Securing group key exchange against strong corruptions and key registration attacks

Emmanuel Bresson

DCSSI Crypto Lab,
51, bld de La Tour-Maubourg,
75700 Paris 07 SP, France
E-mail: emmanuel@bresson.org

Mark Manulis*

UCL Crypto Group,
Place du Levant 3,
1348 Louvain-la-Neuve, Belgium
E-mail: mark.manulis@uclouvain.be
*Corresponding author

Abstract: In Group Key Exchange (GKE) protocols, users usually extract the group key using some auxiliary (ephemeral) secret information generated during the execution. Strong corruptions are attacks by which an adversary can reveal these ephemeral secrets, in addition to the possibly used long-lived keys. Undoubtedly, security impact of strong corruptions is serious, and thus specifying appropriate security requirements and designing secure GKE protocols appears an interesting yet challenging task – the aim of our article. We start by investigating the current setting of strong corruptions and derive some refinements like opening attacks that allow to reveal ephemeral secrets of users without their long-lived keys. This allows to consider even stronger attacks against honest, but 'opened' users. Further, we define strong security goals for GKE protocols in the presence of such powerful adversaries and propose a 3-round GKE protocol, named TDH1, which remains immune to their attacks under standard cryptographic assumptions. Our security definitions allow adversaries to register users and specify their long-lived keys, thus, in particular capture attacks of malicious insiders for the appropriate security goals such as Mutual Authentication, key confirmation, contributiveness, key control and key-replication resilience.

Keywords: authenticated group key exchange; GKE; contributiveness; insider attacks; key registration; mutual authentication; MA; strong corruptions; tree Diffie–Hellman; TDH1.

Reference to this paper should be made as follows: Bresson, E. and Manulis, M. (2008) 'Securing group key exchange against strong corruptions and key registration attacks', *Int. J. Applied Cryptography*, Vol. 1, No. 2, pp.91–107.

Biographical notes: Emmanuel Bresson received his PhD at the École normale supérieure in Paris. He works as a Cryptography Expert for government teams. His main research subjects involve key exchange mechanisms and authentication for multi-party protocols with provable security. He has published his work in many international conference papers and security focusing journals.

Mark Manulis received his PhD in Computer Science from the Ruhr University Bochum in 2007. His research focuses on security and cryptography related to key management, authentication, anonymity and privacy in distributed applications and (wireless) communications.

1 Introduction

A Group Key Exchange (GKE) protocol provides participants with a common secret group key. The main (semantic) security requirement called Authenticated Key Exchange (AKE; Bresson, Chevassut and Pointcheval, 2001, 2002a) aims to ensure that the established key is indistinguishable from a random one by any outsider adversary. The second requirement called Mutual Authentication (MA; Bresson, Chevassut and Pointcheval, 2001) aims to ensure that all legitimate protocol participants and only they have actually computed identical session group keys. These security requirements have been extensively studied in the literature (see the recent survey in Manulis, 2006). In the most basic scenarios, all users are somehow protected, that is, the adversary has no control over them, and is restricted to attacks carried out through

Copyright © 2008 Inderscience Enterprises Ltd.

the network (which nevertheless include impersonation attacks where the adversary talks on the network by pretending being a legitimate user).

In order to take into account, further real-life threats on users and the notion of forward secrecy is usually considered. Forward secrecy means that the established session key remains secure 'in the future', that is, remains indistinguishable from random even if the adversary learns used long-lived keys in the future. The notion is motivated by the fact that, by nature, long-lived keys get more chance to be leaked to an attacker than ephemeral secrets.

The next known kind of corruptions, referred to as strong corruptions in Shoup (1999), Steiner (2002) and Bresson, Chevassut and Pointcheval (2002a), provides the adversary with even more information. Namely, the adversary gets the user's ephemeral secrets in addition to the long-lived keys. But, he is not allowed to get the established session group key. Shoup (1999) explains why such a separation makes sense: session keys are typically controlled by higher-level applications that will use them, while internal, ephemeral secrets are specific to the GKE protocol execution and could be erased once this protocol is finished.

Actually in GKE, it seems impossible to obtain secrecy when ephemeral secrets are revealed during the protocol session: if the adversary (even 'passively') can learn all intermediate key material, then he will likely be able to compute the final group key. On the other hand, in dynamic groups there are many cases where ephemeral secrets of a particular session are subsequently re-used (in addition to some refreshed data) to update the group key. Then, it is important to ask how the knowledge of ephemeral secrets in a corrupted session impacts the security of other sessions (past and future). This is precisely where the notion of strong forward/backward secrecy raises up.

At this point, we precise the corruption types considered in this article. First, we consider users who are corrupted and are introduced by the adversary. We assume that the users are corrupted in a passive mode (rather than active), i.e. the adversary can only 'read' secrets held by the attacked user (whatever these secrets are ephemeral or long-lived). Through the knowledge of the long-lived key, the adversary can (typically) inject signed messages on behalf of the user while preventing the original user's messages from being delivered. In fact, this allows an active participation of the adversary during the protocol execution, and thus we say the adversary is active; but, this refers to his ability to control the network, not the user's behaviour. On the other hand, we also wish to capture security threats coming from the users that are fully controlled by the adversary. Therefore, we allow the adversary to introduce new users and to register their long-lived keys. The adversary that corrupts or adds users is adaptive (opposed to static) in the sense that it chooses which users to corrupt or to introduce based on the information he gained so far and in any stage of the protocol execution. Secondly, when considering user corruptions, in order to further refine the security definitions, our intention is to separate the long-lived key from the internal state which contains ephemeral secrets and to specify when the adversary can learn them. Through this separation, we explicitly allow the adversary to reveal ephemeral secrets without revealing the long-lived key; we call this opening attacks. They are the balanced complement of weak corruption attacks, where long-lived keys are revealed, but ephemeral secrets are not. We note that under opening attacks, there is a hope to prevent the adversary from actively participating in the protocol on behalf of the opened parties, since he does not receive the long-lived keys. Finally, we notice that the strong corruption model in its current form is the best (or worst) of two worlds: if the adversary corrupts then it obtains the long-lived keys and the ephemeral data, if it does not corrupt then it obtains nothing. But, separating the attacks in two distinct modes allows to refine and opt for stronger security definitions.

Consideration of the adversary that corrupts and introduces users allows us to address security threats against GKE protocols that may arise also in the presence of malicious participants/insiders – corrupted or introduced users whose long-lived keys are known to the adversary. The adversary acting as malicious participants might be able via opening attacks to obtain information from the internal states of the honest users; the goal of the adversary is then to influence their behaviour.

Usually, the AKE requirement is defined from the perspective of some (fresh) session, and thus makes sense only if the adversary is restricted to neither participate on behalf of a user nor to obtain any ephemeral secret in that session, i.e. all during the protocol session active users must be honest and not opened. On the other hand, the MA requirement remains meaningful even without such limitations. Even if achieving MA without AKE is of low interest for key exchange protocols, it is still legitimate to ask whether achieving MA under strong corruptions during the attacked session is possible. This especially, since the MA requirement still makes sense in the presence of malicious participants and may also be useful for protocols other than key exchange. Furthermore, consideration of malicious insiders raises attacks related to key control and contributiveness: for instance, think of a participant who can force the same key to be obtained in two different sessions (e.g. key-replication; Krawczyk, 2005). Here, we recall that the question on who controls the value of the group key states the important difference between GKE and group key transport protocols (Bresson and Manulis, 2007). In GKE protocols, it is essential that the key is computed from inputs (contributions) of all participants such that even a strict subset of participants cannot enforce the final value of the group key. Especially, when considering asynchronous communication and malicious participants who can choose own contributions arbitrarily and may additionally reveal internal states of honest participants at any stage of the protocol execution through opening attacks, preventing key control and ensuring contributiveness for the honest users appears to be a challenging task.

1.1 Related work

1.1.1 Original definitions

The AKE- and MA-security requirements (without strong corruptions and only for honest users) were originally given by Bresson et al. (2001), see Katz and Yung, 2003; Dutta, Barua and Sarkar, 2004; Kim, Lee and Lee, 2004, for variants and Bresson, Manulis and Schwenk, 2007, for some flaws. In Bresson, Chevassut and Pointcheval (2002a) and Bresson et al. (2001) modelled strong corruptions, but for AKE-security only, following the ideas of Shoup (1999) and Canetti and Krawczyk (2001) for two-party protocols, for which such strong AKE-security has been recently modelled in LaMacchia, Lauter and Mityagin (2007).

Katz and Shin (2005) extended the definition of MA-security by assuming misbehaving (malicious) protocol participants; and they provided a concrete generic solution (compiler) to prevent these attacks, however, without considering opening attacks against ephemeral secrets as well as key control and contributiveness. The significance of security against malicious participants was also recognised by Choo, Boyd and Hitchcock (2005) through unknown-key share attacks, by which an active adversary tries to make an honest protocol participant believe that the group key is shared with one party when it is in fact shared with another party.

1.1.2 On key control and contributiveness

Mitchell, Ward and Wilson (1998), see also Boyd and Mathuria (2003), gave informal definition of key control, to describe attacks where participants try to influence the resulting value of the key. Yet informally, Ateniese, Steiner and Tsudik (1998) proposed the notion of contributiveness meaning that all participants must equally contribute to the computation of the key and guarantee its freshness (see Steiner, 2002); these definitions emphasise the difference between key distribution and key exchange (Menezes, van Oorschot and Vanstone, 1996). Following these requirements, Bresson and Catalano (2004) have considered the (weaker) case where participants are honest, but have biased source of randomness so that an adversary can possibly gain extra information about the key. Deepening this, Bohli, Vasco and Steinwandt (2007) gave definitions of key control and contributiveness considering a (stronger) case where participants deliberately wish to influence the resulting value of the group key. Still, their definitions are based on the model from Bresson et al. (2001) and thus, do not consider strong corruptions. Finally, Krawczyk (2005) mentioned that a key exchange protocol should prevent key-replication attacks whose goal is to influence the acceptance of the same key in different protocol sessions.

1.1.3 Other work close to ours

Independent of our work, Desmedt et al. (2006) considered a property of non-malleability for GKE protocols, which is close to key control and contributiveness. Their security goal, called shielded-insider privacy, aims to prevent attacks where an outsider adversary upon communication with some malicious participants prior to the protocol execution, obtains information about the later computed group key. In order to ensure shielded-insider privacy, they use Pedersen's (1991) commitments; however, in case of strong corruptions committed secrets can still be revealed to the adversary (due to opening attacks), so that malicious participants would still be able to bias the computation. In our model, we do not consider this scenario explicitly, but focus on the (in)ability of the adversary representing malicious participants to predict the resulting value of the later established group key. Recently, Manulis (2006) analysed several existing models for GKE protocols with respect to considering strong corruptions: he pointed out that security against strong corruptions is currently considered in a rather restrictive way: only for strong forward secrecy of AKE-security. Moreover, none of the available game-based security models is complete enough to unify the most important definitions of AKE-, MA-security, and key control and contributiveness.

1.2 Contributions and organisation

We solve most of the problems put in light above by revisiting the GKE security model from the perspective of strong corruptions and key registration attacks. Further, we design a provably secure GKE protocol that resists these attacks.

1.2.1 Security model and stronger definitions

As our first contribution in Section 2, we provide the following:

- We model a powerful adversary who is given access to strong corruptions, by describing an appropriate game-based security model for GKE protocols, thus significantly extending the ideas from Bresson, Chevassut and Pointcheval (2002a).
- We formalise strong AKE-security by considering opening attacks that may occur in earlier and later protocol sessions.
- In our definition of strong MA-security, we consider the adversary that acts as malicious participants during the attacked session and opens all other (honest) users; due to the opening attacks our definition is stronger than the related one from Katz and Shin (2005).
- We formalise strong contributiveness as security against attacks that enforce any value chosen by the adversary as a group key (this includes key-replication; Krawczyk, 2005); since, the adversary can act as malicious participants and open all other (honest) participants our requirement is stronger compared to Bohli, Vasco and Steinwandt (2007).
- We strengthen the GKE security model by allowing the adversary to introduce users and register their long-lived keys; this is similar to the recent models in 2-party key exchange (LaMacchia, Lauter and Mityagin, 2007; Menezes and Ustaoglu, 2008) and is the main difference to the extended abstract of this article which appeared in Bresson and Manulis (2008) and also to many previous GKE security models.

1.2.2 Group Key Exchange protocol with strong security

As a second contribution in Section 3, we describe a 3-round GKE protocol, named TDH1, and prove that it provides strong versions of AKE-, MA-security and contributiveness, while the deployed techniques can be seen as general for many GKE protocols. TDH1 tolerates the following numbers of malicious insiders (out of n participants in total): for MA-security up to n–2, for contributiveness up to n–1, whereby all remaining honest users might be opened! Our security proofs do not rely on the Random Oracle Model (ROM; Bellare and Rogaway, 1993). The AKE-security of TDH1 is based on the Tree Decisional Diffie–Hellman (TDDH) assumption, introduced by Kim, Perrig and Tsudik (2004a,b). We give a formal definition of the underlying TDDH problem and show its polynomial equivalence to the standard Decisional Diffie–Hellman (DDH) problem (Boneh, 1998) by a proof which addresses arbitrary full binary trees, i.e. trees where each node has exactly zero or two leaves (note, Kim, Perrig and Tsudik, 2004a,b addressed only a subset, i.e. linear and complete trees).

2 Strong security definitions for Group Key Exchange

We start by (re)stating existing definitions and classical notations using the game-based approach. Note that another way (which we do not consider here) to define security requirements is to use the simulation-based approach, e.g. Katz and Shin, 2005, but see Remark 1.

2.1 Protocol execution and participants

2.1.1 Users, instance oracles and long-lived keys

Let $\mathcal{U}$ be a set of at most N users. Each $U_i \in \mathcal{U}$ holds a long-lived key LL_i and has several instances called oracles, denoted $\prod_i^s$ for $s \in \mathbb{N}$, participating in distinct concurrent executions. (When we do not refer to a specific user U_i we use the index U, e.g. $\prod_U^s$).

2.1.2 Internal states

Every $\prod_U^s$ maintains an internal state information state_U^s which is composed of all ephemeral secret information used during the protocol execution. The long-lived key LL_U is, in nature, excluded from it (moreover, the long-lived key is specific to the user, not to the oracle). An oracle $\prod_U^s$ is unused until initialisation (by which it is given the long-lived key LL_U). It then becomes a group member, associated to a particular session, and turns into the stand-by state where it waits for an invocation to execute the protocol. When the protocol starts, the oracle learns its partner id pid_U^s (and possibly session id sid_U^s) and turns into a processing state where it sends, receives and processes messages. During that stage, the internal state information state_U^s is maintained. After having computed k_U^s oracle $\prod_U^s$ accepts and terminates the execution of the protocol operation (possibly after some additional auxiliary steps) meaning that it would not send or receive further messages. If the protocol fails, $\prod_U^s$ terminates without accepting and k_U^s is set to an undefined value.

2.1.3 Session group key, session and partner IDs, group members

Every session is identified by a unique, publicly-known sid_U^s. In each session, each oracle $\prod_U^s$ gets a value pid_U^s that contains the identities of participating users (including U) and computes the session group key $k_U^s \in \{0,1\}^\kappa$, where κ is a security parameter.

By $\mathcal{G}(\prod_i^s) = \{\prod_j^t \text{ where } U_j \in \mathrm{pid}_{U_i}^s \text{ and } \mathrm{sid}_i^s = \mathrm{sid}_j^t\}$, we denote the group of oracle $\prod_i^s$ and say that $\prod_i^s$ and $\prod_j^t$ are partnered if $\prod_j^t \in \mathcal{G}(\prod_i^s)$ and $\prod_i^s \in \mathcal{G}(\prod_j^t)$. Sometimes, we simply write $\mathcal{G}$ to denote the group of oracles participating in the same protocol session. Then, each oracle in $\mathcal{G}$ is called a group member. Note that oracles in $\mathcal{G}$ may be ordered, e.g. lexicographically based on the user identities.

Definition 1. A GKE 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 in $\mathcal{U}$ is provided with a long-lived key LL_U.

P.Setup($\mathcal{S}$). On input a set $\mathcal{S}$ of n unused oracles a new group $\mathcal{G}$ is created and set to be $\mathcal{S}$. A probabilistic interactive protocol is executed between the oracles in $\mathcal{G}$ such that all oracles accept with the session group key and terminate.

A protocol is said to be correct if, when no adversary is present, all participants compute the same key. Note that our definition is independent of the communication channel, e.g. (asymmetric) broadcast, multi-cast or unicast.

2.2 Strong adversarial model

Now, we consider an adversary $\mathcal{A}$ which is a Probabilistic Polynomial-Time (PPT) algorithm having complete control over the network. As described in the following, $\mathcal{A}$ can add users to the set $\mathcal{U}$ and interact with protocol participants via queries to their oracles. Note that our security model (similar to Bresson, Chevassut and Pointcheval, 2002a; Katz and Shin, 2005; Bohli, Vasco and Steinwandt, 2007) does not deal with the issues of denial-of-service and fault-tolerance; our security definitions aim to prevent honest participants from accepting the group key biased by malicious insiders.

AddUser(U, Λ). If $U \notin \mathcal{U}$, then U with the long-lived (public) key contained in Λ is added to $\mathcal{U}$; Λ may also contain some further information.

Execute($\mathcal{S}$). $\mathcal{A}$ eavesdrops an honest execution of P.Setup between a chosen set of oracles and is given the resulting transcript of P.Setup($\mathcal{S}$).

Send($\prod_U^s, m$). $\mathcal{A}$ sends message m to oracle $\prod_U^s$ and receives the response $\prod_U^s$ would have generated after having (honestly) processed message m. The response may be empty if m is incorrect. The adversary can have $\prod_U^s$ invoking P.Setup with the oracles in $\mathcal{S}$ via a query of the form *Send*('start', $\prod_U^s$, $\mathcal{S}$): $\mathcal{A}$ gets the first message that $\prod_U^s$ would generate in this case.

RevealKey($\prod_U^s$). $\mathcal{A}$ is given the session group key k_U^s, provided $\prod_U^s$ has accepted.

RevealState($\prod_U^s$). $\mathcal{A}$ is given the internal state information state_U^s which includes ephemeral secrets.

Corrupt(U). $\mathcal{A}$ is given the long-lived key LL_U.

Test($\prod_U^s$). $\mathcal{A}$ tests the semantic security of k_U^s. Formally, if $\prod_U^s$ has accepted a bit b is privately flipped and $\mathcal{A}$ is given k_U^s if $b = 1$ and a random string if $b = 0$.

The adversary has two ways of learning LL_U: by asking it – *Corrupt*(U), or by registering it – *AddUser*(U, Λ). For simplicity, in all definitions of security unless otherwise stated, we treat U as corrupted if any of these queries had occurred.

Remark 1. The separation of the queries RevealState and Corrupt/AddUser explicitly provides the possibility for the opening attacks mentioned in the introduction. By asking the RevealState query to an instance oracle $\prod_U^s$, the adversary reads out its internal state, but cannot impersonate honest U in the protocol execution, unless a Corrupt(U) query is asked (in which case all instance oracles $\prod_U^s$ become malicious insiders through possible impersonation actions of $\mathcal{A}$). Thus, just opening a user does not make him malicious. In contrast, simulation-based security models (e.g. Universal Composability/Reactive Simulatability) handle strong corruptions typically as follows: upon corrupting a user the adversary learns all information known to that user and controls him thereafter. Obviously, in the simulation-based models opening attacks (which strengthen the adversary) are currently not modelled.

2.3 Strong AKE-security

In case of strong AKE-security, one must also consider the knowledge of the adversary about long-lived keys and ephemeral secrets of session participants. If the adversary obtains a long-lived key before the session is started then it can impersonate a user, and thus, learn the session key. And, if the adversary is allowed to obtain long-lived keys before the session is finished then it should be restricted from actively using these keys during that time (Katz and Yung, 2003).

On the other hand, the adversary should be allowed to reveal ephemeral secrets of participants before the session starts[1] and after the session is finished (defined as strong forward and weak backward secrecy in Bresson, Manulis and Schwenk, 2007). Note that, if one allows long-lived key corruptions in later sessions, revealing ephemeral secrets during the attacked session would not make sense. In order to model the described requirements for the adversarial knowledge, we define the notion of oracle freshness, extending those given in Bresson, Chevassut and Pointcheval (2002a) and Katz and Yung (2003) by the conditions concerning key registration and opening attacks.

Definition 2 (Oracle Freshness). In the execution of P the oracle $\prod_U^s$ is fresh if all of the following holds:

1 no $U_i \in \text{pid}_U^s$ has been added by $\mathcal{A}$ via a corresponding AddUser query

2 no $U_i \in \text{pid}_U^s$ is asked for a query Corrupt prior to a query of the form Send $(\prod_j^t, m)$ with $U_j \in \text{pid}_U^s$ until $\prod_U^s$ and its partners accept

3 neither $\prod_U^s$ nor any of its partners is asked for a query RevealState until $\prod_U^s$ and its partners accept

4 neither $\prod_U^s$ nor any of its partners is asked for a query RevealKey after having accepted.

We say that a session is fresh if all participating oracles are fresh.

We note that the above definition ensures that if at least one oracle participating in a session is fresh then the whole session is fresh too because freshness of one oracle requires freshness of all its partners. This notion of oracle freshness simplifies the following definition of strong AKE-security for GKE protocols.

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

- after initialisation, $\mathcal{A}$ interacts with instance oracles via queries
- at some point $\mathcal{A}$ asks a test query to a fresh oracle $\prod_U^s$ which has accepted
- $\mathcal{A}$ continues interacting with instance oracles
- when $\mathcal{A}$ terminates, it outputs a bit, which we define to be the output of the game.

We define: $\text{Adv}_{\mathcal{A},\text{P}}^{\text{ake}}(\kappa) := \left|2\Pr[\text{Game}_{\mathcal{A},\text{P}}^{\text{ake}-b}(\kappa) = b] - 1\right|$ and denote with $\text{Adv}_{\text{P}}^{\text{ake}}(\kappa)$ the maximum advantage over all

PPT adversaries $\mathcal{A}$. We say that a GKE protocol P provides strong AKE-security if this advantage is negligible.

We again stress that (strong) AKE-security makes sense for adversaries that are not able to corrupt users and act on their behalf during the attacked session or reveal any ephemeral secrets used in that session – this is guaranteed by the freshness property.

2.4 Strong MA-security

We say that $\prod_U^s$ is a malicious participant/insider if the adversary has previously asked *Corrupt*(U) or *AddUser* (U, Λ). In all other cases, $\prod_U^s$ is honest. The following definition of MA-security unifies the requirements of MA, key confirmation and unknown-key share resilience. It considers malicious participants and allows opening attacks against all honest users at any protocol stage.

Definition 4 (Strong MA-Security). Let P be a correct GKE protocol and $\mathcal{A}$ an adversary who is allowed to *query Send, Execute, RevealKey, RevealState, Corrupt and AddUser*. We denote this interaction as $\text{Game}_{\mathcal{A},\text{P}}^{\text{ma}}(\kappa)$. We say that $\mathcal{A}$ wins if at some point, there exists an honest user U_i whose instance oracle $\prod_i^s$ has accepted with k_i^s and another user $U_j \in \text{pid}_i^s$ that is uncorrupted at the time $\prod_i^s$ accepts such that

1 there is no instance oracle $\prod_j^t$ with $(\text{pid}_j^t, \text{sid}_j^t) = (\text{pid}_i^s, \text{sid}_i^s)$ or

2 there is an instance oracle $\prod_j^t$ with $(\text{pid}_j^t, \text{sid}_j^t) = (\text{pid}_i^s, \text{sid}_i^s)$ that has accepted with $k_j^t \neq k_i^s$.

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

2.5 Strong contributiveness

The following definition models attacks related to key control, contributiveness and unpredictability of group keys in the presence of malicious participants. Informally, we consider an active adversary $\mathcal{A}$ that can add, corrupt and open participants at any stage of the protocol execution in such a way that there exists at least one honest oracle (which may nevertheless be opened!) that accepts the session group key chosen by the adversary. This subsumes key-replication attacks (Krawczyk, 2005) by which honest users are forced to accept a group key from another session.

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

- $\mathcal{A}$ (prepare) interacts with instance oracles via queries
- $\mathcal{A}$ (prepare) outputs $\tilde{k} \in \{0,1\}^\kappa$, and some state information ζ
- the following sets are built: $\mathcal{G}_{\text{us}}$ consisting of all honest used oracles, $\mathcal{G}_{\text{std}}$ consisting of all honest oracles that are in the stand-by state[2], and Ψ consisting of session ids sid_i^t for every $\prod_i^t \in \mathcal{G}_{\text{us}}$
- $\mathcal{A}$ (attack, ζ) interacts with instance oracles via queries
- at the end of this stage $\mathcal{A}$ outputs (s, U).

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

1 $\prod_U^s$ is honest, has terminated accepting $\tilde{k}, \prod_U^s \notin \mathcal{G}_{\text{us}} \setminus \mathcal{G}_{\text{std}}$ and $\text{sid}_U^s \notin \Psi$

2 there are at most n–1 corrupted users U_i having oracles $\prod_i^t$ partnered with $\prod_U^s$.

We define: $\text{Suc}_{\mathcal{A},\text{P}}^{\text{con}}(\kappa) := \Pr[\mathcal{A} \text{ wins in Games}_{\mathcal{A},\text{P}}^{\text{con}}(\kappa)]$ and denote with $\text{Suc}_{\text{P}}^{\text{con}}(\kappa)$ 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.

The first requirement ensures that $\prod_U^s$ belongs to an honest user. The set $\mathcal{G}_{\text{us}} \setminus \mathcal{G}_{\text{std}}$ consists of all oracles that at the end of the prepare stage have already terminated or remain in the processing state. Thus, requiring $\prod_U^s \notin \mathcal{G}_{\text{us}} \setminus \mathcal{G}_{\text{std}}$ prevents the case where $\mathcal{A}$ while being a session participant outputs $\tilde{k}$ for the still running protocol execution which is then accepted by $\prod_U^s$ that participates in the same execution (this is not an attack since participants do not compute group keys synchronously). Similarly, the condition $\text{sid}_U^s \notin \Psi$ prevents that $\mathcal{A}$ while being in the attack stage outputs (s, U) such that $\prod_U^s$ has accepted with $\tilde{k}$ already in the prepare stage. Finally, since in every session id is unique, $\text{sid}_U^s \notin \Psi$ holds if at least one new session has been executed with $\prod_U^s$ in the attack stage. The second requirement allows $\mathcal{A}$ to corrupt at most n–1 (out of totally n) participants in the session where $\prod_U^s$ accepts with $\tilde{k}$.

Note also that U must be honest, but $\mathcal{A}$ is allowed to reveal the internal state of $\prod_U^s$ during the execution of the attack stage (this is because our model separates LL_U from state_U^s). The goal of the adversary is to influence the honest participants to accept the chosen key. Our game appears

stronger than Bohli, Vasco and Steinwandt (2007), since the adversary can open honest users' internal state (furthermore, he can make corruptions in an adaptive manner).

Remark 2. The main difference to the non-malleability definition from Desmedt et al. (2006) is that we allow $\mathcal{A}$ to open honest users during the attacked session, however, at the cost that we do not deal with the ability of $\mathcal{A}$ to bias the probability distribution of the resulting group key (similar to Bohli, Vasco and Steinwandt (2007)). It seems to be hard to achieve this goal if $\mathcal{A}$ corrupts $n-1$ users and opens the last nth honest user, which is allowed by our definition. At least, the commitment techniques used in Desmedt et al. (2006) would not help since committed secrets that become part of the internal state can be revealed.

3 TDH1 protocol with strong security

In this section, we present our constant-round GKE protocol denoted TDH1 and show that it satisfies the strong versions of AKE-, MA-security and contributiveness. Its AKE-security relies on the TDDH assumption, introduced by Kim, Perrig and Tsudik (2004a,b).

3.1 Number-theoretic assumptions

First, we formally specify the TDDH assumption and quantify the reduction to the classical DDH assumption (Boneh, 1998). Our protocol and those (unauthenticated) in Kim, Perrig and Tsudik (2004a,b) require a special multiplicative group $\mathbb{G}$ in which DDH is assumed to be hard and for which there exists an efficient bijection[3] from $\mathbb{G}$ to $\mathbb{Z}_{|\mathbb{G}|}$. Thus, not every DDH-hard group can be used, e.g. no such bijection is known for elliptic curves.

3.1.1 Algebraic group

Let p be a safe prime, i.e. $p = 2q + 1$, with q a κ-bit prime. The set of quadratic residue modulo p is a cyclic group $\hat{\mathbb{G}}$ of order q; let g be a generator: $\hat{\mathbb{G}} = \langle g \rangle$. Consider the following mapping u: $\mathbb{Z}_p^* \to \mathbb{Z}_q$ defined as

$$u(z) := \begin{cases} z \bmod q & \text{if } z \le q \\ (p-z) \bmod q & \text{if } q < z < p. \end{cases}$$

Consider the set $\mathbb{G} := \{u(g^i) \mid i \notin \mathbb{Z}_q\}$. It can be shown (Kim, Perrig and Tsudik, 2004b) that the function $f : x \mapsto u(g^x)$ from $\mathbb{Z}_q$ to $\mathbb{G}$ is a bijection, and that the operation $(a,b) \in \mathbb{G}^2 \mapsto u(ab \bmod p)$ is a group law on $\mathbb{G}$. Since, $\mathbb{G} = \mathbb{Z}_q$ (as sets), we can define the exponentiation $a^b := u(a^b \bmod p)$ for all $a, b \in \mathbb{G}$. Also, due to the fact that f is a bijection, we have $f(x)$ is random, uniform in $\mathbb{G}$ as soon as x is so.

Finally, it is believed (Boneh, 1998) that the DDH assumption holds in $\mathbb{G}$, that is, the distributions of $(g^{x_1}, g^{x_2}, g^{x_1 x_2})$ and $(g^{x_1}, g^{x_2}, g^{r})$ are computationally indistinguishable for $x_1, x_2, r \in_R \mathbb{G}$.

3.1.2 Tree Decisional Diffie–Hellman assumption

Let $\mathcal{T}_n$ be the set of all full[4] binary trees with n leaves. For a $T_n \in \mathcal{T}_n$ of depth d_{T_n}, each node is identified via a label $\langle l, v \rangle$, where $l \in [0, d_{T_n}]$ is the level of the node and v its position within this level: the position is such that the child nodes of $\langle l, v \rangle$ (if present) are labelled $\langle l+1, 2v \rangle$ and $\langle l+1, 2v+1 \rangle$ (this implies that the nodes positions are in $[0, 2^l-1]$, but may be not contiguous). The root node is labelled $\langle 0, 0 \rangle$. In the following, we will denote $T_n \backslash \langle 0, 0 \rangle$ by T_n^*. The set of leaf nodes and the set of internal nodes of T_n are defined as (respectively):

$$\mathrm{LN}_{T_n} := \{\langle l,v \rangle \mid \langle l,v \rangle \in T_n, \quad \langle l+1,2v \rangle \notin T_n, \quad \langle l+1,2v+1 \rangle \notin T_n\},$$

$$\mathrm{IN}_{T_n} := \{\langle l,v \rangle \mid \langle l,v \rangle \in T_n^*, \quad \langle l+1,2v \rangle \notin T_n, \quad \langle l+1,2v+1 \rangle \notin T_n\}.$$

For a set X of n randomly chosen variables $x_{\langle l, v \rangle} \in_R \mathbb{G}$, with $\langle l, v \rangle \in \mathrm{LN}_{T_n}$, we (recursively) define for each $\langle l, v \rangle \in \mathrm{IN}_{T_n}$:

$$x_{\langle l,v \rangle} = g^{x_{\langle l+1,2v \rangle} x_{\langle l+1,2v+1 \rangle}}, \text{ and}$$

$$\mathrm{TDH}_{T_n}(X) = \left\{ \left(\langle l,v \rangle, g^{x_{\langle l,v \rangle}} \right) \right\}_{\langle l,v \rangle \in T_n^*}.$$

In addition, for a randomly chosen $r \in_R \mathbb{G}$, we define the tuples $\mathrm{TDDH}^*_{T_n}(X)$ and $\mathrm{TDDH}^{\$}_{T_n}(X)$ as follows:

$$\mathrm{TDDH}^*_{T_n}(X) = \mathrm{TDH}_{T_n}(X) \bigcup (\langle 0,0 \rangle, g^{x_{\langle 1,0 \rangle} x_{\langle 1,1 \rangle}}),$$

and

$$\mathrm{TDDH}^{\$}_{T_n}(X,r) = \mathrm{TDH}_{T_n}(X) \bigcup (\langle 0,0 \rangle, g^{r}).$$

The TDDH assumption states that the respective distributions of these tuples induced by uniform choices of r and the $x_{\langle l, v \rangle}$, for $\langle l,v \rangle \in \mathrm{LN}_{T_n}$ are computationally indistinguishable.

Definition 6 (TDDH Assumption). For all $n > 1$, any $T_n \in \mathcal{T}_n$, any group $\mathbb{G}$, and any PPT algorithm $\mathcal{A}$, the distinguishing advantage $\mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\mathcal{A})$, defined as follows, is negligible (in $\kappa = \log|\mathbb{G}|$):

$$\left|\Pr_X\left[\mathcal{A}\left(\mathrm{TDDH}^*_{T_n}(X)\right)=1\right]-\Pr_{X,r}\left[\mathcal{A}\left(\mathrm{TDDH}^{\$}_{T_n}(X,r)\right)=1\right]\right|.$$

Theorem 1 (DDH ⇔ TDDH). The TDDH problem in $\mathbb{G}$ is polynomially equivalent to the DDH problem in $\mathbb{Z}_q$, and we have: $\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa) \le \mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\kappa) \le (2n-3)\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa)$.

The proof appears in Appendix A and is more general than those in Kim, Perrig and Tsudik (2004a,b) that focus only on complete and linear binary trees. For $n = 2$, we get the classical DDH assumption in $\mathbb{G}$ with the advantage $\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\mathcal{A}')$ defined as

$$\left|\Pr_X\left[\mathcal{A}'(\mathrm{DDH}^*(X))=1\right]-\Pr_{X,r}\left[\mathcal{A}'(\mathrm{DDH}^{\$}(X,r))=1\right]\right|.$$

3.2 Light description of TDH1

The main mechanism of the protocol is that of Kim, Perrig and Tsudik (2004a,b), so we first recall it. The differences will be in message authentication, key derivation and applied erasure technique to prevent the ephemeral session secrets from being leaked once the session is finished. Erasure of the internal state can be seen as a general method to achieve AKE-security in the presence of opening attacks for static GKE protocols.

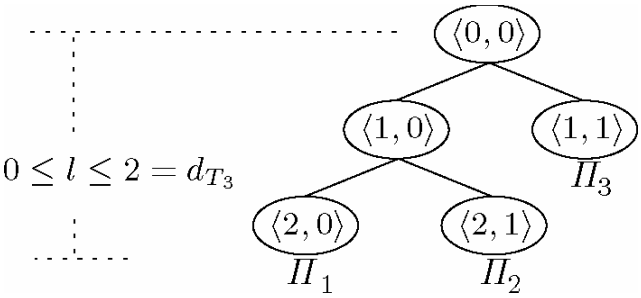
3.2.1 The setup operation

Every oracle is assigned to a leaf node of a so-called linear binary tree T_n: a full binary tree with one leaf at each level, except for the deepest one with two leaves (see Figure 1). In other words, $T^*_n := \{\langle l,v\rangle\}_{l\in[1,n-1],v\in[0,1]}$.

Round 1 – All. Each oracle at position $\langle l_i, v_i\rangle$ chooses a secret exponent $x_{\langle li,\ vi\rangle} \in_R \mathbb{G}$ and broadcasts $y_{\langle l_i,\, v_i\rangle} := g^{x\langle l_i,\, v_i\rangle}$.

Round 2 – First player. $\prod_1$ at position $\langle n-1,0\rangle$ is able to build a set X_1 of secret values for each node $x_{\langle l,0\rangle}$ in its path up to the root $\langle 0, 0\rangle$. This is because for each internal node $x_{\langle l,0\rangle} = (y_{\langle l+1,1\rangle})^{x_{\langle l+1,0\rangle}}$ Then, $\prod^s_1$ computes the set $\hat{Y}$ consisting of $y_{\langle l,0\rangle} := g^{x_{\langle l,0\rangle}}$ for each previously computed internal node's secret value $x_{\langle l,0\rangle}$ except for $x_{\langle 0,0\rangle}$, and broadcasts $\hat{Y}$.

Figure 1 Example of a linear binary tree T_3 for the group $\mathcal{G} = \{\prod_1, \prod_2, \prod_3\}$



Round 3 – No communication. Upon receiving $\hat{Y}$, all other oracles $\prod_{i\neq 1}$ are able to compute their own set X_i consisting of all secret values $x_{\langle l,v\rangle}$ in their paths up to the root. Hence, every oracle finally learns $x_{\langle 0,0\rangle}$. We emphasise that $x_{\langle 0,0\rangle}$ is never exposed, and that there is no $y_{\langle 0,0\rangle}$ in the protocol (see description of function TDH1_Exp*(l, X) below).

3.2.2 Group key confirmation and derivation

To derive the session group key K, each participant iteratively computes a sequence of values $\rho_0, \ldots, \rho_n$ using a Pseudo-Random Function (PRF) f with a public value v_0 as input. The key (secret seed) of f is initially set to $x_{\langle 0,0\rangle}$, and is changed in each invocation of f by embedding successive nonces using an appropriate one-way permutation π. These nonces are provided by participants during the protocol execution.

Intuitively, these successive evaluations of f and π prevent malicious participants from influencing values of the PRF keys and ensures contributiveness for the intermediate value ρ_n, which is then used as a seed for f to derive the key confirmation token μ (on input a constant public value v_1) and the actual session group key K (on input another constant public value $v_2 \neq v_1$). Prior to accepting K:

1 participants exchange and verify signatures on μ to ensure MA-security (similar to Katz and Shin (2005))
2 erase (Crescenzo et al., 1999) all ephemeral secrets, used to obtain K from their internal states, to achieve strong AKE-security.

3.3 Detailed description of TDH1

3.3.1 Preliminary notations

We assume that long-lived keys $\mathrm{LL}_i = (\mathrm{sk}_i,\ \mathrm{pk}_i)$ are generated via Σ. Gen (1^κ), where Σ = (Gen, Sign, Verify) is an existentially unforgeable (under chosen message attacks) digital signature scheme. We define the following key exchange functions:

TDH1_Exp($x_{\langle l,\ v\rangle}$). Simple exponentiation. The function returns $y_{\langle l,\, v\rangle} := g^{x_{\langle l,v\rangle}}$.

TDH1_Pick(1^κ). The function returns a randomly chosen secret exponent $x_{\langle l,v\rangle} \in_R \mathbb{G}$ and the corresponding public value $y_{\langle l,v\rangle} :=$ TDH1_Exp($x_{\langle l,v\rangle}$).

TDH1_Exp*(l, X) where $X = \{x_{\langle j,0\rangle}\}_{1 \le j \le l}$: Computation of corresponding public values for secret exponents in X. The function returns $Y := \{y_{\langle j,0\rangle}\}$ where each $y_{\langle j,0\rangle} :=$ TDH1_Exp($x_{\langle j,0\rangle}$).

TDH1_Up(l, v, $x_{\langle l,v\rangle}$, $y_{\langle l,1-v\rangle}$, Y), where $Y := \{y_{\langle j,1\rangle} \mid j \in [1, l-1]\}$: iterative computation of the Diffie–Hellman values up the tree starting at position $\langle l, v\rangle$. The function computes $x_{\langle l-1,0\rangle} := (y_{\langle l,1-v\rangle})^{x_{\langle l,v\rangle}}$, and returns

$$X := \{x_{\langle l,v\rangle}, x_{\langle l-1,0\rangle}\} \cup \{x_{\langle j,0\rangle} := (y_{\langle j+1,1\rangle})^{x_{\langle j+1,0\rangle}} \mid y_{\langle j+1,1\rangle} \in Y,\ \forall j = l-2, \ldots, 0\}.$$

Let $F := \{\{f_k\}_{k \in \{0,1\}^\kappa}\}_{\kappa \in \mathbb{N}}$ be a collision-resistant PRF ensemble with domain and range $\{0,1\}^\kappa$ (see Appendix B for definitions). Let $\pi : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a one-way permutation. We denote by v_0; v_1 and v_2 three distinct, public values in $\{0,1\}^\kappa$. The following function is used to compute the intermediate value K and the key confirmation token μ.

TDH1_Con$(x_{\langle 0,0\rangle}, r_1 \mid \ldots \mid r_n)$: The function computes $\rho_0 := f_{x_{\langle 0,0\rangle}}(v_0)$, and each $\rho_l := f_{\rho_{l-1} \oplus \pi(r_l)}(v_0)$ for all $l = \{1, \ldots, n\}$. Let $K := \rho_n$. Finally, the function computes $\mu := f_K(v_1)$, and returns (K, μ).

3.3.2 The protocol TDH1.Setup

In the following, we assume that an oracle aborts without accepting if any performed check fails.

Round 1. Given the tree structure T_n, each oracle $\prod_i$ proceeds as follows:

- pick at random nonce $r_i \in_R \{0,1\}^\kappa$
- invoke TDH1_Pick(1^κ) to generate a secret exponent $x_{\langle l_i,v_i\rangle}$; and the value $y_{\langle l_i,v_i\rangle} = g^{x_{\langle l_i,v_i\rangle}}$
- invoke Σ.Sign to obtain a signature σ_i on $0 \mid y_{\langle l_i,v_i\rangle} \mid r_i \mid \mathrm{pid}_i$ using the private key sk_i
- broadcast $U_i \mid 0 \mid y_{\langle l_i,v_i\rangle} \mid r_i \mid \sigma_i$.

Round 2. Each oracle $\prod_i$ proceeds as follows:

- check if Σ.Verify$(pk_j, 0 \mid y_{\langle l_j,v_j\rangle} \mid r_j \mid \mathrm{pid}_i, \sigma_j) = 1$ for $j \neq i$; check if $|r_j| = \kappa$ for $j \neq i$
- define $\mathrm{sid}_i := r_1 \mid \ldots \mid r_n$ and $Y_i := \{y_{\langle 1,1\rangle}\}_{l = l_i - 1, \ldots, 1}$. In addition, $\prod_1$ does the following
- compute $X_1 :=$ TDH1_Up$(n-1, 0, x_{\langle n-1,0\rangle}, y_{\langle n-1,1\rangle}, Y_1)$ and $\hat{Y} :=$ TDH1_Exp* $(n-2, X_1)$
- invoke Σ.Sign to obtain a signature σ_1' on $1 \mid \hat{Y} \mid \mathrm{sid}_1 \mid \mathrm{pid}_1$ using the private key sk_1
- broadcast $U_1 \mid 1 \mid \hat{Y} \mid \sigma_1'$.

Round 3. Each $\prod_i$ with $i > 1$ proceeds as follows:

- check if Σ.Verify$(pk_1, 1 \mid \hat{Y} \mid \mathrm{sid}_i \mid \mathrm{pid}_i, \sigma_1') = 1$
- compute X_i as TDH1_Up $(l_i, v_i, x_{\langle l_i,v_i\rangle}, y_{\langle l_i,1-v_i\rangle}, Y_i)$.

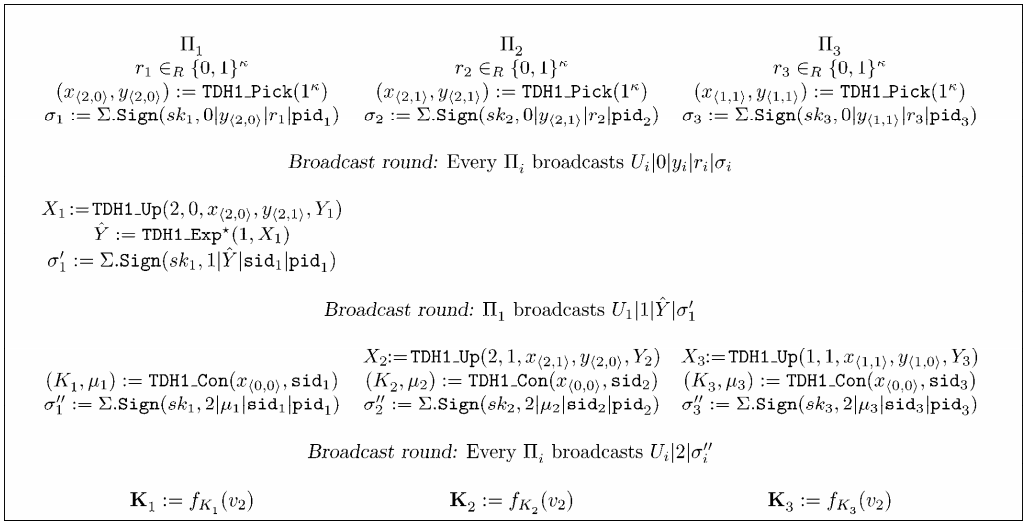
Then, every oracle (including $\prod_1$) does the following:

- compute both K_i and μ_i using TDH1_Con($x_{\langle 0,0\rangle}$, sid_i); [note that $x_{\langle 0,0\rangle} \in X_i$]
- erase any private information from state_i (including all $x_{\langle l,v\rangle}$, and $\rho_0, \ldots, \rho_n$) except for K_i
- invoke Σ.Sign to obtain a signature σ_i'' on $2 \mid \mu_i \mid \mathrm{sid}_i \mid \mathrm{pid}_i$ using the private key sk_i
- broadcast $U_i \mid 2 \mid \sigma_i''$.

Group Key Computation. When $\prod_i$ receives $U_j \mid 2 \mid \sigma_j''$ from all other oracles, it proceeds as follows:

- check if Σ.Verify$(pk_j, 2 \mid \mu_i \mid \mathrm{sid}_i \mid \mathrm{pid}_i, \sigma_j'') = 1$; Compute $\mathrm{K}_i := f_{K_i}(v_2)$
- erase any private information from state_i (including K_i), and accept with K_i.

Figure 2 Example of operation Tree Diffie–Hellman 1 setup with $\mathcal{G} = \{\prod_1, \prod_2, \prod_3\}$



Public values: $\text{sid}_i = r_1|r_2|r_3$, $Y_1 = Y_2 = \{y_{1,1}\}$, $Y_3 = \theta$, $\hat{Y} = \{y_{1,0}\}$, where $y_{\langle 1,1\rangle} = g^{x_{\langle 1,1\rangle}}$, $y_{\langle 1,0\rangle} = g^{x_{\langle 1,0\rangle}}$. Secret values: $X_1 = \{x_{2,0}, x_{1,0}, x_{0,0}\}$, $X_2 = \{x_{2,1}, x_{1,0}, x_{0,0}\}$, $X_3 = \{x_{1,1}, x_{0,0}\}$ where $x_{\langle 0,0\rangle} = g^{x_{\langle 1,0\rangle} x_{\langle 1,1\rangle}}$, $x_{\langle 1,0\rangle} = g^{x_{\langle 2,0\rangle} x_{\langle 2,1\rangle}}$ and $x_{\langle 2,0\rangle}, x_{\langle 2,1\rangle}, x_{\langle 1,1\rangle} \in_R \mathbb{G}$

Figure 2 sketches the execution of TDH1.Setup for three participants (necessary checks and erasure steps are omitted). In this example, oracle $\prod_1$ builds $X_1 = \{x_{\langle 2,0\rangle}, x_{\langle 1,0\rangle}, x_{\langle 0,0\rangle}\}$ with $x_{\langle 1,0\rangle} := (y_{\langle 2,1\rangle})^{x_{\langle 2,0\rangle}}$ and $x_{\langle 0,0\rangle} := (y_{\langle 1,1\rangle})^{x_{\langle 1,0\rangle}}$ (remember that $y_{\langle 2,1\rangle}$ and $y_{\langle 1,1\rangle}$ were received in the previous round). The set broadcasted by $\prod_1$ is $\hat{Y} = \{y_{\langle 1,0\rangle}\}$ where $y_{\langle 1,0\rangle} := g^{x_{\langle 1,0\rangle}}$. In the third round oracle $\prod_2$ computes $X_2 = \{x_{\langle 2,1\rangle}, x_{\langle 1,0\rangle}, x_{\langle 0,0\rangle}\}$, where $x_{\langle 1,0\rangle} := (y_{\langle 2,0\rangle})^{x_{\langle 2,1\rangle}}$, and $x_{\langle 0,0\rangle} := (y_{\langle 1,1\rangle})^{x_{\langle 1,0\rangle}}$. In parallel, oracle $\prod_3$ computes $X_3 = \{x_{\langle 1,1\rangle}, x_{\langle 0,0\rangle}\}$ where $x_{\langle 0,0\rangle} := (y_{\langle 1,0\rangle})^{x_{\langle 1,1\rangle}}$. Hence, every oracle is in possession of $x_{\langle 0,0\rangle}$. Finally, all three oracles can compute the intermediate value K, i.e. $\rho_0 := f_{x_{\langle 0,0\rangle}|\kappa}(v_0), \rho_1 := f_{\rho_0 \oplus \pi(r_1)}(v_0), \rho_2 := f_{\rho_1 \oplus \pi(r_2)}(v_0)$, and $K = \rho_3 := f_{\rho_2 \oplus \pi(r_2)}(v_0)$, and the key confirmation token $\mu := f_K(v_1)$. Note that the value $x_{\langle 0,0\rangle}$ is never sent over the public channel, but computed locally by all participants upon receiving enough information. Furthermore, there exists no $y_{\langle 0,0\rangle}$ in the protocol.

3.4 Security and performance of TDH1

In our security proofs following the specification of our model, we consider that ephemeral secret information kept in state_U is always independent of the long-lived key sk_U. That is, in each session, state_U contains X_U consisting of all secrets $x_{\langle l,v\rangle}$ known to $\prod_U, \rho_0, \ldots, \rho_n$, and possibly some (implementation specific) temporary variables used to compute these values. Furthermore, we assume that the signing algorithm Σ.Sign which implicitly uses sk_U is executed under the same protection mechanism as sk_U, e.g. in a smart card as in Bresson, Chevassut and Pointcheval (2002a, although smart cards have limited resources we observe that in TDH1.Setup each oracle has to generate at most three signatures). This is important since the signing algorithm may generate some randomness which should also be protected from being revealed via a RevealState query; otherwise the adversary may be able to obtain some information about sk_U.

The following theorems show that TDH1 satisfies the requirements of strong AKE-, MA-security and contributiveness; the last two also under consideration of insider attacks. In all theorems, q_s is the total number of executed protocol sessions during the corresponding attack game.

Theorem 2 (Strong AKE-Security of TDH1). If Σ is existentially unforgeable under chosen message attacks, F is pseudo-random, and $\mathbb{G}$ is TDDH-hard then TDH1 provides strong AKE-security, and

$$\text{Adv}^{\text{ake}}_{\text{TDH1}}(\kappa) \le 2N\text{Suc}^{\text{euf-cma}}_{\Sigma}(\kappa) + \frac{Nq_s^2}{2^{\kappa-1}} + 2q_s\text{Adv}^{\text{TDDH}}_{T_N,G}(\kappa)$$
$$+2(N+3)q_s\text{Adv}^{\text{prf}}_{\text{F}}(\kappa).$$

Proof (Sketch). We define a sequence of games G_i, $i = 0, \ldots, 7$ with the adversary $\mathcal{A}$ against the AKE-security of TDH1. In each game, we denote $\text{Win}^{\text{ake}}_i$ the event that the bit b' output by $\mathcal{A}$ is identical to the randomly chosen bit b in game G_i.

Game G_0. This game is the real game $\text{Game}^{\text{ake-}b}_{\mathcal{A},\text{TDH1}}(\kappa)$ where we use a simulator Δ to maintain set $\mathcal{U}$, simulate the execution of the protocol (on behalf of uncorrupted users), and answer all queries of $\mathcal{A}$.

Game G_1. This game is identical to Game G_0 with the only exception that Δ fails and bit b' is set at random if $\mathcal{A}$ asks a send query on some $U_i \mid \text{seqn} \mid m \mid \sigma$, with seqn ∈ {0, 1, 2} and σ a valid signature on m, that has not been previously output by an oracle $\prod_i^s$ before querying *Corrupt*(U_i) (note that *Corrupt* queries can be generally asked after the Test query) or *AddUser*(U_i, Λ).

In other words, the simulation fails if $\mathcal{A}$ outputs a successful forgery; such event is denoted Forge. A classical reductionist argument (see for instance Bresson et al. (2001) shows that in that case we can build a forger against the signature scheme and upper-bound the probability difference between G_1 and G_0 by $N\text{Suc}_{\Sigma}^{\text{euf-cma}}(\kappa)$.

Game G_2. This game is identical to Game G_1 except that Δ fails if two instances of an honest user used the same nonce twice. Since, there are N users and at most q_s sessions the difference between games can be upper-bounded by $(Nq_s^2)/2^{\kappa}$.

This game ensures the uniqueness of sid_i^s and excludes replay attacks on the last two messages of TDH1.

Game G_3. In this game, we add the following rule: Δ chooses $q_s^* \in [1, q_s]$ and aborts if the test query does not occur in the q_s^*th session. Let Q be the event that this guess for q_s^* is correct and Pr[Q] = 1/q_s. Thus, we get

$$\Pr\left[\text{Win}_3^{\text{ake}}\right] = \Pr\left[\text{Win}_3^{\text{ake}} \mid Q\right]\Pr[Q] + \Pr\left[\text{Win}_3^{\text{ake}} \mid \neg Q\right]\Pr[\neg Q]$$
$$= \Pr\left[\text{Win}_2^{\text{ake}}\right]\frac{1}{q_s} + \frac{1}{2}\left(1 - \frac{1}{q_s}\right).$$

This implies,

$$\Pr\left[\text{Win}_2^{\text{ake}}\right] = q_s\left(\Pr\left[\text{Win}_3^{\text{ake}}\right] - \frac{1}{2}\right) + \frac{1}{2}.$$

Game G_4. This game is identical to Game G_3 except that Δ is given a tuple from the real $\text{TDDH}_{T_N}^*$ distribution (as specified in Section 3.1.2) where T_N is a linear tree. In all sessions, except the q_s^*th one, Δ simulates the honest participants as specified by the protocol. In the q_s^*th session with n participants Δ injects public values $g^{x_{\langle l,v\rangle}}$ from $\text{TDDH}_{T_N}^*$ into the protocol execution. Note that in the q_s^*th session the group size n might be smaller than N, thus the simulator will use the subtree T_n which is composed of T_N's nodes from level 0 to level $n-1$. The idea is to assign $\prod_1^s$ to the (internal) node $\langle n-1, 0\rangle$, $\prod_2^s$ to the leaf node $\langle n-1, 1\rangle$, …, $\prod_2^s$ to the leaf node $\langle 1, 1\rangle$, and to use for each node $\langle l, v\rangle \in T_n \setminus \langle 0, 0\rangle$ public values $g^{x_{\langle l,v\rangle}}$ taken from the given $\text{TDDH}_{T_N}^*$ distribution. The secret value $x_{\langle 0,0\rangle}$ used for the key confirmation is also taken from this distribution. Since, the q_s^*th session is fresh, no *RevealState* queries to $\prod_i^s$ or to any of its partners have been asked (Δ would not be able to answer them since it does not know the secret values $x_{\langle l,v\rangle}$ of internal and leaf nodes). Of course, in all other sessions *RevealState* queries can be easily answered. Since, $\text{TDDH}_{T_N}^*$ is a real distribution we conclude that this game is a 'bridging step' (as named in Shoup, 2006) and $\Pr\left[\text{Win}_4^{\text{ake}}\right] = \Pr\left[\text{Win}_3^{\text{ake}}\right]$.

Game G_5. This game is identical to Game G_4 except that Δ is given a tuple from the random $\text{TDDH}_{T_N}^{\$}$ distribution. Thus, for honest players, the secret $x_{\langle 0,0\rangle}$ is simulated using the provided random element g^r. Obviously, $\left|\Pr\left[\text{Win}_5^{\text{ake}}\right] - \Pr\left[\text{Win}_4^{\text{ake}}\right]\right| \le \text{Adv}_{T_N,G}^{\text{TDDH}}(\kappa)$.

Game G_6. This game is identical to Game G_5 except that in the q_s^*th session Δ replaces f by a truly random function, implying the uniform distribution of $K = \rho_n$. Considering $n \le N$, we obtain by a 'hybrid argument'[5]

$$\left|\Pr\left[\text{Win}_6^{\text{ake}}\right] - \Pr\left[\text{Win}_5^{\text{ake}}\right]\right| \le (N+1)\,\text{Adv}_F^{\text{prf}}(\kappa).$$

Game G_7. This is the continuation of the hybrid argument, but for clarity we specify a separate game; the confirmation token μ and the session key K are replaced by two random κ-bit values, s.t., $\left|\Pr\left[\text{Win}_7^{\text{ake}}\right] - \Pr\left[\text{Win}_6^{\text{ake}}\right]\right| \le 2\text{Adv}_F^{\text{prf}}(\kappa)$. Since, K is uniform: $\Pr\left[\text{Win}_7^{\text{ake}}\right] = 1/2$. Combining the previous equations, we conclude the proof. □

Theorem 3 (Strong MA-Security of TDH1). If Σ is existentially unforgeable under chosen message attacks and F is collision-resistant then TDH1 provides strong MA-security, and

$$\text{Suc}_{\text{TDH1}}^{\text{ma}}(\kappa) \le N\text{Suc}_{\Sigma}^{\text{euf-cma}}(\kappa) + \frac{Nq_s^2}{2^{\kappa}} + q_s\text{Suc}_F^{\text{coll}}(\kappa).$$

Proof (Sketch). We define a sequence of games $G_i, i = 0, \ldots, 2$ and corresponding events Win_i^{ma} meaning that $\mathcal{A}$ wins in G_i.

Game G_0. This is the real game $\text{Game}_{\text{TDH1}}^{\text{ma}}(\kappa)$ played between a simulator Δ and $\mathcal{A}$. The goal of $\mathcal{A}$ is to achieve that there exists an honest user U_i whose corresponding oracle $\prod_i^s$ accepts with K_i^s and another user $U_j \in \text{pid}_i^s$ who is uncorrupted at the time $\prod_i^s$ accepts and either does not have a corresponding oracle $\prod_j^t$ with $\left(\text{pid}_j^t, \text{sid}_j^t\right) = \left(\text{pid}_i^s, \text{sid}_i^s\right)$ or has such an oracle, but this oracle accepts with $K_j^t \ne K_i^s$.

Game G_1. Here, we proceed as in the previous proof and eliminate executions in which forgeries occur, obtaining $\left|\Pr\left[\mathrm{Win}_1^{\mathrm{ma}}\right]-\Pr\left[\mathrm{Win}_0^{\mathrm{ma}}\right]\right| \le N\mathrm{Suc}_{\Sigma}^{\text{euf-cma}}(\kappa)$.

Game G_2. This game is identical to Game G_1 except that Δ fails if a nonce r_i is used by any uncorrupted user's oracle $\prod_i^s$ in two different sessions. Similar to the previous proof we get $\left|\Pr\left[\mathrm{Win}_2^{\mathrm{ma}}\right]-\Pr\left[\mathrm{Win}_1^{\mathrm{ma}}\right]\right| \le Nq_s^2/2^{\kappa}$. Having excluded forgeries and replay attacks we follow that for every user $U_j \in \mathrm{pid}_i^s$ that is uncorrupted at the time $\prod_i^s$ accepts there exists a corresponding instance oracle $\prod_j^t$ with $\left(\mathrm{pid}_j^t, \mathrm{sid}_j^t\right) = \left(\mathrm{pid}_i^s, \mathrm{sid}_i^s\right)$. Thus, according to Definition 4 $\mathcal{A}$ wins in this game only if any of these oracles has accepted with $\mathrm{K}_j^t = f_{K_j^t}(v_2) \ne f_{K_i^s}(v_2) = \mathrm{K}_i^s$. However, the validity of signatures on tokens μ_i and μ_j implies that $\mu_I = \mu_j$. Thus, the probability difference between these games is upper-bounded by $q_s \Pr\left[\mathrm{K}_j^t \ne \mathrm{K}_i^s \wedge f_{K_j^t}(v_1) = f_{K_i^s}(v_1)\right]$, which is equivalent to $q_s \Pr\left[f_{K_j^t}(v_2) \ne f_{K_i^s}(v_2) \wedge f_{K_j^t}(v_1) = f_{K_i^s}(v_1)\right]$, and results in $q_s \mathrm{Suc}_F^{\mathrm{coll}}(\kappa)$.

Combining the previous equations, we get the desired result. □

Theorem 4 (Strong Contributiveness of TDH1). If F is collision-resistant pseudo-random and π is one-way then TDH1 provides strong contributiveness, and

$$\mathrm{Suc}_{\mathrm{TDH1}}^{\mathrm{con}}(\kappa) \le \frac{Nq_s^2 + Nq_s + 2q_s}{2^{\kappa}} + (N+2)q_s\mathrm{Suc}_F^{\mathrm{coll}}(\kappa) + q_s\mathrm{Adv}_F^{\mathrm{prf}}(\kappa) + Nq_s\mathrm{Suc}_{\pi}^{\mathrm{ow}}(\kappa).$$

Note that in TDH1 the adversary is able to enforce the resulting value for $x_{\langle 0,0\rangle}$ by opening oracles of honest users during the protocol execution. More precisely, $\mathcal{A}$ can enforce that the same $x_{\langle 0,0\rangle}$ is computed by the oracles of some uncorrupted user in two different sessions. To show this, assume for simplicity three participants: $\prod_1^s$, $\prod_2^s$ and $\prod_3^s$, and consider that $\prod_1^s$ and $\prod_3^s$ are malicious (corrupted) whereas $\prod_2^s$ is honest. We consider two different sessions: sessions A and B, whereby session B takes place later than A. In both sessions, the tree is as in Figure 1. Assume that in session A, all oracles behave as specified in the protocol except that neither $\prod_1^s$ nor $\prod_3^s$ erase their states. At some point before session B is started, the adversary $\mathcal{A}$ (that can impersonate $\prod_1^s$ and $\prod_3^s$) computes $z := x_{\langle 1,0\rangle}x_{\langle 1,1\rangle}$ where $x_{\langle 1,0\rangle}$ is a value computed by $\prod_1^s$ and $x_{\langle 1,1\rangle}$ is the exponent chosen by $\prod_3^s$, both in session A. Obviously, g^z equals to $x_{\langle 0,0\rangle}$ computed in session A. The goal of $\mathcal{A}$ is to inuence honest $\prod_2^{s'}$ to compute the same $x_{\langle 0,0\rangle}$ in session B. In session B, the exponent $x_{\langle 2,1\rangle}$ used by honest $\prod_2^{s'}$ is likely to be different compared to session A. To proceed with the attack $\mathcal{A}$ waits for $\prod_2^{s'}$ to broadcast $y_{\langle 2,1\rangle} = g^{x_{\langle 2,1\rangle}}$ in session B (note the communication is asymmetric). Then, the adversary opens the oracle holding node $x_{\langle 2,1\rangle}$ (via the RevealState($\prod_2^{s'}$) query); chooses on behalf of $\prod_1^{s'}$ $x_{\langle 2,0\rangle}$ truly at random, computes $x_{\langle 1,0\rangle} := g^{x_{\langle 2,0\rangle}x_{\langle 2,1\rangle}}$ and $x_{\langle 1,1\rangle} := z / x_{\langle 1,0\rangle}$. To complete the attack, $\mathcal{A}$ broadcasts $y_{\langle 2,0\rangle} := g^{x_{\langle 2,0\rangle}}$ and $y_{\langle 1,1\rangle} := g^{x_{\langle 1,1\rangle}}$ on behalf of $\prod_1^{s'}$ and $\prod_3^{s'}$, respectively. It is easy to check that $\prod_2^{s'}$ computes $x_{\langle 0,0\rangle} = g^z$ in session B.

In our proof of Theorem 4, we show that despite of being able to enforce $x_{\langle 0,0\rangle}$ the adversary is still unable to enforce the resulting session group key K. We make use of the following 'difference lemma'.

Lemma 1. Let A, B, C be events defined in some probability distribution, and suppose that $\Pr[\mathrm{B}] = \Pr[\mathrm{A} \mid \mathrm{C}]$. Then,

$$\Pr[\mathrm{A}] - \Pr[\mathrm{B}] \le \Pr[\neg\mathrm{C}].$$

The proof follows from the equation:

$$\begin{aligned}\Pr[\mathrm{A}] &= \Pr[\mathrm{A}|\mathrm{C}]\Pr[\mathrm{C}] + \Pr[\mathrm{A}|\neg\mathrm{C}]\Pr[\neg\mathrm{C}] \\ &= \Pr[\mathrm{B}]\Pr[\mathrm{C}] + \Pr[\mathrm{A}|\neg\mathrm{C}]\Pr[\neg\mathrm{C}] \\ &\le \Pr[\mathrm{B}] + \Pr[\neg\mathrm{C}].\end{aligned}$$

With this lemma, we can define sequence games based on condition events. In game G_{i+1} constructed from G_i with respect to some appropriate condition event C the event Win_{i+1} is defined as $\mathrm{Win}_i|\mathrm{C}$. Then, according to Lemma 1 $\Pr[\mathrm{Win}_i] - \Pr[\mathrm{Win}_{i+1}] \le \Pr[\neg C]$. In order to estimate the probability distance between G_i and G_{i+1} it is sufficient to compute $\Pr[\neg C]$. Note that G_i and G_{i+1} proceed identical from the adversarial perspective. Therefore, it is not necessary for the simulator to detect whether this condition event occurs or not (this in contrast to failure events, used for example in G_1 of Theorem 2). Furthermore, by conditioning the success of the adversary with C we do not restrict the adversarial strategy. Note that the inequality

$$\begin{aligned}\Pr[\mathrm{Win}_i] &= \Pr[\mathrm{Win}_{i+1}]\Pr[\mathrm{C}] + \Pr[\mathrm{Win}_i \mid \neg\mathrm{C}]\Pr[\neg\mathrm{C}] \\ &\le \Pr[\mathrm{Win}_{i+1}] + \Pr[\neg\mathrm{C}]\end{aligned}$$

considers Win_{i+1} and $\mathrm{Win}_i \mid \neg\mathrm{C}$, and so focusing on one strategy represented by $\mathrm{Win}_{i+1}=\mathrm{Win}_i \mid \mathrm{C}$ in G_{i+1} does not rule out all other strategies represented by $\mathrm{Win}_i \mid \neg\mathrm{C}$ because of the total probability $\Pr[\mathrm{Win}_i] \le \Pr[\mathrm{Win}_{i+1}] + \Pr[\neg\mathrm{C}]$.

The main idea of the following proof is to use the fact that every honest oracle $\prod_j^s \in \mathcal{G}, i \in [1,n]$ computes the sequence $\rho_1, \ldots, \rho_n$ prior to the acceptance of K so that each

ρ_1, $l \in [1,n]$ depends on the previously computed ρ_{l-1}. We consider the probability that for an honest $\prod_{i*}^{s}$ the adversary $\mathcal{A}$ is able to enforce any of the values $\rho_{i*}, \ldots, \rho_n$ (note that $K = \rho_n$), or K computed by the collision-resistant PRF *f*. This is equivalent to the event that in the prepare stage $\mathcal{A}$ is able to output any $\rho_{i*}, \ldots, \rho_n$, or K which $\prod_{i*}^{s}$ computes in any session of the attack stage. On the other hand, applying Lemma 1 in our proof we do also consider the upper-bound for the success probability of the adversary in case that its strategy differs from inuencing any value in $\rho_{i*}, \ldots, \rho_n$.

Proof (of Theorem 4, Sketch). Assume that an adversary $\mathcal{A}$ from Definition 5 wins in $\mathrm{Game}^{\mathrm{con}}_{\mathcal{A},\mathrm{TDH1}}(\kappa)$ (which event we denote $\mathrm{Win}^{\mathrm{con}}$). Then, at the end of the stage prepare it returned $\tilde{\mathrm{K}}$ such that in the stage attack there exist $i^* \in [1, n]$ and an honest oracle $\prod_{i*}^{s} \in \mathcal{G}$ that accepts with $\mathrm{K}_{i*}^{s} = \tilde{\mathrm{K}}$. Remind that $\tilde{\mathrm{K}} = f_{K_{i*}^{s}}(v_2)$ where K_{i*}^{s} is the intermediate value computed by $\prod_{i*}^{s}$.

Game G_0. This is the real game $\mathrm{Game}^{\mathrm{con}}_{\mathcal{A},\mathrm{TDH1}}(\kappa)$, in which the honest players are simulated by Δ.

Game G_1. In this game Δ aborts if the same nonce r_i is used by any honest oracle $\prod_{i}^{s}$ in two different sessions. As in previous proofs we get: $\Pr[\mathrm{Win}_0^{\mathrm{con}}] - \Pr[\mathrm{Win}_1^{\mathrm{con}}] \le Nq_s^2 / 2^{\kappa}$.

Game G_2. This game is identical to Game G_1 with the condition event that $\mathcal{A}$ being in the prepare stage is NOT able to output ρ_{i*} computed by $\prod_{i*}^{s}$ in any session of the attack stage.[6] We show how to evaluate the probability that $\mathcal{A}$ outputs ρ_{i*} in the prepare stage. Recall, ρ_{i*} is computed as $f_{\rho_{i*-1} \oplus \pi(r_{i*})}(v_0)$ in the attack stage. If $\mathcal{A}$ does not know the PRF key in the prepare stage, he can either use a different PRF key (thus finding a PRF-collision) or guess ρ_{i*} at random. If $\mathcal{A}$ knows the PRF key in the first stage, he has to force $\prod_{i*}^{s}$ to compute that key in the attack stage. However, since r_i's are uniform and chosen in the second stage, $\mathcal{A}$ must inuence ρ_{i*-1} this would allow to distinguish *f* from a random function. Since there are at most q_s sessions we have (according to Lemma 1):

$$\Pr[\mathrm{Win}_1^{\mathrm{con}}] - \Pr[\mathrm{Win}_2^{\mathrm{con}}] \le q_s \mathrm{Suc}_F^{\mathrm{coll}}(\kappa) + q_s \mathrm{Adv}_F^{\mathrm{PRF}}(\kappa) + q_s / 2^{\kappa}.$$

Game G_3. In this game, we consider a condition event that $\mathcal{A}$ (being in the prepare stage) is NOT able to output $K_{i*}^{s} := \rho_n$ computed by $\prod_{i*}^{s}$ in any session of the attack stage. Evaluating probabilities that ρ_n, $\rho_{n-1}, \ldots$, can be predicted is done via a hybrid argument. In a nutshell, either the adversary can find the same output with a different key (which breaks collision-resistance) or he inuences the PRF key $\rho_{i-1} \oplus \pi(r_i)$: this can be done either by inverting π or by a random guess. According to Lemma 1 we finally obtain:

$$\Pr[\mathrm{Win}_2^{\mathrm{con}}] - \Pr[\mathrm{Win}_3^{\mathrm{con}}] \le Nq_s \mathrm{Suc}_F^{\mathrm{coll}}(\kappa) + Nq_s \mathrm{Suc}_{\pi}^{\mathrm{ow}}(\kappa) + (Nq_s) / 2^{\kappa}.$$

Game G_4. The condition event here is that $\mathcal{A}$ (being in the prepare stage) is NOT able to output K_{i*}^{s} computed by $\prod_{i*}^{s}$ in any session of the attack stage. Having excluded the case where K_{i*}^{s} is known to $\mathcal{A}$, the probability of such event is (as above) bounded by: $\Pr[\mathrm{Win}_3^{\mathrm{con}}] - \Pr[\mathrm{Win}_4^{\mathrm{con}}] \le q_s \mathrm{Suc}_F^{\mathrm{coll}}(\kappa) + q_s / 2^{\kappa}$. Having $\Pr[\mathrm{Win}_4^{\mathrm{con}}] = 0$ (by definition of the game) one can conclude. □

3.4.1 Comparison of security and performance of TDH1 and other static group key exchange protocols

In Table 1, we compare TDH1 protocol with several well-known provably secure GKE protocols in terms of their performance and achieved security goals. Our comparison is done based on the security arguments and adversarial settings given in the original publications (sometimes transformed to the terminology of our model). In general, 'weak' (or 'strong') denotes consideration of weak (or strong) corruptions for each of the security requirements, whereas 'honest' (or 'malicious') denotes the assumption on the type of the protocol participants. Note again that by strong corruptions we mean not only adaptive attacks revealing the long-lived key (thus, weak corruptions), but also opening attacks which read out the ephemeral secrets. We also distinguish whether a protocol has been proven under standard or non-standard assumptions such as Ideal Cipher Model (ICM) or ROM. We remark that TDH1 is the only protocol which provably satisfies strong versions of AKE-, MA-security and contributiveness (under consideration of malicious insiders where appropriate, that is for MA and contributiveness) while being proven in the standard model. The protocol proposed by Desmedt et al. (2006) has similar properties as TDH1, but deals only with weak corruptions (ephemeral secrets never leak). The work by Katz and Shin (2005)[7] can also be seen as close to ours since they provide MA-security against malicious insiders; the main differences are that their model (although considering strong corruptions) does not allow separate opening attacks, i.e. the scenario in which the adversary learns the ephemeral secrets of other honest users is not considered, and it also does not allow the adversary to register long-lived keys of the users under its control.

Last, but not least, we note that the overall effciency of TDH1 is similar to the most efficient currently known provably secure GKE protocols (in the standard model).

Table 1 Efficiency and security goals of TDH1 and other *static* provably secure group key exchange protocols

	Efficiency			*Security goals*		
GKE protocol	*Comm*	*Comp*	*AKE*	*MA*	*Contributiveness*	*Model*
Abdalla et al. (2006)	O(1)	O(n)	weak	–	–	ICM, ROM
Bresson and Catalano (2004)	O(1)	O(n)	weak	weak, honest	weak, honest	standard
Bresson et al. (2001)	O(n)	O(n)	weak	weak, honest	–	ROM
Bresson, Chevassut and Pointcheval (2002b)	O(n)	O(n)	weak	weak, honest	–	ROM
Desmedt et al. (2006)	O(1)	O(n)	weak	weak, malicious	weak, malicious	standard
Dutta, Barua and Sarkar (2004)	O(1)	O(n)	weak	–	–	standard
Katz and Shin (2005)	O(1)	O(n)	strong	strong[8], malicious	–	standard
Katz and Yung (2003)	O(1)	O(n)	weak	weak, honest	–	standard
TDH1	O(1)	O(n)	strong	strong, malicious	strong, malicious	standard

4 Conclusions and future work

In this article, we have addressed security of GKE protocols against strong (adaptive) corruptions which reveal internal states (incl. ephemeral secrets) of participants and proposed appropriate definitions of strong AKE-, MAsecurity, and contributiveness. Additionally, we presented a 3-round GKE protocol TDH1 which satisfies strong security under standard cryptographic assumptions.

The function TDH1_Con $(x_{\langle 0,0\rangle}, r_1 \mid \ldots \mid r_n)$ is of independent interest and can be seen as an add-on compiler for our definition of contributiveness if $x_{\langle 0,0\rangle}$ is the common ephemeral secret computed in the underlying GKE protocol (see (Bresson and Manulis, 2007) for details).

The equivalence between the TDDH and DDH assumptions is also of independent interest since it is valuable for the construction of other cryptographic schemes with provable security in the standard model. An interesting open question: Is TDDH randomly self-reducible?

Beside the extension of TDH1 towards dynamic groups, general future work in the area of GKE security might address: consideration of strong corruptions in combination with fault-tolerance and security against DoS attacks discussed in Cachin and Strobl (2004) and Desmedt et al. (2006) and strengthening of the simulation-based security models for GKE protocols (e.g. (Katz and Shin, 2005)) towards opening attacks due to our Remark 1.

Acknowledgement

The authors wish to thank Berkant Ustaoglu for his comments on key registration attacks.

References

Abdalla, M., Bresson, E., Chevassut, O. and Pointcheval, D. (2006) 'Password-based group key exchange in a constant number of rounds', Paper presented in the Proceedings of the *PKC'06 of LNCS*, Vol. 3958, pp.427–442, Springer, April.

Ateniese, G., Steiner, M. and Tsudik, G. (1998) 'Authenticated group key agreement and friends', Paper presented in the Proceedings of the *ACM CCS'98*, pp.17–26. ACM Press.

Bellare, M. and Rogaway, P. (1993) 'Random oracles are practical: a paradigm for designing efficient protocols', Paper presented in the Proceedings of the *ACM CCS'93*, pp.62–73. ACM Press.

Bohli, J-M., Vasco, M.I.G. and Steinwandt, R. (2007) 'Secure group key establishment revisited', *Int. J. Information Security*, Vol. 6, pp.243–254.

Boneh, D. (1998) 'The decision Diffie–Hellman problem', Paper presented in the Proceedings of the *ANTS-III*, pp.48–63. Springer.

Boyd, C. and Mathuria, A. (2003) *Protocols for Authentication and Key Establishment*. New York, NY: Springer.

Bresson, E. and Catalano, D. (2004) 'Constant round authenticated group key agreement via distributed computation', Paper presented in the Proceedings of the *PKC'04 of LNCS*, Vol. 2947, pp.115–129. Springer.

Bresson, E. and Manulis, M. (2007) 'Malicious participants in group key exchange: key control and contributiveness in the shadow of trust', Paper presented in the Proceedings of the *ATC '07 of LNCS*, Vol. 4610, pp.395–409. Springer.

Bresson, E. and Manulis, M. (2008) 'Securing group key exchange against strong corruptions', Paper presented in the Proceedings of the *ASI-ACCS '08*, pp.249–260. ACM.

Bresson, E., Chevassut, O. and Pointcheval, D. (2001) 'Provably authenticated group Diffie–Hellman key exchange – the dynamic case', *ASIACRYPT'01 of LNCS*, Vol. 2248, pp.290–390. Springer.

Bresson, E., Chevassut, O. and Pointcheval, D. (2002a) 'Dynamic group Diffie–Hellman key exchange under standard assumptions', *EUROCRYPT'02 of LNCS*, Vol. 2332, pp.321–336. Springer.

Bresson, E., Chevassut, O. and Pointcheval, D. (2002b) 'Group Diffie–Hellman key exchange secure against dictionary attacks', *ASIACRYPT'02 of LNCS*, Vol. 2501, pp.497–514. Springer, December.

Bresson, E., Manulis, M. and Schwenk, J. (2007) 'On security models and compilers for group key exchange protocols', Paper presented in the Proceedings of the *IWSEC '07 of LNCS*, Vol. 4752, pp.292–307. Springer.

Bresson, E., Chevassut, O., Pointcheval, D. and Quisquater, J-J. (2001) 'Provably authenticated group Diffie–Hellman key exchange', Paper presented in the Proceedings of the *ACM CCS'01*, pp.255–264. ACM Press.

Cachin, C. and Strobl, R. (2004) 'Asynchronous group key exchange with failures', Paper presented in the Proceedings of the *PODC '04*, pp.357–366. ACM Press.

Canetti, R. and Krawczyk, H. (2001) 'Analysis of key-exchange protocols and their use for building secure channels', *EUROCRYPT'01*, Vol. 2045 of *LNCS*, pp.453–474. Springer.

Choo, K-K.R., Boyd, C. and Hitchcock, Y. (2005) 'Examining indistinguishability-based proof models for key establishment protocols', *ASIACRYPT'05*, Vol. 3788 of *LNCS*, pp.585–604. Springer.

Crescenzo, G.D., Ferguson, N., Impagliazzo, R. and Jakobsson, M. (1999) 'How to forget a secret', Paper presented in the Proceedings of the *STACS'99*, Vol. 1563 of *LNCS*, pp.500–509. Springer.

Desmedt, Y.G., Pieprzyk, J., Steinfeld, R. and Wang, H. (2006) 'A non-malleable group key exchange protocol robust against active insiders', Paper presented in the Proceedings of the *ISC'06 of LNCS*, Vol. 4176, pp.459–475. Springer.

Dutta, R., Barua, R. and Sarkar, P. (2004) 'Provably secure authenticated tree-based group key agreement', Paper presented in the Proceedings of the *ICICS'04*, Vol. 3269 of *LNCS*, pp.92–104. Springer.

Katz, J. and Shin, J.S. (2005) 'Modeling insider attacks on group key exchange protocols', Paper presented in the Proceedings of the *ACM CCS'05*, pp.180–189. ACM Press.

Katz, J. and Yung, M. (2003) 'Scalable protocols for authenticated group key exchange', *CRYPTO'03 of LNCS*, Vol. 2729, pp.110–125. Springer.

Kim, H-J., Lee, S-M. and Lee, D.H. (2004) 'Constant-round authenticated group key exchange for dynamic groups', *ASIACRYPT'04 of LNCS*, Vol. 3329, pp.245–259.

Kim, Y., Perrig, A. and Tsudik, G. (2004a) 'Group key agreement efficient in communication', *IEEE Transactions on Computers*, Vol. 53, pp.905–921.

Kim, Y., Perrig, A. and Tsudik, G. (2004b) 'Tree-based group key agreement', *ACM Transactions on Information and System Security*, Vol. 7, pp.60–96.

Krawczyk, H. (2005) 'HMQV: 'a high-performance secure Diffie–Hellman protocol', *CRYPTO'05 of LNCS*, Vol. 3621, pp.546–566. Springer.

LaMacchia, B., Lauter, K. and Mityagin, A. (2007) 'Stronger security of authenticated key exchange', Paper presented in the Proceedings of the *ProvSec'07 of LNCS*, Vol. 4784, pp.1–16. Springer.

Manulis, M. (2006) *Survey on Security Requirements and Models for Group Key Exchange. Technical Report 2006/02*. Horst-Görtz Institute, November.

Menezes, A. and Ustaoglu, B. (2008) 'Security arguments for the UM key agreement protocol in the NIST SP 800-56A standard', Paper presented in the Proceedings of the *ASIACCS '08*, pp.261–270. ACM.

Menezes, A., van Oorschot, P. and Vanstone, S. (1996) *Hand-Book of Applied Cryptography*. New York, NY: CRC Press.

Mitchell, C.J., Ward, M. and Wilson, P. (1998) 'Key control in key agreement protocols', *Electronic Letters*, Vol. 34, pp.980–981.

Pedersen, T.P. (1991) 'Non-interactive and information-theoretic secure verifiable secret sharing', *CRYPTO '91 of LNCS*, Vol. 576, pp.129–140. Springer.

Shoup, V. (1999) *On Formal Models for Secure Key Exchange (Version 4). Technical Report RZ 3120*, IBM Research, November.

Shoup, V. (2006) 'Sequences of games: a tool for taming complexity in security proofs', *Cryptology ePrint Archive, Report 2004/332*, January.

Steiner, M. (2002) *Secure Group Key Agreement*. PhD Thesis, Saarland University, March.

Notes

[1] A GKE protocol may use auxiliary secrets pre-computed offline in order to achieve better performance during the communication phase. Such protocols are not strong AKE-secure since the adversary can break into the internal states prior to the protocol execution.

[2] Note that $\mathcal{G}_{\text{std}} \subseteq \mathcal{G}_{\text{us}}$.

[3] The exponentiation $x \mapsto g^x$ is a bijection from $\mathbb{Z}_{|\mathbb{G}|}$ to $\mathbb{G}$. We require that there exists an efficiently computable (bijective) mapping in the opposite direction, but we do NOT require this mapping to be the discrete logarithm!

[4] Binary tree is called *full* if each of its nodes has exactly 0 or 2 children. Sometimes such trees are also called *proper*.

[5] More precisely, one constructs $n + 1$ auxiliary 'hybrid games' $G_{6,1}$, $l = 0, \ldots, n$ and replaces in each game ρ_l by a random value from $\{0, 1\}^{\kappa}$. The difference between two neighbour hybrids is upper-bounded by the PRF advantage.

[6] Note, in G_0 and G_1 the adversary only outputs a value for the resulting group key. In G_2, we consider the additional (in)ability of the adversary to output the value for ρ_i^*. Since we are only interested in the success probability of $\mathcal{A}$ under this condition Δ does not need to detect whether $\mathcal{A}$ is able to output the correct value or not. The same considerations are applicable to G_3 with respect to $K_{i^*}^s$.

[7] Note that Katz and Shin proposed an add-on compiler and not a concrete protocol.

[8] MA-security related definitions in Katz and Shin (2005) do not consider opening attacks, i.e. the adversary is not allowed to obtain internal states of other uncorrupted participants.

Appendix A

Proof of theorem 1 (DDH $\Leftrightarrow$ TDDH)

$\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa) \le \mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\kappa)$: this holds trivially since distributions $\mathrm{TDDH}^{*}_{T_n}$ and $\mathrm{TDDH}^{\$}_{T_n}$ contain a triple of the form $\left(\left(\langle 1,0\rangle, g^{x_{\langle 1,0\rangle}}\right), \left(\langle 1,1\rangle, g^{x_{\langle 1,1\rangle}}\right), (\langle 0,0\rangle, Z)\right)$ where $Z = g^{x_{\langle 1,0\rangle}\, x_{\langle 1,1\rangle}}$ in case of $\mathrm{TDDH}^{*}_{T_n}$ or Z is random otherwise.

$\mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\kappa) \le (2n-3)\,\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa)$: to prove this, we use a TDDH_{T_n}-distinguisher $\mathcal{A}$, and show how to solve an instance of the DDH problem: on input $(A, B, C) \in \mathbb{G}^3$, where $A = g^a$ and $B = g_b$ for random a and b, we build a PPT algorithm Δ that distinguishes whether $C = g^{ab}$ or C is random.

First, we sort the nodes of any $T_n \in_R T_n$ in the postfix order, s.t. each node is listed *after* its two children (if any). For simplicity, we slightly modify this order: we separate nodes by re-numbering all n leaves to negative indices (without changing their order), and the internal nodes to indices 1 to n–1 (also without changing their order). In other words, for internal nodes, we 'shrink' the sequence that remains after having moved the leaves. This results in the following map σ from T_n to $[-n,-1] \cup [1,n-1]$:

$$\underbrace{-n, -n+1, \ldots, -1}_{\text{number assigned to leaves}}\ ,\ \underbrace{1, \ldots, n-1}_{\text{internal nodes}}$$

Note that any node still appears after its children, e.g. the root node is assigned number $\sigma(\langle 0,0\rangle) = n-1$. By a 'hybrid argument' we consider the following sequence of games. In each game G_i for $i = 0, \ldots, n-2$, Δ chooses a set X of n random values in $\mathbb{G}$, denoted x_{-n} through x_{-1}. In addition, for $i > 0$, in G_i, Δ chooses a set Y_i of i random values in $\mathbb{G}$ that are denoted x_1 through x_i. Then, Δ builds a set $\mathrm{TDH}_i(X, Y_i)$ defined (recursively) as $\left\{\left(\langle l,v\rangle, g^{x_{\langle l,v\rangle}}\right)\right\}_{\langle l,v\rangle \in T^{*}_n}$, with

$$\begin{cases} x_{\langle l,v\rangle} = x_{\sigma(\langle l,v\rangle)} \in X & \text{if } \sigma(\langle l,v\rangle) < 0, \\ x_{\langle l,v\rangle} = x_{\sigma(\langle l,v\rangle)} \in Y_i & \text{if } 0 < \sigma(\langle l,v\rangle) \le i, \\ x_{\langle l,v\rangle} = g^{x_{\langle l,2v\rangle} x_{\langle l+1,2v+1\rangle}} & \text{otherwise.} \end{cases}$$

Finally, in each game, Δ flips a coin b and provides $\mathcal{A}$ with a set $\mathrm{TDDH}_i(X, Y_i, b, r) = \mathrm{TDH}_i(X, Y_i) \cup \{(\langle 0,0\rangle, Z)\}$ where $Z = g^{x_{\langle 1,0\rangle}\, x_{\langle 1,1\rangle}}$ if $b = 1$ and $Z = g^r$ is a random element in $\mathbb{G}$ if $b = 0$.

Let $\Pr_i[\ldots]$ denote the probabilities as induced by random choices in G_i. In G_0 the constructed $\mathrm{TDH}_0(X, Y_i)$ is exactly $\mathrm{TDH}_{T_n}(X)$ (due to $Y_0 = \varnothing$) s.t. the distance between $\Pr_0[\mathcal{A}(\mathrm{TDDH}_0(X, \varnothing, b, r)) = 1 | b = 1]$ and $\Pr_0[\mathcal{A}(\mathrm{TDDH}_0(X, \varnothing, b, r)) = 1 | b = 0]$ is upper-bounded by $\mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\kappa)$. On the other hand, in G_{n-2}, all values $x_{\langle l,v\rangle}$ for $l > 1$ are random and independent, and also independent from random $x_{\langle 1,0\rangle}$ and $x_{\langle 1,1\rangle}$. Furthermore, $x_{\langle 0,0\rangle} = g^{x_{\langle 1,0\rangle} x_{\langle 1,1\rangle}}$ iff $b = 1$ s.t. the distance between $\Pr[\mathcal{A}(\mathrm{TDDH}_{n-2}(X, Y_{n-2}, b, r)) = 1 | b = 0]$ and $\Pr[\mathcal{A}(\mathrm{TDDH}_{n-2}(X, Y_{n-2}, b, r)) = 1 | b = 0]$ is upper-bounded by $\mathrm{Adv}^{\mathrm{D}\mathbb{G}\mathrm{DH}}_{\mathbb{G}}(\kappa)$. The last experiment G_* is identical to G_{n-2} except that A, B and C are used in the computation of $\mathrm{TDDH}_{n-2}(X, Y_{n-2}, b, r)$ instead of $g^{x_{\langle 1,1\rangle}}$ and $x_{\langle 0,0\rangle}$, respectively. In particular, the flipping of b is ignored here: whatever b, the last input of $\mathcal{A}$ is set to C. Let β denote the hidden bit that Δ is trying to guess. Note, the random variables $(b,\ g^{x_{\langle 1,0\rangle}},\ g^{x_{\langle 1,1\rangle}},\ x_{\langle 0,0\rangle})$ and (β, A, B, C) are identically distributed. It follows that (for simplicity we removed $\mathcal{A}$'s inputs that are identical and independent from the rest in both cases):

$$\begin{aligned} &\Pr_{n-2}\left[\mathcal{A}\left(g^{x_{\langle 1,0\rangle}}, g^{x_{\langle 1,1\rangle}}, x_{\langle 0,0\rangle}\right) = 1 \mid b = 1\right] \\ &\quad = \Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 1\right] \\ &\Pr_{n-2}\left[\mathcal{A}\left(g^{x_{\langle 1,0\rangle}}, g^{x_{\langle 1,1\rangle}}, x_{\langle 0,0\rangle}\right) = 1 \mid b = 0\right] \\ &\quad = \Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 0\right]. \end{aligned}$$

Finally, it is straightforward to see that the computational distance between two consecutive games is upper-bounded by $\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa)$ since the only difference between them is to replace a value $g^{x_{\langle l+1,2v\rangle} x_{\langle l+1,2v+1\rangle}}$ by a random one. Hence,

$$\begin{aligned} &\left|\Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 1\right] - \Pr_0\left[\mathcal{A}(\mathrm{TDDH}_0(X,\varnothing,b,r))\right] = 1 \mid b = 1\right| \\ &\quad \le (n-2)\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa) \\ &\left|\Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 0\right] - \Pr_0\left[\mathcal{A}(\mathrm{TDDH}_0(X,\varnothing,b,r))\right] = 1 \mid b = 0\right| \\ &\quad \le (n-2)\mathrm{Adv}^{\mathrm{DDH}}_{\mathbb{G}}(\kappa). \end{aligned}$$

Their sum gives us the desired inequality:

$$\begin{aligned} \mathrm{Adv}^{\mathrm{TDDH}}_{T_n,\mathbb{G}}(\kappa) &= \left|\Pr_0\left[\mathcal{A}(\mathrm{TDDH}_0(X,\varnothing,b,r)) = 1 \mid b = 1\right]\right. \\ &\quad \left. - \Pr_0\left[\mathcal{A}(\mathrm{TDDH}_0(X,\varnothing,b,r)) = 1 \mid b = 0\right]\right| \\ &\le \left|\Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 1\right]\right| \\ &\quad - \Pr_{*}\left[\mathcal{A}(A,B,C) = 1 \mid \beta = 0\right]\left| \right. \\ &\quad + 2(n-2)\mathrm{Adv}^{\mathrm{DDh}}_{\mathbb{G}}(\kappa) \\ &\le (2n-3)\mathrm{Adv}^{\mathrm{DDh}}_{\mathbb{G}}(\kappa). \end{aligned}$$

□

Appendix B

Security definitions for F

Definition 7 (PRF Family). A family of functions $F := \left\{\left\{f_k : \{0,1\}^{p(\kappa)} \to \{0,1\}^{p(\kappa)}\right\}_{k \in \{0,1\}^{\kappa}}\right\}_{\kappa \in \mathbb{N}}$ with p a

polynomial, is called an (efficiently computable) PRF ensemble if:

1 *Efficient computation*. There exists a polynomial-time algorithm that on input k and x returns $f_k(x)$.

2 *Pseudo-Randomness*. Choose uniformly $k \in_R \{0,1\}^\kappa$ and a function $\tilde{f}$ in the set of all functions with domain and range $\{0,1\}^{p(\kappa)}$. Consider a PPT adversary $\mathcal{A}$ asking queries of the form Tag(x) and participating in one of the following two games:

- $\text{Game}_{\mathcal{A},F}^{\text{prf-1}}(\kappa)$ where a query Tag(x) is answered with $f_k(x)$
- $\text{Game}_{\mathcal{A},F}^{\text{prf-0}}(\kappa)$ where a query Tag(x) is answered with $\tilde{f}(x)$.

At the end of the execution $\mathcal{A}$ outputs a bit b trying to guess which game was played. The output of $\mathcal{A}$ is also the output of the game.

We define:

$$\text{Adv}_{\mathcal{A},F}^{\text{prf}}(\kappa) := \left| 2 \Pr\left[\text{Game}_{\mathcal{A},F}^{\text{prf-}b}(\kappa) = b \right] - 1 \right|.$$

and denote $\text{Adv}_{\mathcal{A},F}^{\text{prf}}(\kappa)$ the maximum advantage over all adversaries $\mathcal{A}$. We say that F is pseudo-random if this advantage is a negligible function in κ.

By an (efficiently computable) PRF, we mean a function $f_k \in F$ for some random $k \in_R \{0,1\}^\kappa$.

Definition 8 (Collision-Resistance). Let $F := \left\{ \{f_k\}_{k \in \{0,1\}^\kappa} \right\}_{\kappa \in \mathbb{N}}$ be a PRF ensemble. We say that F is collision-resistant if there is an efficient procedure Sample such that for all PPT adversaries $\mathcal{A}$ the following success probability (over all adversaries) is a negligible function in κ:

$$\text{Suc}_F^{\text{coll}}(\kappa) := \Pr\left[\begin{array}{lcl} x \leftarrow \text{Sample}(1^\kappa); & & k, k' \in \{0,1\}^\kappa \wedge \\ & : & k \neq k' \wedge \\ k, k' \leftarrow \mathcal{A}(1^\kappa, x) & & f_k(x) = f_{k'}(x) \end{array} \right].$$