsider and insider security using techniques that are close to those proven secure for R-TDH1 and IR-TDH1.

To ensure fair comparison we adopt [14] to the reliable broadcast setting as described in [22] and add authentication costs to non-authenticated protocols from [21, 22] based on the technique from [24].

We highlight that R-TDH1 and the reliable broadcast version of [14] are the only RGKE protocols that have been formally proven to achieve strong outsider security. The protocols from [22] are proven under consideration of weak corruptions only. In terms of complexity and robustness R-TDH1 is similar to both RGKA and the modified version of [14].

We proved that IR-TDH1 provides strong outsider and insider security, while [14, 22] did not address insider security, and [21] did not consider strong corruptions. Compared to R-TDH1, IR-TDH1 has loss in the broadcast and computation complexity by factor $O(l)$ where l ranges from the number of users that do not fail ($n-k$) to 1. Finally, we notice that compared to the original TDH1 from [12] the use of our tree replication technique achieves full robustness but increases the communication complexity by factors $O(n)$ and $O(nl)$, respectively.

7 Conclusion

This paper introduced two fully robust versions of the Tree-Diffie-Hellman protocol TDH1 from [12] based on our novel tree replication technique. R-TDH1 preserves the strong outsider security of the original protocol, whereas IR-TDH1 presents the first construction of a fully robust GKE protocol that remains resilient to strong insider attacks. We proved both protocols in a security model which is of independent interest as it combines strengths of several previous modeling approaches.

As mentioned, some existing robust GKE protocols can also be modified to achieve insider security using NIZK proofs, however, this would require random oracles as well. Hence, designing a fully robust

GKE protocol with strong outsider and insider security in the standard model remains an interesting open problem.

References

1. Y. Amir, C. Nita-Rotaru, J. L. Schultz, J. R. Stanton, Y. Kim, G. Tsudik. Exploring Robustness in Group Key Agreement. In *Proc. of ICDCS'01*, p. 399–408. IEEE CS, 2001.
2. G. Ateniese, D. X. Song, and G. Tsudik. Quasi-Efficient Revocation in Group Signatures. In *Financial Cryptography'02*, *LNCS* 2357, p. 183–197. Springer, 2002.
3. G. Ateniese, M. Steiner, and G. Tsudik. Authenticated Group Key Agreement and Friends. In *Proc. of ACM CCS'98*, p. 17–26. ACM Press, 1998.
4. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS'93*, p. 62–73. ACM Press, 1993.
5. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT'94*, *LNCS* 950, p. 275–286. Springer, 1994.
6. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.
7. E. Bresson, O. Chevassut, and D. Pointcheval. Provably Authenticated Group Diffie-Hellman Key Exchange — The Dynamic Case. In *ASIACRYPT'01*, *LNCS* 2248, p. 290–390. Springer, 2001.
8. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *EUROCRYPT'02*, *LNCS* 2332, p. 321–336. Springer, 2002.
9. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *ACM CCS'01*, p. 255–264. ACM Press, 2001.
10. E. Bresson and M. Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *Proc. of ATC '07*, *LNCS* 4610, p. 395–409. Springer, 2007.
11. E. Bresson, M. Manulis, and J. Schwenk. On Security Models and Compilers for Group Key Exchange Protocols. In *Proc. of IWSEC '07*, *LNCS* 4752, p. 292–307. Springer, 2007.
12. E. Bresson and M. Manulis. Securing Group Key Exchange against Strong Corruptions. In *Proc. of ASIACCS '08*, p. 249–261. ACM, 2008.
13. C. Cachin, K. Kursawe, and V. Shoup. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement using Cryptography. In *Proc. of PODC'00*, p. 123–132. ACM Press, 2000.
14. C. Cachin and R. Strobl. Asynchronous Group Key Exchange with Failures. In *Proc. of PODC'04*, p. 357–366. ACM Press, 2004.
15. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *EUROCRYPT'01*, *LNCS* 2045, p. 453–474. Springer, 2001.
16. R. Canetti and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *STOC'93*, p. 42–51. ACM Press, 1993.
17. G. V. Chockler and I. Keidar and R. Vitenberg. Group Communication Specifications: A Comprehensive Study. *ACM Computing Surveys*, 33(4):427-469. ACM, 2001.
18. K.-K. R. Choo, C. Boyd, and Y. Hitchcock. Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. In *ASIACRYPT'05*, *LNCS* 3788, p. 585–604. Springer, 2005.
19. M. Choudary Gorantla, C. Boyd, and J. M. González Nieto. Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In *PKC'09*, *LNCS* 5443, p. 105–123. Springer, 2009.
20. G. D. Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson. How to Forget a Secret. In *Proc. of STACS'99*, *LNCS* 1563, p. 500–509. Springer, 1999.
21. Y. G. Desmedt, J. Pieprzyk, R. Steinfeld, and H. Wang. A Non-Malleable Group Key Exchange Protocol Robust Against Active Insiders. In *Proc. of ISC'06*, *LNCS* 4176, p. 459–475. Springer, 2006.
22. S. Jarecki and J. Kim and G. Tsudik. Robust Group Key Agreement using Short Broadcasts. In *Proc. of ACM CCS '07*, p. 411–420. ACM, 2007.
23. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key Exchange Protocols. In *Proc. of ACM CCS'05*, p. 180–189. ACM Press, 2005.
24. J. Katz and M. Yung. Scalable Protocols for Authenticated Group Key Exchange. In *CRYPTO'03*, *LNCS* 2729, p. 110–125. Springer, 2003.
25. Y. Kim, A. Perrig, and G. Tsudik. Group Key Agreement Efficient in Communication. *IEEE Transactions on Computers*, 53(7):905–921, 2004.

26. Y. Kim, A. Perrig, and G. Tsudik. Tree-Based Group Key Agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, February 2004.
27. J. Kim and G. Tsudik. Survival in the Wild: Robust Group Key Agreement in Wide-Area Networks. In *ICISC'08*, *LNCS* 5461, p. 66–83, Springer, 2008.
28. H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Proc. of CRYPTO'05*, *LNCS* 3621, p. 546–566. Springer, 2005.
29. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger Security of Authenticated Key Exchange. In *Proc. of ProvSec'07*, *LNCS* 4784, p. 1–16. Springer, 2007.
30. M. Manulis. *Provably Secure Group Key Exchange.* PhD thesis, Ruhr Univ. Bochum, 2007.
31. C. J. Mitchell, M. Ward, and P. Wilson. Key Control in Key Agreement Protocols. *Electronic Letters*, 34(10):980–981, 1998.
32. T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO '91*, *LNCS* 576, p. 129–140. Springer, 1991.
33. B. Pfitzmann and M .Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *IEEE S&P '01*, p. 184–200. IEEE CS, 2001.
34. J. Pieprzyk and H .Wang. Key Control in Multi-party Key Agreement Protocols. In *CCC/PCS '03*, vol. 23, 2003.
35. A.-R. Sadeghi and M. Steiner. Assumptions Related to Discrete Logarithms: Why Subtleties Make a Real Difference. In *EUROCRYPT '01*, *LNCS* 2045, p. 244–261. Springer, 2001.
36. V. Shoup. On Formal Models for Secure Key Exchange (Version 4). TR-RZ 3120, IBM Research, November 1999.
37. M. Stadler. Publicly Verifiable Secret Sharing. In *EUROCRYPT '96*, *LNCS* 1070, p. 90–99. Springer, 1996.
38. M. Steiner. *Secure Group Key Agreement*. PhD thesis, Saarland University, 2002.