

A Flexible Framework for Secret Handshakes

(Multi-Party Anonymous and Un-observable Authentication)

Gene Tsudik* and Shouhuai Xu**

Abstract. In the society increasingly concerned with the erosion of privacy, privacy-preserving techniques are becoming very important. This motivates research in cryptographic techniques offering built-in privacy.

A secret handshake is a protocol whereby participants establish a secure, anonymous and unobservable communication channel only if they are members of the same group. This type of “private” authentication is a valuable tool in the arsenal of privacy-preserving cryptographic techniques. Prior research focused on 2-party secret handshakes with one-time credentials.

This paper breaks new ground on two accounts: (1) it shows how to obtain secure and efficient secret handshakes with reusable credentials, and (2) it represents the first treatment of group (or *multi-party*) secret handshakes, thus providing a natural extension to the secret handshake technology. An interesting new issue encountered in multi-party secret handshakes is the need to ensure that all parties are indeed distinct. (This is a real challenge since the parties cannot expose their identities.) We tackle this and other challenging issues in constructing **GCD** – a flexible framework for secret handshakes. The proposed **GCD** framework lends itself to many practical instantiations and offers several novel and appealing features such as self-distinction and strong anonymity with *reusable* credentials. In addition to describing the motivation and step-by-step construction of the framework, this paper provides a thorough security analysis and illustrates two concrete framework instantiations.

Keywords: secret handshakes, privacy-preservation, anonymity, credential systems, unobservability, key management.

1 Introduction

Much of today’s communication is conducted over public networks which naturally prompts a number of concerns about security and privacy. Communication security has been studied extensively and a number of effective and efficient security tools and techniques are available.

Unfortunately, privacy concerns have not been addressed to the same extent. Yet, it is quite obvious to anyone who keeps up with the news that our society is very concerned with privacy. At the same time, privacy is being eroded by (often legitimate) concerns about crime, terrorism and other malfeasances. Furthermore, the proliferation of wireless communication (among laptops, cell phones, PDAs, sensors and RFIDs) drastically lowers the bar for eavesdropping and tracking of both people and their devices.

Popular techniques to provide communication privacy include email MIX-es, anonymizing routers and proxy web servers as well as purely cryptographic tools, such as private information retrieval. Despite important advances, the privacy continuum has not been fully explored. One particular issue that has not been widely recognized is the need for unobservable, untraceable and anonymous authentication, i.e., **privacy-preserving authentication**. Such a notion might seem counter-intuitive at first, since authentication traditionally goes hand-in-hand with identification. However, in the context of groups or roles, authentication identifies not a distinct entity but a collection thereof. To this end,

* Contact Author. Department of Computer Science, University of California, Irvine. gts@ics.uci.edu.

** Department of Computer Science, University of Texas, San Antonio. shxu@cs.utsa.edu.

some advanced cryptographic techniques have been developed, such as group signatures [1] and privacy-preserving trust negotiation [9, 25].

We focus on *interactive privacy-preserving mutual authentication*; more specifically, on *secret handshakes*. A secret handshake scheme (SHS) allows two or more group members to authenticate each other in an anonymous, unlinkable and unobservable manner such that one’s membership is not revealed unless every other party’s membership is also ensured.¹

In more detail, a secure handshake allows members of the same group to identify each other *secretly*, such that each party reveals its affiliation to others if and only if the latter are also group members. For example, in a 2-party setting, an FBI agent (Alice) wants to authenticate to Bob only if Bob is also an FBI agent. Moreover, if Bob is *not* an FBI agent, he should be unable to determine whether Alice is one (and vice versa). This property can be further extended to ensure that group members’ affiliations are revealed only to members who hold specific *roles* in the group. For example, Alice might want to authenticate herself as an agent with a certain clearance level *only if* Bob is also an agent with at least the same clearance level.

In a more general sense, secret handshakes offer a means for privacy-preserving mutual authentication with many possible applications, especially, in hostile environments.

Goals: We set out to develop techniques for supporting efficient **multi-party** secret handshakes while avoiding certain drawbacks present in some or all of the previous **2-party** secret handshake solutions. These drawbacks include: (1) use of one-time credentials or pseudonyms, (2) ability of the group authority to cheat users, (3) requirement to maintain information about many irrelevant groups (groups that one is not a member of), and (4) lack of support for handshakes of three or more parties. Some of these drawbacks are self-explanatory, while others are clarified later in the paper.

1.1 Overview and Summary of Contributions

We are interested in multi-party secret handshakes, whereby $m \geq 2$ parties establish a secure, anonymous and unobservable communication channel provided that they are members of the same group. We achieve this by constructing a secret handshake framework called **GCD**. This framework is essentially a *compiler* that transforms three main ingredients – a **G**roup signature scheme, a **C**entralized group key distribution scheme, and a **D**istributed group key agreement scheme – into a secure secret handshake scheme. We formally specify this framework based on desired functionality and security properties.

From the functionality perspective, existing solutions are only able to support 2-party secret handshakes [3], [14, 35]. Our framework represents the first result that supports truly *multi-party* secret handshakes. Moreover, our work is first to solve the problem of *partially-successful* secret handshakes.²

¹ This informal definition broadens the prior version [3] which limited secret handshakes to two parties.

² A partially successful handshake occurs whenever not all parties engaged in a handshake protocol are members of the same group. For example, if 5 parties take part in a secret handshake and 2 of them are members of group A, while the rest are members of group B, the desired outcome is for both the former and the latter to complete the secret handshake protocol

From the security perspective, our framework has two novel features. First, it can be resolved into concrete schemes that provide the novel and important **self-distinction** property which ensures the uniqueness of each handshake participant. In other words, it guarantees that the protocol is a multi-party computation with the exact number of players that claim to be participating. Without **self-distinction**, a malicious insider can easily impersonate any number of group members by simultaneously playing multiple roles in a handshake protocol.³ Thus, an honest participant may be fooled into making a wrong decision when the number of participating parties is a factor in the decision-making policy. We also note that self-distinction is trivial for 2-party secret handshakes. However, it becomes more challenging for handshakes of three or more, since the parties cannot simply expose their identities; otherwise, anonymity would be lost.

Second, in contrast with prior work [3, 14] which relies on one-time credentials to achieve **unlinkability** – this ensures that multiple handshake sessions involving the same participant(s) cannot be linked by an adversary – our approach provides **unlinkability** with multi-show (or reusable) credentials. This greatly enhances its usability. Moreover, our approach does not require users to be aware of other groups, in contrast with [35].

In addition, our framework has some interesting *flexibility* features. In particular, it is model-agnostic: if the building blocks operate in the asynchronous communication model (with guaranteed delivery), so does the resulting secret handshake scheme. Also, it supports a set of selectable properties that can be tailored to application needs and semantics (e.g., the two specific instantiations have two different sets of properties). Finally, it lends itself to many practical instantiations: we present two concrete examples where a handshake participant computes only $O(m)$ modular exponentiations and sends/receives $O(m)$ messages, where m is the number of handshake participants.

Organization: Section 2 presents our system model and definitions of secret handshake schemes. Then we proceed to discuss the design space and lay the foundation for the framework in Section 3. The models and definitions for the three building blocks are discussed in Sections 4, 5, and 6. Next, Section 7 presents the actual **GCD** framework and the analysis of its properties, followed by two concrete instantiations in Section 8. Some practical issues are considered in Section 9 and related work is overviewed in Section 10. The paper concludes with the summary and future research directions.

Appendix: For the sake of clarity and because of space limitations, we placed some technical material into the Appendix. Appendices A, B, C and D describe the formal security properties of, respectively: (1) secret handshake schemes, (2) group signature schemes, (3) group key distribution schemes, and (4) distributed group key agreement schemes. Appendices E, F and G contain security proofs for the framework and one of its instantiations. Appendix H overviews a specific group signature scheme used in the second instantiation.

and determine that their respective handshakes were performed with 2 and 3 members, respectively. Our scheme achieves this desired goal.

³ This is reminiscent of the well-known Sybil attack [19], which is nevertheless different and not addressed in the present paper.

2 Secret Handshakes: Model and Definition

In this section, we specify the system model and discuss the security of secret handshakes.

Let κ be a security parameter and \mathcal{U} be a set of all users: $\mathcal{U} = \{U_i \mid 0 < i < n\}$ where n is bounded by $\text{poly}(\kappa)$. Let \mathbf{G} be a set of groups, where each group⁴ $G \in \mathbf{G}$ is a set of members managed by a *group authority* \mathcal{GA} , which is responsible for admitting members, revoking their membership and updating system state information. For simplicity’s sake we assume that each user is a member of exactly one group. (Of course, all results can be easily generalized to the case that users are allowed to join multiple groups.) An adversary \mathcal{A} is allowed to corrupt various participants. All participants (including \mathcal{A}) are modeled as probabilistic polynomial-time algorithms.

We assume the existence of *anonymous* channels between all the legitimates participants, where the term “anonymous” means that an outside attacker cannot determine identities of the \mathcal{GA} , group members, as well as the dynamics and size of a group, and that a malicious insider cannot determine the identities of other honest group members as well as the the dynamics and size of the group. This assumption is necessary in most privacy-preserving authentication schemes; otherwise, anonymity could be trivially compromised. However, we note that the fact that secret handshake protocols themselves rely on anonymous channels does *not* necessarily present a problem. This is because a typical secret handshake application would be in a wireless setting where all communication is done via *broadcast* which offers receiver anonymity as a “built-in” feature.⁵ (See Section 9 for further discussion of practical issues.)

Fig. 1. Definition of Multi-Party Secret Handshake Schemes

SHS.CreateGroup: executed by \mathcal{GA} to establish a group G . It takes as input appropriate security parameters, and outputs a cryptographic context specific to this group. The context may include a certificate/membership revocation list, \mathcal{CRL} , which is originally empty. The cryptographic context is made public, while the \mathcal{CRL} is made known only to current group members.

SHS.AdmitMember: executed by \mathcal{GA} to admit a user to the group under its jurisdiction. We assume that \mathcal{GA} admits members according to a certain admission policy. Specification and enforcement of such policy is out the scope of this paper. After executing the algorithm, group state information has been appropriately updated, the new member holds some secret(s) as well as a membership certificate(s), and existing members obtain updated system information from \mathcal{GA} via the aforementioned authenticated anonymous channel.

SHS.RemoveUser: executed by \mathcal{GA} . It takes as input the current \mathcal{CRL} and a user identity U_i such that $U_i \in \mathcal{U}$ and $U_i \in G$. The output includes an updated \mathcal{CRL} which includes the newly revoked certificate for U_i . The state update information is sent to the existing group members through the authenticated anonymous channel.

SHS.Update: executed by each current group member upon receiving, via the authenticated anonymous channel, system state update information from \mathcal{GA} . It is used to update each member’s system state information.

SHS.Handshake(Δ): executed by a set Δ of m users purporting to be members of a group G , where $\Delta = \{U_1, \dots, U_m\}$ and $m \geq 2$. The input to this protocol includes the secrets of all users in Δ , and possibly some public information regarding the current state of the systems. At the end of a protocol execution, it is ensured that each $U_i \in \Delta$ determines that $\Delta \setminus \{U_i\} \subseteq G$ if and only if each $U_j \in \Delta$ ($j \neq i$) discovers $\Delta \setminus \{U_j\} \subseteq G$.

SHS.TraceUser: executed by \mathcal{GA} . On input of a transcript of a successful secret handshake protocol $\text{SHS.Handshake}(\Delta)$, \mathcal{GA} outputs the identities of all m participants involved in the handshake, i.e., U_1, \dots, U_m .

A secret handshake scheme (SHS) consists of the algorithms and protocols shown in Figure 1. We note that the definition says nothing about the participants establishing a common key following (or during) a successful handshake.

⁴ We use “group” to refer to a set of users, unless explicitly stated otherwise.

⁵ This does not contradict our claim in Section 1 that wirelessness heightens privacy concerns. Although eavesdropping is easier in wireless networks, receiver anonymity is, at the same time, also easier to achieve in wireless (rather than in wired) networks.

It is indeed straightforward to establish such a key if a secret handshake succeeds. However, allowing further communication based on a newly established key would require concealing the outcome of the handshake. (See also Section 9.) The definition also does not ensure any form of “agreement” in the sense of [20], since the adversary is assumed to have complete control over all communication, and can corrupt parties. This also explains why we only achieve a somewhat *weak* form of traceability.

Desired security properties are informally specified in Figure 2. In order to highlight the intuition, the formal treatment is deferred to Appendix A. We notice that for certain applications **full-unlinkability** may be desirable, while for certain other applications **unlinkability** and **self-distinction** may be desirable. In other words, the framework specifies the important properties, while leaving the decision on which subset of the properties to satisfy to the specific applications.

Fig. 2. Informal Security for Multi-Party Secret Handshake Schemes

- * **Correctness:** If all handshake participants $\{U_1, \dots, U_m\}$ belong to the same group, the protocol returns “1”; otherwise, the protocol returns “0”.
- * **Resistance to impersonation:** an adversary $\mathcal{A} \notin G$ who does not corrupt any members of G has only a negligible probability in convincing an honest user $U \in G$ that $\mathcal{A} \in G$. This remains to be true even if \mathcal{A} plays the roles of multiple participants.
- * **Resistance to detection:** no adversary $\mathcal{A} \notin G$ can distinguish between an interaction with an honest user $U \in G$ and an interaction with a simulator. This remains to be true even if \mathcal{A} plays the roles of multiple participants.
- * **Full-unlinkability:** no adversary \mathcal{A} is able to associate two handshakes involving a same honest user $U \in G$, even if $\mathcal{A} \in G$ and \mathcal{A} participated in both executions, and U has been corrupt. This remains to be true even if \mathcal{A} plays the roles of multiple participants.
- * **Unlinkability:** no adversary \mathcal{A} is able to associate two handshakes involving a same honest user $U \in G$, even if $\mathcal{A} \in G$ and \mathcal{A} participated in both executions. This remains to be true even if \mathcal{A} plays the roles of multiple participants.
- * **Indistinguishability to eavesdroppers:** no adversary \mathcal{A} who does not participate in a handshake protocol can distinguish between a successful handshake between $\{U_1, \dots, U_m\} \subseteq G$ and an unsuccessful one, even if $\mathcal{A} \in G$.
- * **Traceability:** $\mathcal{G}\mathcal{A}$ can trace all users involved in the handshake session of a given transcript.
- * **No-misattribution:** no coalition of malicious parties (including any number of group members and the $\mathcal{G}\mathcal{A}$) is able to frame an honest member as being involved in a secret handshake.
- * **Self-distinction:** each participant is ensured that all the participants are distinct.

Remark: If needed, our definitions of **resistance to impersonation** and **resistance to detection** can be naturally extended to capture the case when \mathcal{A} corrupts some group members but does not use their secrets in the subsequent handshake protocols.

The flavor of **traceability** achieved in the framework is relatively weak since the protocol participant who is last to send out the values (to facilitate traceability) can always neglect to do so. However, we observe that this holds in other schemes, even in those based on one-time credentials [3, 14]. The subtle issue is that the last sender could always use a “fake” token before other (honest) participants can verify its validity. This is inevitable because of the basic impossibility result in [20]. While there are some purely theoretical ways to mitigate this problem, we are interested in efficient (i.e., practical) solutions. Consequently, we are prepared to tolerate some unfairness, which, nevertheless, only exists between *legitimate* users. As a result, the achieved traceability is still valuable for investigating activities of group members before they become corrupt.

3 Design Space

As mentioned earlier, the **GCD** framework is essentially a compiler that outputs a multi-party secret handshake scheme satisfying all desired properties specified in Section 2. Its input includes:

- A group signature scheme (**GSIG**): a scheme that allows any group member to produce signatures on behalf of the group in an anonymous and unlinkable manner; only a special entity (called a group manager) is able to revoke anonymity and “open” a group signature thereby revealing the signer’s identity. (See Section 4.)
- A centralized group key distribution (broadcast encryption) scheme (**CGKD**): a key management scheme for large one-to-many groups that handles key changes due to dynamic group membership and facilitates secure broadcast encryption. (See Section 5.)
- A distributed group key agreement scheme (**DGKA**): a scheme that allows a group of peer entities to dynamically (on-the-fly) agree on a common secret key to be used for subsequent secure communication within that group. (See Section 6.)

We now discuss the choices made in designing **GCD**. As a first try, one might be tempted to construct a secret handshake scheme directly upon a **CGKD** that enables secure multicast. It is easy to see that $m \geq 2$ members can conduct efficient secret handshakes based on a group key k . However, this approach would have some significant drawbacks:

- (1) No indistinguishability-to-eavesdroppers. A *passive* malicious (or even honest-but-curious) group member can detect, by simply eavesdropping, whenever other members are conducting a secret handshake.
- (2) No traceability. A dishonest member who takes part in a handshake (or is otherwise malicious) can not be traced and held accountable.
- (3) No self-distinction. For handshakes of more than two parties, self-distinction is not attained since a rogue member can play multiple roles in a handshake.

Alternatively, one could employ a **GSIG** scheme as a basis for a secret handshake scheme. This would avoid the above drawback (2), however, drawback (1) remains. Also, resistance to detection attacks would be sacrificed, since (as noted in [3]), group signatures are verifiable by anyone in possession of the group public key.

A natural next step is to combine a **CGKD** with a **GSIG**. This way, the **GSIG** group public key is kept secret among all current group members (along with the **CGKD** group-wide secret key k), and – during the handshake – group signatures would be encrypted under the group-wide key k . Although traceability would be re-gained, unfortunately, drawbacks (1) and (3) would remain.

In order to avoid (1), we need the third component, an interactive distributed key agreement protocol. With it, any member who wants to determine if other parties are members (or are conducting a secret handshake) is forced to participate in a secret handshake protocol. As a result, the group signatures are encrypted with a key derived from

both: (a) the group-wide key and (b) the freshly established key. Moreover, we can thus ensure that, as long as a group signature is presented by a corrupt member, the traceability feature enables the group authority to hold that member accountable.

As pertains to drawback (3) above (no self-distinction), we defer the discussion to later in the paper. Suffice it to say that group signature schemes do not provide self-distinction by design, since doing so undermines their version of the unlinkability property. (Unlinkability in group signatures is different from that in group secret handshakes; see Section 8.2.) To remedy the situation, we need some additional tools, as described in Section 8 below.

Since our approach involves combining a group signature scheme with a centralized group key distribution scheme, it is natural to examine potentially redundant components. In particular, both GSIG and CGKD schemes include a revocation mechanism. Furthermore, revocation in the former is quite expensive, usually based on dynamic accumulators [12]. Thus, it might seem worthwhile to drop the revocation of component of GSIG altogether in favor of the more efficient revocation in CGKD. This way, a revoked member would simply not receive the new group-wide key in CGKD but would remain un-revoked as far as the underlying GSIG is concerned. To illustrate the problem with this optimization, consider an attack whereby a malicious but unrevoked member reveals the CGKD group-wide key to a revoked member. The latter can then take part in secret handshakes and successfully fool legitimate members. Whereas, if both revocation components are in place, the attack fails since the revoked member’s group signature (exchanged as part of the handshake) would not be accepted as valid.

4 Building Block I: Group Signature Schemes

Let \mathcal{U} be the universe of user identities. In a group signature scheme, there is an authority called a group manager (\mathcal{GM}) responsible for admitting users and identifying the actual signer of a given group signature⁶. There is also a set of users who can sign on behalf of the group. In addition, there is a set of entities called verifiers. All participants are modeled as probabilistic polynomial-time algorithms.

A group signature scheme, denoted by GSIG, is specified in Figure 4.

Informally, we require a group signature scheme to be **correct**, i.e., any signature produced by an honest group member using **Sign** is always accepted by **Verify**.

Following notable prior work [4, 7, 23], we say a group signature scheme is **secure** if it satisfies the following three properties (see Appendix B for a formal definition): (1) **full-traceability** – any valid group signature can be traced back to its actual signer, (2) **full-anonymity** – no adversary can identify the actual signer of a group signature, even if the actual signer’s secret has been compromised, and (3) **no-misattribution** – no malicious group manager can misattribute a group signature to an honest group member.

In order to achieve secret handshakes of **self-distinction**, we may also adopt group signature schemes achieving a somewhat weaker privacy notion. Specifically, we can substitute the following weaker notion of **anonymity** for the

⁶ Sometimes, the two functionalities are assigned to two separate entities.

Fig. 3. Definition: Group Signature Scheme

Setup: a probabilistic polynomial-time algorithm that, on input of a security parameter κ , outputs the specification of a cryptographic context including the group manager’s public key $pk_{\mathcal{GM}}$ and secret key $sk_{\mathcal{GM}}$. This procedure may be denoted by $(pk_{\mathcal{GM}}, sk_{\mathcal{GM}}) \leftarrow \text{Setup}(1^\kappa)$.

Join: a protocol between \mathcal{GM} and a user (conducted over a private and authenticated channel) that results in the user becoming a group member U . Their common output includes the user’s unique membership public key pk_U , and perhaps some updated information that indicates the current state of the system. The user’s output includes a membership secret key sk_U . This procedure may be denoted by $(pk_U, sk_U, certificate_U; pk_U, certificate_U) \leftarrow \text{Join}[U \leftrightarrow \mathcal{GM}]$, where $\text{Join}[U \leftrightarrow \mathcal{GM}]$ denotes an interactive protocol between U and \mathcal{GM} , $pk_U, sk_U, certificate_U$ is the output of U , and $pk_U, certificate_U$ is the output of \mathcal{GM} . Besides, there may be some system state information that is made public to all participants.

Revoke: an algorithm that, on input of a group member’s identity (and perhaps her public key pk_U), outputs updated information that indicates the current state of the system after revoking the membership of a given group member.

Update: a deterministic algorithm that may be triggered by any Join or Revoke operation. It is run by each group member after obtaining system state information from the group manager.

Sign: a probabilistic algorithm that, on input of: key $pk_{\mathcal{GM}}, (sk_U, pk_U)$ and a message M , outputs a group signature σ of M . This procedure may be denoted by $\sigma \leftarrow \text{Sign}(pk_{\mathcal{GM}}, pk_U, sk_U, M)$.

Verify: an algorithm that, on input of: $pk_{\mathcal{GM}}$, an alleged group signature σ and a message M , outputs a binary value TRUE/FALSE indicating whether σ is a valid group signature (under $pk_{\mathcal{GM}}$) of M . This procedure may be denoted by $\text{TRUE}/\text{FALSE} \leftarrow \text{Verify}(pk_{\mathcal{GM}}, M, \sigma)$.

Open: an algorithm executed by the group manager \mathcal{GM} . It takes as input of a message M , a group signature σ , $pk_{\mathcal{GM}}$ and $sk_{\mathcal{GM}}$. It first executes **Verify** on the first three inputs and, if the output of **Verify** is TRUE, outputs some incontestable evidence (e.g., a membership public key pk_U and a proof) that allows anyone to identify the actual signer. This procedure may be denoted, without loss of generality, by $U \leftarrow \text{Open}(pk_{\mathcal{GM}}, sk_{\mathcal{GM}}, M, \sigma)$ if $\text{TRUE} \leftarrow \text{Verify}(pk_{\mathcal{GM}}, M, \sigma)$.

above **full-anonymity**: (2’) **anonymity** – no adversary can identify the actual signer of a group signature, as long as the actual signer’s secret has *not* been compromised. As we will see, our specific handshake scheme achieving **self-distinction** is based on the variant group signature scheme of [22], which fulfills the above **anonymity** rather than **full-anonymity**.

5 Building Block II: Centralized Group Key Distribution Scheme

Let κ be a security parameter, and \mathbb{ID} be the set of possible group members (i.e., users, receivers, or principals) such that $|\mathbb{ID}|$ is polynomially-bounded in κ . There is a special entity called a *Group Controller* (i.e., key server, center, server, or sender), denoted by \mathcal{GC} , such that $\mathcal{GC} \notin \mathbb{ID}$. A centralized group key distribution scheme (CGKD) is defined in Figure 4, which is adopted from [34].

Since a (stateful) group communication scheme is driven by “rekeying” events (because of joining or leaving operations below), it is convenient to treat the events occur at “virtual time” $t = 0, 1, 2, \dots$, because the group controller is able to maintain such an execution history. At time t , let $\Delta^{(t)}$ denote the set of legitimate group members, $k^{(t)} = k_{\mathcal{GC}}^{(t)} = k_{U_1}^{(t)} = \dots$ the group (or session) key, $K_{\mathcal{GC}}^{(t)}$ the set of keys held by \mathcal{GC} , $K_U^{(t)}$ the set of keys held by $U \in \Delta^{(t)}$, $\text{acc}_U^{(t)}$ the state indicating whether $U \in \Delta^{(t)}$ has successfully received the rekeying message. Initially, $\forall U \in \mathbb{ID}, t \in \mathbb{N}$, set $\text{acc}_U^{(t)} \leftarrow \text{FALSE}$. We assume that \mathcal{GC} treat joining and leaving operation separately (e.g., first fulfilling the leaving operation and then immediately the joining one), even if the requests are made simultaneously. This strategy has indeed been adopted in the group communication literature.

To simplify the presentation, we assume that during system initialization (i.e., **Setup** described below) \mathcal{GC} can communicate with each legitimate member U through an *authenticated private* channel. In practice, this assumption can be implemented with a two-party authenticated key-exchange protocol. Further, we assume that \mathcal{GC} can establish a common secret, if needed, with a joining user, and that after the system initialization \mathcal{GC} can communicate with any $U \in \mathbb{ID}$ through an *authenticated* channel.

Fig. 4. Definition: Centralized Group Key Distribution

Setup: The group controller \mathcal{GC} generates a set of keys $K_{\mathcal{GC}}^{(0)}$, and distributes them to the current group members (that may be determined by the adversary), $\Delta^{(0)} \subseteq \mathbb{ID}$, through the authenticated private channels. (If some users were corrupted before this setup procedure, we may let the adversary select the keys held by the corrupt users.) Each member $U_i \in \Delta^{(0)}$ holds a set of keys denoted by $K_{U_i}^{(0)} \subset K_{\mathcal{GC}}^{(0)}$, and there is a key, $k^{(0)}$ that is common to all the current members, namely $k^{(0)} \in K_{\mathcal{GC}}^{(0)} \cap K_{U_1}^{(0)} \cap \dots \cap K_{U_{|\Delta^{(0)}|}}^{(0)}$.

Join: This algorithm is executed by the group controller \mathcal{GC} at certain time t following a join request by a prospective member. (We abstract away the out-of-band authentication and establishment of an individual key for each new member). It takes as input: (1) a set of identities of current group members – $\Delta^{(t-1)}$, (2) identities of newly admitted group member, $\Delta' \subseteq \mathbb{ID} \setminus \Delta^{(t-1)}$, (3) keys held by the group controller, $K_{\mathcal{GC}}^{(t-1)}$, and (4) keys held by group members, $\{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}} = \{K_{U_i}^{(t-1)} : U_i \in \Delta^{(t-1)}\}$.

It outputs updated system state information, including: (1) identities of new group members, $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \cup \Delta'$, (2) new keys for \mathcal{GC} itself, $K_{\mathcal{GC}}^{(t)}$, (3) new keys for new group members, $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$, which are *somehow* sent to the legitimate users through the authenticated channels (depending on concrete schemes), (4) new group key $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \dots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$.

Formally, denote it by $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \text{Join}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$.

Leave: This algorithm is executed by the group controller \mathcal{GC} at time, say, t due to leave or revocation operation(s). It takes as input: (1) identities of previous group members, $\Delta^{(t-1)}$, (2) identities of leaving group members, $\Delta' \subseteq \Delta^{(t-1)}$, (3) keys held by the controller, $K_{\mathcal{GC}}^{(t-1)}$, and (4) keys held by group members, $\{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}}$.

It outputs updated system state information, including: (1) identities of new group members, $\Delta^{(t)} \leftarrow \Delta^{(t-1)} \setminus \Delta'$, (2) new keys for \mathcal{GC} , $K_{\mathcal{GC}}^{(t)}$, (3) new keys for new group members, $\{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}$, which are *somehow* sent to the legitimate users through the authenticated channels (depending on concrete schemes), (4) new group key $k^{(t)} \in K_{\mathcal{GC}}^{(t)} \cap K_{U_1}^{(t)} \cap \dots \cap K_{U_{|\Delta^{(t)}|}}^{(t)}$.

Formally, denote it by $(\Delta^{(t)}, K_{\mathcal{GC}}^{(t)}, \{K_{U_i}^{(t)}\}_{U_i \in \Delta^{(t)}}) \leftarrow \text{Leave}(\Delta^{(t-1)}, \Delta', K_{\mathcal{GC}}^{(t-1)}, \{K_{U_i}^{(t-1)}\}_{U_i \in \Delta^{(t-1)}})$.

Rekey: This algorithm is executed by the legitimate group members at some time t , namely all $U_i \in \Delta^{(t)}$ where $\Delta^{(t)}$ is derived from a **Join** or **Leave** event. In other words, $U_i \in \Delta^{(t)}$ runs this algorithm upon receiving the message from \mathcal{GC} over the authenticated channel. The algorithm takes as input the received message and U_i 's secrets, and is supposed to output the updated keys for the group member. If the execution of the algorithm is successful, U_i sets: (1) $\text{acc}_{U_i}^{(t)} \leftarrow \text{TRUE}$, (2) $K_{U_i}^{(t)}$, where $k_{U_i}^{(t)} \in K_{U_i}^{(t)}$ is supposed to be the new group key.

If the rekeying event is incurred by a **Join** event, every $U_i \in \Delta^{(t)}$ erases $K_{U_i}^{(t-1)}$ and any temporary storage after obtaining $K_{U_i}^{(t)}$. If the rekeying event is incurred by a **Leave** event, every $U_i \in \Delta^{(t)}$ erases $K_{U_i}^{(t-1)}$ and any temporary storage after obtaining $K_{U_i}^{(t)}$, and every *honest* leaving group member $U_j \in \Delta'$ erases $K_{U_j}^{(t-1)}$ (although a *corrupt* one does not have to follow this protocol).

We require for a CGKD scheme to be **correct**, meaning that after each rekey process, all the group members share a common key with the group controller, and **secure**, meaning that no adversary learns any information about a group key at time t_1 , even if there are corrupt users at time $t_2 > t_1$. This is the strongest notion, called *strong-security in the active outsider attack model* in [34] (somewhat surprisingly, existing popular group communication schemes do not achieve this property, but many of them can be made secure in this sense without incurring any significant extra complexity [34]). We defer formal definition and discussions to Appendix C.

6 Building Block III: Distributed Group Key Agreement

Let κ be a security parameter. We assume a polynomial-size set \mathbb{ID} of potential players. Any subset of \mathbb{ID} may decide at any point to invoke distributed group key agreement. A distributed group key agreement scheme, DGKA, is specified in Figure 5; it follows the results in [5] and [21].

Fig. 5. Definition: Distributed Group Key Agreement

Environment: Since each principal can take part in many runs of the `GroupKeyAgreement` protocol (described below), we denote an instance i of $U \in \mathbb{ID}$ as Π_U^i . Each instance Π_U^i is associated with variables $\text{acc}_U^i, \text{sid}_U^i, \text{pid}_U^i, \text{sk}_U^i$. Initially, $\forall U \in \mathbb{ID}$ and $i \in \mathbb{N}$, $\text{acc}_U^i \leftarrow \text{FALSE}$ and $\text{sid}_U^i, \text{pid}_U^i, \text{sk}_U^i \leftarrow \text{UNDEFINED}$.

GroupKeyAgreement: a protocol that performs distributed unauthenticated (or “raw”) group agreement between any set of $m \geq 2$ parties. After executing the protocol, each party outputs an indication of the protocol outcome (success or failure), and some secret information, in case of success. In more detail, the protocol is executed by m instances: $\Pi_{U_1}^{i_1}, \dots, \Pi_{U_m}^{i_m}$, where $\{U_1, \dots, U_m\} \subseteq \mathbb{ID}$. If the execution of $\Pi_{U_j}^{i_j}$ is successful, it sets:

1. $\text{acc}_{U_j}^{i_j} \leftarrow \text{TRUE}$,
2. $\text{sid}_{U_j}^{i_j}$ as the session id of instance $\Pi_{U_j}^{i_j}$, namely a protocol-specific function of all communication sent and received by $\Pi_{U_j}^{i_j}$ (e.g., we can simply set $\text{sid}_{U_j}^{i_j}$ as the concatenation of all messages sent and received by $\Pi_{U_j}^{i_j}$ in the course of its execution),
3. $\text{pid}_{U_j}^{i_j}$ as the session id of instance $\Pi_{U_j}^{i_j}$, namely the identities of the principals in the group with whom $\Pi_{U_j}^{i_j}$ intends to establish a session key (including U_j itself), and (4) $\text{sk}_{U_j}^{i_j}$ as the newly established session key.

Remark: We stress that this definition does not offer any *authentication*, i.e., it does not capture *authenticated group key agreement*. For example, the above definition can be satisfied (instantiated) with a straight-forward extension to any of the several group Diffie-Hellman protocols, such as BD or GDH [11, 30]. Of course, we are aware that unauthenticated key agreement protocols are susceptible to man-in-the-middle (MITM) attacks; this is addressed later, through the use of our second building block – CGKD.

Informally speaking (see Appendix D for a formal definition), we require for a scheme to have **correctness** and **security**. **Correctness** means that all participants must obtain the same new session secret (key), and **security** means that a passive adversary – who does not compromise any principal during protocol execution – does not learn any information about the new group session key.

7 GCD Secret Handshake Framework

The **GCD** framework has the following components:

GCD.CreateGroup: The group authority (\mathcal{GA}) plays the roles of both group manager in **GSIG** and group controller in **CGKD**.

- \mathcal{GA} executes **GSIG.Setup**. This initializes a group signature scheme.
- \mathcal{GA} executes **CGKD.Setup**. This initializes a centralized group key distribution (broadcast encryption) scheme.
- \mathcal{GA} generates a pair of public/private keys (pk_T, sk_T) with respect to an IND-CCA2 secure public key cryptosystem. This pair of keys enables \mathcal{GA} to identify handshake participants in any handshake transcript.
- Note that no real group-specific setup is required for initializing the distributed group key agreement component – **DGKA**. We assume that there is a set of system-wide (not group-specific) cryptographic parameters for the

DGKA scheme, e.g., all groups use the same group key agreement protocol with the same global parameters.
(More on this below.)

GCD.AdmitMember: \mathcal{GA} executes CGKD.Join and GSIG.Join.

CGKD.Join results in a new group key and GSIG.Join causes an update to GSIG state information. The updated GSIG state information is encrypted under the new CGKD group key and distributed to all group members through an authenticated anonymous channel, e.g., posted on a public bulletin board.

GCD.RemoveUser: \mathcal{GA} executes CGKD.Leave and GSIG.Revoke, except: (1) the updated system state information corresponding to GSIG is encrypted under CGKD's new group session key, and distributed as part of CGKD's state information updating message, and (2) the update messages are distributed via an authenticated anonymous channel.

GCD.Update: All non-revoked members execute GSIG.Update and CGKD.Rekey, except: (1) the updated system state information is obtained from an authenticated anonymous channel, and (2) if CGKD.Rekey succeeds, the update information corresponding to GSIG is decrypted using CGKD's new group key.

GCD.Handshake: Suppose m (≥ 2) users want to determine if they belong to the same group. We denote their group keys with respect to CGKD as: k_1, \dots, k_m , respectively. Note that, if they belong to the same group, then $k_1 = \dots = k_m$.

Phase I: Preparation: All m parties jointly execute DGKA.GroupKeyAgreement. We denote the resulting keys as: k_1^*, \dots, k_m^* , respectively. If the execution is successful, then $k_1^* = \dots = k_m^*$, and each party computes $k'_i = k_i^* \oplus k_i$.

Phase II: Preliminary Handshake: Each party publishes a tag $\text{MAC}(k'_i, s, i)$ corresponding to a message authentication code MAC (e.g., HMAC-SHA1), where s is a string unique to party i , e.g., the message(s) it sent in the DGKA.GroupKeyAgreement execution.⁷

Phase III: Full Handshake: There are two cases:

CASE 1: If all message authentication tags are valid (i.e., they belong to the same group), each party executes the following:

1. Encrypt k'_i to obtain ciphertext δ_i under the group authority's tracing public key pk_T ; $\delta_i \leftarrow \text{ENC}(pk_T, k'_i)$.
2. Generate a group signature σ_i on δ_i via GSIG.Sign.
3. Encrypt σ_i using a symmetric key encryption algorithm and key k'_i to obtain a ciphertext θ_i ; $\theta_i \leftarrow \text{SENC}(k'_i, \sigma_i)$.
4. Publish (θ_i, δ_i) .
5. Upon receiving (θ_i, δ_i) , execute the following:
 - Obtain the group signature by performing symmetric key decryption algorithm using k'_i ; $\sigma_i \leftarrow \text{SDEC}(k'_i, \theta_i)$.

⁷ If a broadcast channel is available, the tag is sent on it; else, it is sent point-to-point.

- Run `GSIG.Verify` to check if σ_i is a valid group signature. If all group signatures are deemed valid, the party concludes that the corresponding parties all belong to the same group and stores the transcript including $\{(\theta_i, \delta_i)\}_{1 \leq i \leq m}$.

CASE 2: If at least one message authentication tag is invalid, each of party picks and publishes a pair (θ_i, δ_i) randomly selected from the ciphertext spaces corresponding to the symmetric key and public key cryptosystems, respectively.

GCD.TraceUser: Given a transcript of a secret handshake instance: $\{(\theta_i, \delta_i)\}_{1 \leq i \leq m}$, the group authority \mathcal{GA} decrypts all δ_i 's to obtain the corresponding session keys: k'_1, \dots, k'_m . In the worst case, the authority needs to try to search the right session key and decrypt all θ_i 's to obtain the cleartext group signatures. Then, it executes `GSIG.Open` to identify the handshake parties.

Remark: In order to enable modular construction, we specify the handshake protocol as a three-phase protocol. Thus, the resulting framework is flexible, i.e., tailorable to application semantics. For example, if traceability is not required, a handshake may only involve Phase I and Phase II.

The following theorem is proved in Appendix E.

Theorem 1. *Assume that GSIG possesses the properties of correctness, full-traceability, full-anonymity, and no-misattribution specified in Section 4. Assume also that DGKA and CGKD are secure with respect to their corresponding definitions in Sections 5-6. Then, the GCD framework possesses the properties of correctness, resistance to impersonation, resistance to detection, full-unlinkability, indistinguishability to eavesdroppers, traceability, and no-misattribution specified in Section 2.*

The following theorem is proved in Appendix F.

Theorem 2. *Assume that GSIG possesses the properties of correctness, full-traceability, anonymity, and no-misattribution specified in Section 4. Assume also that DGKA and CGKD are secure with respect to their corresponding definitions in Sections 5-6. Then, the GCD framework possesses the properties of correctness, resistance to impersonation, resistance to detection, unlinkability, indistinguishability to eavesdroppers, traceability, and no-misattribution specified in Section 2.*

Extension: A natural extension of the above framework can fulfill the aforementioned *partially-successful* secret handshakes, namely that all such $\Delta \subset \{1, \dots, m\}$ that consists of $|\Delta| > 1$ members of a same group can always succeed in their handshakes without incurring any extra complexity. Each participant i can immediately tell the Δ such that $i \in \Delta$ as i knows which message authentication tags are valid.

8 Two Concrete Instantiations

We now present two concrete secret handshake schemes. The first scheme employs “raw” (unauthenticated) contributory group key agreement, and the second scheme ensures that all handshake participants are distinct.

8.1 Example Scheme 1

This is a straight-forward instantiation of the **GCD** framework. We simply plug in unauthenticated group key agreement (DGKA) derived from any of [11, 30, 21], CGKD based on [33, 26], and GSIG based on [1, 12]. Theorem 1 immediately implies that this instantiation satisfies all properties specified in Section 2, excluding **self-distinction**.

Computational complexity for each party is the sum of the respective complexities incurred in each of the three building blocks. Note that, in an m -party handshake, each party only needs to compute $O(m)$ modular exponentiations in total. Moreover, the communication complexity is $O(m)$ per-user in number of messages.

8.2 Example Scheme 2

As mentioned above, the first instantiation does not offer self-distinction, i.e., some of the m parties in a handshake protocol could in fact be “played” by the same party. We now discuss the basic idea for attaining the self-distinction property. Naturally, neither group key agreement nor centralized key distribution (i.e., the CGKD and DGKA components) can provide self-distinction. Thus, we turn to group signatures to obtain it. However, group signature schemes do not natively offer self-distinction since it runs against one of their basic tenets, informally summarized as:

Given any two group signatures it should be impossible to determine with any certainty whether the same signer (or two distinct signers) generated both signatures

Nonetheless, the need for self-distinction in group signatures (not in secret handshakes) has been recognized prior to this paper. In particular, the work in [2] introduces the concept of *subgroup signatures* motivated by certain applications, such as anonymous petitions. (In an anonymous petition, t group members want to sign a document in a way that any verifier can determine with certainty that all t signers are distinct.) The example technique for constructing sub-group signatures in [2] involves slight modifications to the underlying group signature scheme. This is very similar to what we need to achieve self-distinction in the proposed framework.

Unfortunately, we cannot use the example in [2] since it is based on a group signature scheme [13] which is inefficient and not provably secure. However, we can modify a more recent (as well as much more efficient and provably secure) group signature scheme by Kiayas and Yung [22] (see Appendix H). In this scheme each group signature is accompanied by a pair: $\langle T_6 = g^{\beta\alpha}, T_7 = g^\alpha \rangle$, where β is the signer’s secret, and α is the signer’s randomness for this specific signature. This structure has a nice feature that T_7 serves only as an “anonymity shield” in the sense that the signer does not even need to prove the knowledge of α . Instead, it is crucial that $T_6 = T_7^\beta$. Intuitively, this allows us to obtain **self-distinction** if we let each handshake participant use the same T_7 , since they should all provide distinct T_6 ’s. This can be achieved by simply utilizing an idealized hash function [6] $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbf{R}$ to the input of, for instance, the concatenation of all messages sent by the handshake participants.⁸ This ensures that, as long as there is at least

⁸ While it would suffice for \mathbf{R} to be $QR(n)$, what is needed in [22] is in fact that $g \in QR(n)$ and α is chosen from an appropriate interval, i.e., $\mathbf{R} \subset QR(n)$.

one honest participant, the resulting T_7 is uniformly distributed over \mathbf{R} , and the security proof in [22] remain sufficient for our purposes.

We now present a scheme based on the modified version of [22]. We only illustrate the handshake protocol (in Figure 6) since it is the only distinctive feature of this scheme.

Fig. 6. Example Scheme 2: Self-Distinction

GCD.Handshake: Assume m (≥ 2) users are taking part in the protocol. We denote their group keys with respect to the CGKD by k_1, \dots, k_m , respectively. As before, if they belong to the same group, $k_1 = \dots = k_m$.

Phase I: Preparation: m parties jointly execute **DGKA.GroupKeyAgreement**. Let the resulting keys be denoted as: k_1^*, \dots, k_m^* , respectively. (After a successful run $k_1^* = \dots = k_m^*$.) Then, each party computes $k'_i = k_i^* \oplus k_i$

Phase II: Preliminary Handshake: Each party publishes a pair $\text{MAC}(k'_i, s, i)$, where s is a string unique to i .

Phase III: Full Handshake: We consider two cases:

Case 1: If all message authentication tags are valid (i.e., they belong to the same group), each party executes as follows:

1. Encrypt k'_i under the public key pk_T to obtain ciphertext δ_i .
2. Generate a variant group signature σ_i on δ_i on s via **GSIG.Sign**, which is the same as in [22] except that T_7 is chosen using an ideal hash function as discussed above, and the *same* T_7 is common to all handshake participants. (We stress that **self-distinction** is obtained from requiring all participants to use the same T_7 which forces them to compute distinct T_6 values.)
3. Encrypt σ_i using a symmetric key encryption algorithm and key k'_i to obtain ciphertext $\theta_i \leftarrow \text{SENC}(k'_i, \sigma_i)$.
4. Publish (θ_i, δ_i) .
5. Upon receiving (θ_i, δ_i) , execute as follows:
 - (a) Obtain $\sigma_i \leftarrow \text{SDEC}(k'_i, \theta_i)$
 - (b) Run **GSIG.Verify** to check if each σ_i is a valid group signature (if all group signatures are valid, it concludes that they all belong to the same group and records the transcript).

Case 2: If at least one message authentication tag is invalid, each party simulates the above execution of the Pederson protocol and then picks a pair of (θ_i, δ_i) randomly selected from the ciphertext spaces corresponding to the symmetric key and public key cryptosystems, respectively.

A proof sketch of the following theorem can be found in Appendix G.

Theorem 3. *Assume that GSIG possesses the properties of correctness, full-traceability, anonymity, and no-misattribution specified in Section 4. Assume also that DGKA and CGKD are secure with respect to their corresponding definitions in Sections 5-6. Then, the above instantiation possesses the properties of correctness, resistance to impersonation, resistance to detection, unlinkability, indistinguishability to eavesdroppers, no-misattribution, traceability, and self-distinction specified in Section 2.*

Computational complexity in number of modular exponentiations (per-user) remains $O(m)$ and communication complexity (also per-user) in number of messages also $O(m)$, where m is the number of participants.

9 Discussion

There are several practical issues that need to be addressed. First, if there is only a single group that uses a secret handshake scheme, an adversary can simply figure out that the handshake peers belong to that group. In fact, if a secret handshake scheme is implemented as a TLS or IKE cipher suite, then the parties will exchange a cipher suite

designator that clearly shows that they wish to engage in a secret handshake. Second, in any secret handshake scheme, utilizing one-time or reusable credentials alike, it is assumed that there is no easy way to identify the party who sent or received a certain message; otherwise, it is easy for an adversary to discover who is interacting with whom. This assumption is also true in privacy-preserving authentication mechanisms [24, 8, 16, 17, 28, 27]. Third, if an adversary observes that handshake participants continue communicating after finishing the handshake protocol, it can deduce that they belong to the same group. (This applies to any secret handshake scheme utilizing one-time or reusable credentials.)

The above issues can be mitigated by various means. First, it is reasonable to assume that there are many groups, as long as it is not illegal to conduct secret handshakes. Second, there may be settings where the identity (for the purpose of authentication) of a party is not directly derivable from the address that must appear in the clear in protocol messages. A common example is the case of mobile devices wishing to prevent an attacker from correlating their (changing) locations with the device’s logical identity [24]. Furthermore, some form of anonymous communication could make it hard to decide exactly who is engaging in a secret handshake. Third, protection against traffic analysis (e.g., an adversary simply observing continued communication after a handshake) can be achieved by utilizing mechanisms such as steganographic techniques, or anonymous communication channels.

In summary, if all assumptions are satisfied, then our secret handshake schemes (as well as [3, 24, 8]) can provide provable privacy-preserving authentication, whereby two (or in our case, more) participants authenticate each other’s membership *simultaneously*. Otherwise, all schemes attain heuristic or *best-effort* anonymity.

10 Related Work

The first secret handshake scheme [3] is based on the protocol of Sakai et al. [29], which targets the key exchange problem. Indeed, a secret handshake can be appropriately turned into an authenticated key exchange, but an authenticated key exchange does not necessarily imply a secret handshake, e.g., the two-party Diffie-Hellman key agreement scheme [18] does not lend itself to solving the secret handshake problem; see [3]. The scheme in [3] is based on bilinear maps in the setting of elliptic curves and its security is based on the associated assumptions. This scheme uses one-time pseudonyms to achieve unlinkability and does not offer the *No-misattribution* property.

A more recent follow-on result is due to Castelluccia, et al. [14]. This work constructs several handshake schemes in more standard cryptographic settings (avoiding bilinear maps) and provides some extensions for satisfying *No-misattribution*. However, it still relies on one-time pseudonyms to satisfy unlinkability. Another recent result by [35] requires each player to be aware of the information of other groups and offers weaker anonymity (referred to as *k-anonymity*).

11 Conclusions and Future Work

To summarize, we present **GCD**— a flexible secret handshake framework. **GCD** is a compiler that can transform a group signature scheme, a centralized group key distribution scheme, and a distributed group key agreement scheme into a secure secret handshake scheme. As illustrated by three concrete examples, **GCD** lends itself to actual practical instantiations and offers several interesting new features. **GCD** avoids the use of one-time pseudonyms and, unlike prior techniques, supports handshakes among an arbitrary number of parties. Furthermore, **GCD** can be instantiated to support the important new property of **self-distinction** important in handshakes of more than two participants.

We believe that the work described in this paper is a *first* step towards achieving practical anonymous interactive multi-party authentication protocols. Much remains to be done. First, the **GCD** framework needs to be implemented and experimented with. Second, we have made no attempt to optimize the efficiency of the framework. Further investigation is clearly called for. Third, efficient constructions are needed for those settings where the **GCD** framework does not apply (because, e.g., the lack of a centralized group key distribution scheme).

References

1. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Proc. CRYPTO 2000*, pages 255–270. Springer, 2000. Lecture Notes in Computer Science No. 1880.
2. G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. Financial Cryptography’99, February 1999. Lecture Notes in Computer Science No. 1880.
3. D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *24th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.
4. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
5. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. EUROCRYPT 2000*, pages 139–155. Springer, 2000. Lecture Notes in Computer Science No. 1807.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.
7. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. Cryptology ePrint Archive, Report 2004/077, 2004. <http://eprint.iacr.org/>.
8. C. Boyd, W. Mao, and K. Paterson. Deniable authenticated key establishment for internet protocols.
9. R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS’04)*, pages 146–157. ACM Press, 2004.
10. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In L. R. Knudsen, editor, *Proc. of Eurocrypt 02*, volume 2332 of *LNCS*, page 321336, Amsterdam, NL, AprilMay 2002. IACR, Springer-Verlag.
11. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. D. Santis, editor, *Proc. EUROCRYPT 94*, pages 275–286. Springer, 1994. Lecture Notes in Computer Science No. 950.
12. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In M. Yung, editor, *Proc. CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer-Verlag, 2002.
13. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B.S. Kaliski Jr., editor, *Proc. CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.
14. C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2004.
15. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, Feb. 1981.
16. D. Chaum. Blind signatures for untraceable payments. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 199–203, New York, 1983. Plenum Press.
17. D. Chaum and E. V. Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology — Eurocrypt ’91*, pages 257–265, Berlin, 1991. Springer-Verlag. Lecture Notes in Computer Science No. 547.

18. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, Nov. 1976.
19. J. Douceur. The sybil attack. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'01)*, pages 251–260, London, UK, 2002. Springer-Verlag.
20. M. Fischer, N. Lynch, and M. Patterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
21. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *Proc. CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer-Verlag, 2002.
22. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2004.
23. A. Kiayias and M. Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. <http://eprint.iacr.org/>.
24. H. Krawczyk. Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike-protocols. In D. Boneh, editor, *Proc. CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425. Springer-Verlag, 2002.
25. N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 182–189. ACM, 2003.
26. D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Proc. CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer-Verlag, 2001.
27. M. Naor. Deniable ring authentication. In M. Yung, editor, *Proc. CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer-Verlag, 2002.
28. R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology-ASIACRYPT '2001*, pages 552–565.
29. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the Symposium on Cryptography and Information Security (SCIS)*, 2002.
30. M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Trans. on Parallel and Distributed Systems*, 11(8):769–780, 2000.
31. Y. Sun and K. Liu. Securing dynamic membership information in multicast communications. In *IEEE Infocom'04*, pages ???–???
32. D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. Internet Draft, Sept. 1998.
33. C. Wong, M. Gouda, and S. Lam. Secure group communication using key graphs. *IEEE/ACM Transactions on Networking* (Preliminary version in SIGCOMM'98), 8, 2000.
34. S. Xu. On the security of group communication schemes based on symmetric key cryptosystems. In *Proceedings of the Third ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05)*, pages ??–?? ACM, 2005.
35. S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pages 158–167. ACM Press, 2004.

A Security Properties of Secret Handshake Schemes

Consider a probabilistic polynomial-time adversary \mathcal{A} that is allowed access to the following oracles:

- $\mathcal{O}_{CG}(\cdot)$: activate a new \mathcal{GA} to create a group via `SHS.CreateGroup`. The identity, \mathcal{GA} , may be given by \mathcal{A} as input. We assume that a \mathcal{GA} is not under \mathcal{A} 's control before the group is established. However, \mathcal{GA} may be corrupted immediately after its establishment, i.e., before any users are admitted into the group.
- $\mathcal{O}_{AM}(\cdot, \cdot)$: input includes the identity of a \mathcal{GA} and, optionally, the identity U of a user under \mathcal{A} 's control. In case of $\mathcal{O}_{AM}(\mathcal{GA}, U)$, \mathcal{GA} may admit the corrupt user U by executing `SHS.AdmitMember`; in case of $\mathcal{O}_{AM}(\mathcal{GA})$, \mathcal{GA} executes `SHS.AdmitMember` to admit an honest user and assign it a unique pseudonym U , if needed.
- $\mathcal{O}_{RU}(\cdot, \cdot)$: input includes the identity of a \mathcal{GA} and a pseudonym U . The oracle activates `SHS.RemoveUser` to place U onto the corresponding \mathcal{CRL} . The system state information is appropriately updated.
- $\mathcal{O}_{Update}()$: whenever $\mathcal{O}_{AM}(\cdot, \cdot)$ or $\mathcal{O}_{RU}(\cdot, \cdot)$ is called, this oracle is also called so that the corresponding non-revoked group members can update their system state information.
- $\mathcal{O}_{HS}(\cdot, \dots, \cdot)$: activate `SHS.Handshake` between members U_1, \dots, U_m , where none or some (but not all) are under \mathcal{A} 's control. The honest members execute according to the protocol.
- $\mathcal{O}_{TU}(\cdot)$: on input of a successful handshake transcript, \mathcal{GA} identifies the participating members.
- $\mathcal{O}_{Corrupt}(\cdot, \cdot)$: takes as input the identity of \mathcal{GA} and, optionally, a pseudonym in $\mathcal{U} \cup \{\perp, \top\}$. In case of $\mathcal{O}_{Corrupt}(\mathcal{GA}, U \in \mathcal{U})$, the oracle returns U 's current internal state information (including all secrets) to \mathcal{A} ; in case of $\mathcal{O}_{Corrupt}(\mathcal{GA}, \perp)$, the oracle returns \mathcal{GA} 's current internal state information for admitting members to \mathcal{A} ; in case of $\mathcal{O}_{Corrupt}(\mathcal{GA}, \top)$, the oracle returns \mathcal{GA} 's current internal state information for tracing members to \mathcal{A} . Once \mathcal{GA} or U is corrupt, it executes according to \mathcal{A} 's wishes until the corruption is detected by some outside mechanism (e.g., intrusion detection systems). When the corruption of user U is detected, it is excluded from the group via `SHS.RemoveUser`; when the corruption of \mathcal{GA} is detected, the corresponding group is simply excluded from the system.

We now specify the desired security properties for a secret handshake scheme. We note that they form a superset of security properties found in prior work [3, 35, 14].

Correctness: Suppose a set of participants, $\Delta = \{U_1, \dots, U_m\}$, are conducting the secret handshake protocol. If all the U_i 's belong to the same group, then $\text{Handshake}(\Delta)$ always returns “1”; otherwise, it always returns “0”.

Resistance to impersonation: no adversary $\mathcal{A} \notin G$ who does not corrupt any members of G has only a negligible probability in convincing an honest user $U \in G$ that $\mathcal{A} \in G$. The remains to be true even if \mathcal{A} plays the roles of multiple participants. Formally, consider the following game or experiment.

Experiment $\text{RIA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

$(\mathcal{G}\mathcal{A}, \Delta' = \{U'_i\}_{i \in \mathbb{I}}, b) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{U\text{pdate}}(\cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$

Return “1” if the following holds and “0” otherwise:

- (1) there is no $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A})$ query
- (2) if there is an $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, X \in \mathcal{U})$ query, then there is also an $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, X)$ query
- (3) if there is an $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A}, X \in \mathcal{U})$ query, then there is also an $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, X)$ query
- (4) $\forall i \in \mathbb{I}$: either U'_i belongs to the group managed by $\mathcal{G}\mathcal{A}$ or $U'_i = \mathcal{A}$
- (5) $\forall i \in \mathbb{I}$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, U'_i)$ query
- (6) $\exists i \in \mathbb{I}$: there is no $\mathcal{O}_{Corrupt}(U'_i)$ query
- (7) $|\Delta'| = m - 1$ and $\Delta = \{\mathcal{A}\} \cup \Delta' = \{U_1, \dots, U_{b-1}, \mathcal{A}, U_{b+1}, \dots, U_m\}$
- (8) $\text{SHS.Handshake}(\Delta)$ returns “1”

Let $\text{AdvRIA}_{\text{SHS}, \mathcal{A}}(1^\kappa) = \Pr[\text{RIA}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns “1”}]$, which is the probability that the experiment $\text{RIA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ returns “1”, where the probability is taken over the coins used for selecting the cryptographic parameters, the coins used by the oracles, and the coins used by the adversary. A secret handshake scheme SHS is “**resistant to impersonation attacks**” if for $\forall \mathcal{A}$, $\text{AdvRIA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ is negligible in κ .

Resistance to detection: no adversary $\mathcal{A} \notin G$ can tell it is interacting with an honest user $U \in G$ or a simulator. The remains to be true even if \mathcal{A} plays the roles of multiple participants. Formally, consider the following game or experiment.

Experiment $\text{RDA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

$\text{bit} \in_R \{0, 1\}$

$(\mathcal{G}\mathcal{A}, \Delta' = \{U'_i\}_{i \in \mathbb{I}}, B = \{b_1, \dots, b_\ell\}, \text{stateinfo}) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{U\text{pdate}}(\cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$

where $|\Delta'| = m - \ell > 0$ and $\Delta' \subseteq G$ for G managed by $\mathcal{G}\mathcal{A}$

set $\Delta = \Delta' \cup \{\mathcal{A}\} = \{U_1, \dots, U_{b_1-1}, \mathcal{A}, U_{b_1+1}, \dots, U_{b_\ell-1}, \mathcal{A}, U_{b_\ell+1}, \dots, U_m\}$

if $\text{bit} == 0$ execute $\text{SHS.Handshake}(U_1, \dots, U_{b_1-1}, \mathcal{A}, U_{b_1+1}, \dots, U_{b_\ell-1}, \mathcal{A}, U_{b_\ell+1}, \dots, U_m)$

else execute $\text{SHS.Handshake}(\text{SIM}_1, \dots, \text{SIM}_{b_1-1}, \mathcal{A}, \text{SIM}_{b_1+1}, \dots, \text{SIM}_{b_\ell-1}, \mathcal{A}, \text{SIM}_{b_\ell+1}, \dots, \text{SIM}_m)$

where the SIM_i 's are simulators

$\text{bit}' \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{U\text{pdate}}(\cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo})$

Return “1” if the following hold and “0” otherwise

- (1) there is no $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A})$ query
- (2) if there is an $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, X \in \mathcal{U})$ query, then there is also an $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, X)$ query
- (3) if there is an $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A}, X \in \mathcal{U})$ query, then there is also an $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, X)$ query
- (4) $\forall i \in \mathbb{I}$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, U'_i)$ query
- (5) $\text{bit} == \text{bit}'$

Let $\text{AdvRDA}_{\text{SHS}, \mathcal{A}}(\kappa) = |\Pr[\text{RDA}_{\text{SHS}, \mathcal{A}}(\kappa) \text{ returns “1”} | \text{bit} = 1] - \Pr[\text{RDA}_{\text{SHS}, \mathcal{A}}(\kappa) \text{ returns “1”} | \text{bit} = 0]|$. A scheme SHS is “**resistant to detection**” if for $\forall \mathcal{A}$, $\text{AdvRDA}_{\text{SHS}, \mathcal{A}}(\kappa)$ is negligible in κ .

Full-unlinkability: no adversary \mathcal{A} is able to associate two handshakes involving a same honest user $U \in G$, even if $\mathcal{A} \in G$ and \mathcal{A} participated in both executions and U has indeed been corrupt. Of course, the adversary is assumed not to reside on any corrupt machine. The remains to be true even if \mathcal{A} plays the roles of multiple participants. Formally, consider the following experiment.

Experiment $\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

$\text{bit} \in_R \{0, 1\}$

$(\Delta_0 = \{U_1, \dots, U_{m_0}\}, \text{stateinfo}_0) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$

$(\Delta_1 = \{U'_1, \dots, U'_{m_1}\}, \text{stateinfo}_1) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo}_0)$

if $\text{bit} == 0$ then execute

$\text{Handshake}(\{U_1, \dots, U_{i-1}, U_i, U_{i+1}, U_{m_0}\})$ and

$\text{Handshake}(\{U'_1, \dots, U'_{j-1}, U'_j, U'_{j+1}, \dots, U'_{m_1}\})$

else execute

$\text{Handshake}(\{U_1, \dots, U_{i-1}, \text{SIM}, U_{i+1}, U_{m_0}\})$ and

$\text{Handshake}(\{U'_1, \dots, U'_{j-1}, \text{SIM}, U'_{j+1}, \dots, U'_{m_1}\})$

Return “1” if the following hold and “0” otherwise:

- (1) $\text{bit}' \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo}_0, \text{stateinfo}_1)$
- (2) $\forall U \in \Delta_0$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, U)$ query
where $\mathcal{G}\mathcal{A}$ is the group authority of the group to which U belongs
- (3) $\forall U' \in \Delta_1$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}', U')$ query
where $\mathcal{G}\mathcal{A}'$ is the group authority of the group to which U' belongs
- (4) $U_i = U'_j$
- (5) there is no $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, U_i)$ query
- (6) $\text{bit} == \text{bit}'$

Let $\text{AdvUnlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) = |\Pr[\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \text{bit} = 1] - \Pr[\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \text{bit} = 0]|$, where the probability is taken over all tossed coins. A scheme SHS is “fully-unlinkable” if for $\forall \mathcal{A}$, $\text{AdvUnlink}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ is negligible in κ .

Unlinkability: no adversary \mathcal{A} is able to associate two handshakes involving a same honest user $U \in G$, even if $\mathcal{A} \in G$ and \mathcal{A} participated in both executions. Of course, there should be at least two uncorrupt machines; otherwise, the adversary could trivially deduce that the two executions involve the same honest member. The remains to be true even if \mathcal{A} plays the roles of multiple participants. Formally, consider the following experiment.

Experiment $\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

- $\text{bit} \in_R \{0, 1\}$
- $(\Delta_0 = \{U_1, \dots, U_{m_0}\}, \text{stateinfo}_0) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$
- $(\Delta_1 = \{U'_1, \dots, U'_{m_1}\}, \text{stateinfo}_1) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo}_0)$
- if $\text{bit} == 0$ then execute
 - Handshake($\{U_1, \dots, U_{i-1}, U_i, U_{i+1}, U_{m_0}\}$) and
 - Handshake($\{U'_1, \dots, U'_{j-1}, U'_j, U'_{j+1}, \dots, U'_{m_1}\}$)
- else execute
 - Handshake($\{U_1, \dots, U_{i-1}, \text{SIM}, U_{i+1}, U_{m_0}\}$) and
 - Handshake($\{U'_1, \dots, U'_{j-1}, \text{SIM}, U'_{j+1}, \dots, U'_{m_1}\}$)
- Return “1” if the following hold and “0” otherwise:
 - (1) $\text{bit}' \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo}_0, \text{stateinfo}_1)$
 - (2) $\forall U \in \Delta_0$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, U)$ query
where $\mathcal{G}\mathcal{A}$ is the group authority of the group to which U belongs
 - (3) $\forall U' \in \Delta_1$: there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}', U')$ query
where $\mathcal{G}\mathcal{A}'$ is the group authority of the group to which U' belongs
 - (4) $U_i = U'_j$
 - (5) there is no $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, U_i)$ query
 - (6) there is no $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A}, U_i)$ query
 - (7) $\exists V \neq U_i$: V belongs to the group managed by $\mathcal{G}\mathcal{A}$ and there is neither $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A}, V)$ nor $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, V)$ query
 - (8) $\text{bit} == \text{bit}'$

Let $\text{AdvUnlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) = |\Pr[\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \text{bit} = 1] - \Pr[\text{Unlink}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \text{bit} = 0]|$, where the probability is taken over all tossed coins. A scheme SHS is “unlinkable” if for $\forall \mathcal{A}$, $\text{AdvUnlink}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ is negligible in κ .

Indistinguishability to eavesdroppers: no adversary \mathcal{A} who does not participate in a handshake protocol can tell a successful handshake between $\{U_1, \dots, U_m\}$ from an unsuccessful one, even if $\mathcal{A} \in G$. Formally, consider the following experiment.

Experiment $\text{INDeav}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

- $\text{bit} \in_R \{0, 1\}$
- $(\mathcal{G}\mathcal{A}, \Delta = \{U_1, \dots, U_m\}, \text{stateInfo}) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$
- if $\text{bit} == 0$ then execute $\text{SHS.Handshake}(U_1, \dots, U_m)$
- else execute $\text{SHS.Handshake}(\text{SIM}, \dots, \text{SIM})$ where SIM is a simulator
- $\text{bit}' \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(\text{stateinfo})$
- Return “1” if the following hold and “0” otherwise:
 - (1) $\text{bit} == \text{bit}'$
 - (2) $\forall U \in \Delta$, there is no $\mathcal{O}_{RU}(\mathcal{G}\mathcal{A}, U)$ query
 - (3) there is no $\mathcal{O}_{Corrupt}(\mathcal{G}\mathcal{A}, \top)$ query
 - (4) if there is an $\mathcal{O}_{AM}(\mathcal{G}\mathcal{A}, X)$ query, then $X \notin \Delta$ query

Let $\text{AdvINDeav}_{\text{SHS}, \mathcal{A}} = |\Pr[\text{INDeav}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"}] - 1/2|$, where the probability is taken over all flipped coins. A scheme SHS is “indistinguishable to eavesdroppers” if for $\forall \mathcal{A}$, $\text{AdvINDeav}_{\text{SHS}}(\mathcal{A})$ is negligible in κ .

Traceability: $\mathcal{G}\mathcal{A}$ can trace all users involved in a given handshake session, as long as one of them is honest. Formally, consider the following experiment.

Experiment $\text{TraceUsers}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:

- $(\mathcal{G}\mathcal{A}, \Delta = \{U_1, \dots, U_m\}, \text{stateInfo}) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Update}(), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$

where $\Delta \subseteq G$ for G managed by \mathcal{GA}
 execute $\text{Handshake}(U_1, \dots, U_m)$ to obtain a transcript τ
 Return “1” if the following hold and “0” otherwise

- (1) there is no $\mathcal{O}_{\text{Corrupt}}(\mathcal{GA}, \top)$ query
- (2) there exists $U \in \Delta$ such that $\neg \mathcal{O}_{\text{Corrupt}}(\mathcal{GA}, U)$ or $\neg \mathcal{O}_{AM}(\mathcal{GA}, U)$
- (3) $\Delta' \leftarrow \text{TraceUser}(\tau)$
- (4) $\Delta' \setminus \Delta \neq \emptyset$

Let $\text{AdvTraceUser}_{\text{SHS}, \mathcal{A}} = |\Pr[\text{TraceUser}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns “1”}]|$, where the probability is taken over the choice of the cryptographic context, and the coins used in the experiment. A scheme **SHS** achieves “**traceability**” if for $\forall \mathcal{A}$, $\text{AdvTraceUser}_{\text{SHS}, \mathcal{A}}$ is negligible in κ .

No-misattribution: no participant, no coalition of malicious parties (including any number of group members and the \mathcal{GA}) is able to frame an honest member as being involved in a secret database. Formally, consider the following experiment.

Experiment $\text{Misattribution}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:
 $(\mathcal{GA}, \Delta = \{U_1, \dots, U_m\}) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Update}(), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$
 where $\Delta \subseteq G$ for G managed by \mathcal{GA}
 execute $\text{Handshake}(U_1, \dots, U_m)$ to obtain a transcript τ
 Return “1” if the following hold and “0” otherwise

- (1) $\Delta' \leftarrow \text{TraceUser}(\tau)$
- (2) $U \in \Delta' \setminus \Delta$
- (3) there is no $\mathcal{O}_{\text{Corrupt}}(U)$ or $\mathcal{O}_{AM}(\mathcal{GA}, U)$ query

Let $\text{AdvMisattribution}_{\text{SHS}, \mathcal{A}} = |\Pr[\text{Misattribution}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns “1”}]|$, where the probability is taken over the choice of the cryptographic context, and the coins used in the experiment. A scheme **SHS** achieves “**No-misattribution**” if for $\forall \mathcal{A}$, $\text{Misattribution}_{\text{SHS}, \mathcal{A}}$ is negligible in κ .

Self-distinction: if more than 2 parties are involved in a handshake, each participant is ensured that all other participants are different. Formally, consider the following experiment.

Experiment $\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa)$:
 $(\mathcal{GA}, \Delta = \{U_1, \dots, U_m\}) \leftarrow \mathcal{A}^{\mathcal{O}_{CG}(\cdot), \mathcal{O}_{AM}(\cdot, \cdot), \mathcal{O}_{RU}(\cdot, \cdot), \mathcal{O}_{Update}(), \mathcal{O}_{HS}(\cdot, \cdot), \mathcal{O}_{TU}(\cdot), \mathcal{O}_{Corrupt}(\cdot, \cdot)}(1^\kappa)$
 where $\Delta \subseteq G$ for G managed by \mathcal{GA}
 execute $\text{Handshake}(U_1, \dots, U_m)$ to obtain a transcript τ
 $\Delta' \leftarrow \text{TraceUser}(\tau)$
 Return “1” if the following holds and “0” otherwise

- (1) there is no $\mathcal{O}_{\text{Corrupt}}(\mathcal{GA}, \top)$ query
- (2) $\exists U \in \Delta$ s.t. there is no $\mathcal{O}_{\text{Corrupt}}(\mathcal{GA}, U)$ query
- (3) $|\Delta'| < |\Delta|$

Let $\text{AdvSelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) = |\Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns “1”}]|$, where the probability is taken over the choice of the cryptographic context, and the coins used in the experiment. A scheme **SHS** achieves “**self-distinction**” if for $\forall \mathcal{A}$, $\text{AdvSelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ is negligible in κ .

B Security Properties of Group Signature Schemes

In this paper, we follow the definitions presented in [4, 7, 23, 22] which include the following properties: **full-traceability**, **full-anonymity**, **anonymity**, and **no-misattribution**. In order to meaningfully specify the security properties, we first consider the oracles an adversary is allowed to have access to.

- $\mathcal{O}_{\text{Setup}}(1^\kappa)$: activate \mathcal{GM} to execute GSIG.Setup on security parameter 1^κ .
- $\mathcal{O}_{\text{Join}}(U)$: result in a user U becoming a member of the group.
- $\mathcal{O}_{\text{Revoke}}(U)$: result in the membership of U being revoked.
- $\mathcal{O}_{\text{Update}}()$: result in the non-revoked members updating their system state information.
- $\mathcal{O}_{\text{Sign}}(M)$: activate a group member U to sign a message M by executing the **Sign** algorithm.
- $\mathcal{O}_{\text{Open}}(\sigma)$: activate the group manager \mathcal{GM} to execute the **Open** algorithm on signature σ .
- $\mathcal{O}_{\text{Corrupt}}(U)$: return U ’s private information.
- $\mathcal{O}_{\text{Corrupt}}(\mathcal{GM}, \perp)$: return \mathcal{GM} ’s private information used to admit group members (i.e., the secrets used to issue membership certificates).
- $\mathcal{O}_{\text{Corrupt}}(\mathcal{GM}, \top)$: return \mathcal{GM} ’s private information used to **Open** group signatures (i.e., the secrets used to revoke anonymity).

Remark. First, a user or (part of) a group manager may be corrupted before becoming part of a group signature scheme. This is captured by allowing an adversary to immediately corrupt an entity after its initialization such as being admitted as a group member. Second, we differentiate the case that an adversary corrupts a group manager’s capability for admitting members from the case that an adversary corrupts a group manager’s capability for opening group signatures. This reflects the good practice that one server is responsible for admitting members, and another is responsible for opening group signatures.

Full-traceability: any group signature can be traced back to its actual signer. Formally, consider the following game or experiment.

Experiment $\text{FullTrace}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$
 $(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}) \leftarrow \text{Setup}(1^\kappa)$
 $(M, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \top)}()$
 Return “1” if
 (1) $\text{Verify}(M, \sigma)$ returns TRUE
 (2) $\Delta = \{U' \in \mathbb{ID} \mid \exists \mathcal{O}_{\text{Corrupt}}(U')\}$
 (3) $U \leftarrow \text{Open}(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}, M, \sigma)$
 (4) $U \notin \Delta$

A group signature scheme fulfills **full-traceability** if $\forall \mathcal{A}$, $\Pr[\text{FullTrace}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$ returns “1”] is negligible in κ .

Full-anonymity: no adversary can identify the actual signer of a group signature, even if the actual signer’s secret has been compromised. Formally, consider the following game or experiment.

Experiment $\text{FullAnon}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$
 $(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}) \leftarrow \text{Setup}(1^\kappa)$
 $b \in_R \{0, 1\}$
 $(U_0 \in \mathbb{ID}, U_1 \in \mathbb{ID}, M, \text{stateinfo}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \perp)}(1^\kappa)$
 Return “1” if
 (1) $\sigma \leftarrow \text{Sign}(pk_{\mathcal{G}\mathcal{M}}, pk_{U_b}, sk_{U_b}, M)$
 (2) $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \perp)}(\text{stateinfo})$,
 (3) there is no $\mathcal{O}_{\text{Open}}(\sigma)$ query
 (4) $b' = b$

A group signature scheme fulfills **full-anonymity** if $|\Pr[\text{FullAnon}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$ returns “1”] $- 1/2|$ is negligible in κ .

Anonymity: no adversary can identify the actual signer of a group signature, as long as the actual signer’s secret has not been compromised. Formally, consider the following game or experiment.

Experiment $\text{Anon}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$
 $(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}) \leftarrow \text{Setup}(1^\kappa)$
 $b \in_R \{0, 1\}$
 $(U_0 \in \mathbb{ID}, U_1 \in \mathbb{ID}, M, \text{stateinfo}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \perp)}(1^\kappa)$
 Return “1” if
 (1) $\sigma \leftarrow \text{Sign}(pk_{\mathcal{G}\mathcal{M}}, pk_{U_b}, sk_{U_b}, M)$
 (2) $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \perp)}(\text{stateinfo})$,
 (3) there is no $\mathcal{O}_{\text{Open}}(\sigma)$ query
 (4) there is no $\mathcal{O}_{\text{Corrupt}}(U_0)$ or $\mathcal{O}_{\text{Corrupt}}(U_1)$ query
 (5) $b' = b$

A group signature scheme fulfills **anonymity** if $|\Pr[\text{Anon}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$ returns “1”] $- 1/2|$ is negligible in κ .

No-misattribution: a compromised group manager cannot misattribute a group signature to an honest group member. Formally, consider the following game or experiment.

Experiment $\text{NoMisAttri}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$
 $(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}) \leftarrow \text{Setup}(1^\kappa)$
 $(M, U \in \mathbb{ID}, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Join}}(\cdot), \mathcal{O}_{\text{Sign}}(\cdot, \cdot), \mathcal{O}_{\text{Open}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\cdot), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \perp), \mathcal{O}_{\text{Corrupt}}(\mathcal{G}\mathcal{M}, \top)}(1^\kappa)$
 Return “1” if
 (1) σ is returned by $\mathcal{O}_{\text{Sign}}(U, M)$
 (2) there is no $\mathcal{O}_{\text{Corrupt}}(U)$ query
 (3) $U \leftarrow \text{Open}(pk_{\mathcal{G}\mathcal{M}}, sk_{\mathcal{G}\mathcal{M}}, M, \sigma)$

A group signature scheme fulfills **no-misattribution** if $\Pr[\text{NoMisAttri}_{\text{GSIG}, \mathcal{A}}(1^\kappa)$ returns “1”] is negligible in κ .

C Security Properties of Centralized Key Distribution Schemes

We consider an adversary that has complete control over all the communications in the network. To simplify the definition, we assume that the group controller is never compromised.

We adopt the notion of **security** in the so-called active outside attack model introduced in [34], because it provides the desired security assurance. In this adversarial model, an adversary's interaction with principals in the network is modeled by allowing it to have access to (some of) the following oracles:

- $\mathcal{O}_{Send}(U, t, M, action)$: Send a message M to $U' \in \{\mathcal{GC}\} \cup \mathbb{ID}$ at time $t \geq 0$, and output its reply, where $action \in \{\text{Setup, Join, Leave, Rekey}\}$ meaning that U' will execute according to the corresponding protocol, and M specifies the needed information for executing the protocol. Of course, the query of type **Setup** is only made at time $t = 0$.
- $\mathcal{O}_{Reveal}(U, t)$: Output group key held by $U \in \Delta^{(t)}$ at time t , namely $k_U^{(t)}$.
- $\mathcal{O}_{Corrupt}(U, t)$: Output the keys held by $U \in \Delta^{(t)}$ at time t , namely $K_U^{(t)}$.
- $\mathcal{O}_{Test}(U, t)$: This oracle may be queried only once, at any time during the adversary's execution. A random bit b is generated: if $b = 1$ the adversary is given $k_U^{(t)}$ where $U \in \Delta^{(t)}$, and if $b = 0$ the adversary is given a random key of length $|k_U^{(t)}|$.

Definition 4. (security) Intuitively, it means that an adversary learns no information about a group key if, with respect to the rekeying event of interest there is no corrupt legitimate member (this implicitly implies that all the previously corrupt members have been revoked). Formally, consider the following event **Succ**:

- (1) The adversary queries the $\mathcal{O}_{Test}(U, t)$ oracle with $\text{acc}_U^{(t)} = \text{TRUE}$, and correctly guesses the bit b used by the $\mathcal{O}_{Test}(U, t)$ oracle in answering this query.
- (2) There is no $\mathcal{O}_{Reveal}(V, t)$ query for any $V \in \Delta^{(t)}$. (Otherwise, the group key is trivially compromised.)
- (3) If there was any $\mathcal{O}_{Corrupt}(V, t_1)$ query where $t_1 < t$, then there must have been an $\mathcal{O}_{Send}(\mathcal{GC}, t_2, V, \text{Leave})$ query where $t_1 < t_2 \leq t$. This captures that the corrupt members must have been revoked before the rekeying message at time t .
- (4) There is no $\mathcal{O}_{Corrupt}(V, t_3)$ query for $t_3 = t$ and $V \in \Delta^{(t_3)}$. (This does not rule out that there could be some $\mathcal{O}_{Corrupt}(V, t_3)$ query for $t_3 > t$.)

The advantage of the adversary \mathcal{A} in attacking the group communication scheme is defined as $\text{Adv}_{\mathcal{A}}(\kappa) = |2 \cdot \Pr[\text{Succ}] - 1|$, where $\Pr[\text{Succ}]$ is the probability that the event **Succ** occurs, and the probability is taken over the coins used by \mathcal{GC} and by \mathcal{A} . We say a scheme is **secure** if for all probabilistic polynomial-time adversary \mathcal{A} it holds that $\text{Adv}_{\mathcal{A}}(\kappa)$ is negligible in κ .

Instantiations: We note that the authenticated private channels can be implemented using standard cryptographic techniques – this has been achieved in many specific schemes such as [33] – except for the following: an adversary observing the communication channels may still be able to identify the membership of the non-compromised members. This can be resolved by letting the group controller publish information over a broadcast channel, or post it on a bulletin board to which all the group members can have anonymous access (e.g., via the standard MIX channels [15]). We stress that any centralized group key distribution scheme satisfying the above functionality and security requirements, such as [33, 32] with the enhancement presented in [34], can be integrated into our secret handshake framework.

There is one other issue that needs to be addressed: we may need to ensure that the system update information published by the group controller does not allow a passive adversary to discover the group identity, or even group dynamics. Although this may not be a real concern for many secret handshake applications (since a compromised group member has to know the identity of the group controller) we note that enhancement techniques suggested in [31] can be utilized to conceal the group dynamics of many popular centralized key distribution schemes, including [33, 32].

We notice that the above discussions also apply to the stateless group communication schemes (e.g., [26]); we refer to [34] for details.

D Security Properties of Distributed Group Key Agreement Schemes

We allow an adversary to have complete control over all the communications in the network. An adversary's interaction with the principals in the network (more specifically, with the various instances) is modeled by allowing it to have access to the following oracles:

- $\mathcal{O}_{Send}(U, i, M)$: Send message M to instance Π_U^i , and outputs the reply generated by this instance. Moreover, we even allow the adversary to prompt the unused instances Π_U^i to initiate the protocol with partners U_2, \dots, U_m by calling $\mathcal{O}_{Send}(U, i, \langle U_2, \dots, U_m \rangle)$.
- $\mathcal{O}_{Execute}(U_1, \dots, U_m)$: Execute the protocol between unused instances of players $U_1, \dots, U_m \in \mathbb{ID}$ and outputs the transcript of the execution. The number of participants and their identities are chosen by the adversary.
- $\mathcal{O}_{Reveal}(U, i)$: Output session key sk_U^i , where $U \in \mathbb{ID}$.
- $\mathcal{O}_{Corrupt}(U)$: Output the long-term secret LLK_U of principal $U \in \mathbb{ID}$.
- $\mathcal{O}_{Test}(U, i)$: This query is only allowed once, at any time during the adversary's execution. A random bit b is generated; if $b = 1$ the adversary is given sk_U^i , and if $b = 0$ the adversary is given a random session key.

Correctness. We require that for all U, U', i, j such that:

$$\text{sid}_U^i = \text{sid}_{U'}^i, \text{pid}_U^i = \text{pid}_{U'}^j, \text{and } \text{acc}_U^i = \text{acc}_{U'}^j = \text{TRUE}$$

to be the case that: $\text{sk}_U^i = \text{sk}_{U'}^j \neq \text{null}$.

Security. Intuitively, an adversary that does not compromise any principals during the execution does not learn any information about the new group session key. Formally, consider the following event **Succ**:

- (1) The adversary queries $\mathcal{O}_{Test}(U, i)$ on instance Π_U^i for which $\text{acc}_U^i = \text{TRUE}$ and correctly guesses the bit b used by the oracle in answering this query.
- (2) For any $\Pi_{U'}^{i'}$ such that $\text{pid}_U^i = \text{pid}_{U'}^{i'}$ and $\text{sid}_U^i = \text{sid}_{U'}^{i'}$, there is no $\mathcal{O}_{Reveal}(U, i)$ or $\mathcal{O}_{Reveal}(U', i')$ query.
- (3) No query $\mathcal{O}_{Corrupt}(U')$ (with $U' \in \text{pid}_U^i$) was asked before a query of the form $\mathcal{O}_{Send}(U', i', *)$. Note that, while (3) is not necessary in general, it is required in some specific schemes.

The advantage of the adversary \mathcal{A} attacking the scheme DGKA is defined as: $\text{Adv}_{\text{DGKA}, \mathcal{A}}(1^\kappa) = |2 \cdot \Pr[\text{Succ}] - 1|$, where $\Pr[\text{Succ}]$ is the probability that the event **Succ** occurs. We say a distributed key agreement scheme DGKA is *secure* if, for any probabilistic polynomial-time adversary \mathcal{A} , $\text{Adv}_{\text{DGKA}, \mathcal{A}}(1^\kappa)$ is negligible in κ .

Instantiations: Any distributed group key agreement scheme satisfying the above requirements can be integrated into our framework. Popular schemes include [11, 30] and [10]. In particular, the scheme by Burmester and Desmedt [11] and its later variant [21] are particularly efficient – each participant needs to compute a constant number of modular exponentiations.

E Proof Sketch for Theorem 1

Proof. (SKETCH) **Correctness.** When all m parties belong to the same group, the handshake always succeeds; otherwise, the handshake succeeds only with a negligible probability (see **resistance to impersonation** below).

Resistance to impersonation. We observe that

$$\begin{aligned} \Pr[\text{RIA}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"}] &= \Pr[\mathcal{A} \text{ provides a valid MAC tag} \wedge \mathcal{A} \text{ provides a valid group signature}] \\ &\leq \Pr[\mathcal{A} \text{ provides a valid MAC tag}] \\ &\leq \Pr[\mathcal{A} \text{ knows the MAC key}] + \\ &\quad \Pr[\mathcal{A} \text{ provides a valid MAC tag} | \mathcal{A} \text{ doesn't know the MAC key}]. \end{aligned}$$

Suppose \mathcal{A} breaks the **resistance to impersonation** property. This means that $\Pr[\text{SHS.Handshake}(\Delta)$ returns “1”] is non-negligible, and thus that either $\Pr[\mathcal{A}$ knows the MAC key] or $\Pr[\mathcal{A}$ provides a valid MAC tag | \mathcal{A} doesn't know the MAC key] is non-negligible.

If $\Pr[\mathcal{A}$ knows the MAC key] is non-negligible, then we can construct a polynomial-time algorithm \mathcal{B} to break the CGKD scheme. Algorithm \mathcal{B} operates as follows: It establish a simulated SHS environment as in the real-life system, except that the challenge CGKD environment is associated with a group G that is chosen uniformly at random, and \mathcal{B} maintains the DGKA and GSIG with respect to G by itself. The simulated SHS environment operates exactly the same as in the real life system, except the following: Whenever \mathcal{A} makes a query to the SHS environment with respect to group G , if the query cannot be answered by \mathcal{B} based on its information regarding the DGKA and GSIG with respect to G , \mathcal{B} makes the same queries to the challenge CGKD environment. Clearly, the simulated SHS environment is perfectly the same as in a real-life system. Since \mathcal{A} has a non-negligible probability in learning a MAC key corresponding to group G , so is \mathcal{B} because \mathcal{B} also participates in DGKA.GroupKeyAgreement. The fact that \mathcal{A} does not corrupt any member when the MAC key is valid implies that \mathcal{B} does not corrupt any member. Assuming the existence of polynomially many groups, we conclude that \mathcal{B} breaks the challenge CGKD environment with non-negligible probability.

If $\Pr[\mathcal{A}$ provides a valid MAC tag | \mathcal{A} doesn't know the MAC key] is non-negligible, then we can construct a polynomial-time algorithm \mathcal{B} to break the MAC scheme. Algorithm \mathcal{B} operates as follows: It establish a simulated SHS environment as in the real-life system, except that the challenge MAC scheme and an oracle scheme CGKD are associated with a group G that is chosen uniformly at random, and \mathcal{B} maintains the DGKA and GSIG with respect to G by itself. The simulated SHS environment operates exactly the same as in the real life system, except the following: Whenever \mathcal{A} makes a query to the SHS environment with respect to group G , if the query cannot be answered by \mathcal{B} based on its information regarding the DGKA and GSIG with respect to G , \mathcal{B} makes the same queries to the challenge MAC scheme or the oracle CGKD environment. Clearly, the simulated SHS environment is perfectly the same as in a real-life system. Since \mathcal{A} has a non-negligible probability in forging, perhaps after seeing valid tags via \mathcal{B} by having oracle access to the MAC scheme, a MAC tag corresponding to a key that is unknown to \mathcal{A} (and thus unknown to \mathcal{B}), the forgery is also a successful attack by \mathcal{B} . Conditioned on the fact that there are polynomial many groups and there are polynomial many instances of SHS.Handshake, we conclude that \mathcal{B} breaks the challenge MAC scheme with a non-negligible probability.

Resistance to detection. We notice that we already proved that $\Pr[\text{RDA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ returns “1” | bit = 0] is negligible. Therefore, if $\text{Adv}_{\text{RDA}_{\text{SHS}, \mathcal{A}}}$ is non-negligible, then $\Pr[\text{RDA}_{\text{SHS}, \mathcal{A}}(1^\kappa)$ returns “1” | bit = 1] is non-negligible. If this probability is non-negligible,

then we can construct a simulator \mathcal{B} to break the MAC scheme.⁹ Algorithm \mathcal{B} operates as follows: It establish a simulated SHS environment as in the real-life system. Note that \mathcal{B} has access to a challenge MAC environment. The simulated SHS environment operates exactly the same as in the real life system, except the following: When \mathcal{A} incurs the $\text{RDA}_{\text{SHS},\mathcal{A}}$, \mathcal{B} chooses a party $U' \in \Delta'$ uniformly at random and provides its MAC tag via oracle query to the challenge MAC environment; for any other $U \in \Delta' \setminus \{U'\}$, \mathcal{B} also feeds with MAC tags via oracle query to the challenge MAC environment. If \mathcal{A} outputs a valid MAC tag with respect to the MAC tag of U' , then this is also a successful forgery of \mathcal{B} against the challenge MAC scheme; otherwise, \mathcal{B} fails. Conditioned on the fact that $|\Delta'|$ is polynomially bounded, \mathcal{B} breaks the challenge MAC scheme with a non-negligible probability.

Full-unlinkability. Suppose an adversary can link two handshake sessions involving a same honest party. Without loss of generality, we assume that both sessions involve the same malicious party. There are two cases:

- **CASE 1:** Both sessions are successful handshakes. First, recall that no information in the transcript, except θ_i , contains any information specific to the identity of the honest party. If the adversary can assert that the same party is involved in the two instances, then we claim that **full-anonymity** of GSIG is violated. This is so because the adversary’s decision can only be based upon the two group signatures produced by the same party. Specifically, we can construct a polynomial-time algorithm \mathcal{B} , which simulates the SHS environment as in the real life, except that \mathcal{B} only has oracle access to a challenge group signature scheme GSIG. Whenever \mathcal{A} makes a query to the SHS environment, if the query cannot be answered by \mathcal{B} based on its information regarding the CGKD and DGKA, \mathcal{B} makes the same queries to the challenge GSIG scheme. Clearly, the simulated SHS environment is perfectly the same as in a real-life system. Finally, when \mathcal{A} outputs the two handshake sessions involving the same person, \mathcal{B} decrypt the corresponding θ ’s and output the pair of group signatures.
- **CASE 2:** At least one of them is an unsuccessful handshake. This is impossible since, in Phase III of the unsuccessful session, the honest party’s output is wholly based on random simulations (i.e., independent of the honest party’s secrets).

Indistinguishability to eavesdroppers. The only difference in the adversary’s view is whether $\{(\theta_i, \delta_i)\}_{1 \leq i \leq m}$ are “real” or random, where δ_i is the symmetric encryption of group signature σ_i under the key $k'_i = k_i^* \oplus k_i$ and θ_i is the encryption of k'_i under $\mathcal{G}\mathcal{A}$ ’s public key pk_T . (Note that this is indeed the so-called hybrid encryption). We observe that

$$\begin{aligned} & \Pr[\text{INDeav}_{\text{SHS},\mathcal{A}}(1^\kappa) \text{ returns “1”}] \\ &= \Pr[\mathcal{A} \text{ knows } k'_i \vee \mathcal{A} \text{ can distinguish encryption of group signature}] \\ &\leq \Pr[\mathcal{A} \text{ knows } k'_i] + \Pr[\mathcal{A} \text{ can distinguish encryption of group signature}] \end{aligned}$$

We claim that $\Pr[\mathcal{A} \text{ knows } k'_i]$ is negligible; otherwise, such an adversary can immediately break the above proved **resistance to detection**. If $\Pr[\text{AdvINDeav}_{\text{SHS},\mathcal{A}}(1^\kappa) \text{ returns “1”}]$ is negligible, so is $\Pr[\mathcal{A} \text{ can distinguish encryption of group signature}]$, which is the probability that \mathcal{A} can tell the encryption of a valid group signature (\mathcal{A} knows the group public key in the worst case) from the encryption of a random string, even if \mathcal{A} does not have any information about sk_T or k'_i . This immediately reduce to the security of the hybrid encryption scheme, namely that we can construct another polynomial-time algorithm to break either a challenge public key cryptosystem (with respect to T) or a challenge symmetric key cryptosystem (with respect to k'_i).

Traceability. As long as the transcript includes the encryption of a valid session key k' under the $\mathcal{G}\mathcal{A}$ ’s public key pk_T , $\mathcal{G}\mathcal{A}$ can always recover k' . As a result, given any pair (θ_i, δ_i) , $\mathcal{G}\mathcal{A}$ is also able to recover the encrypted group signature, and therefore the identify of the actual signer. If there is a valid group signature that cannot be opened, then **full-traceability** with respect to GSIG is broken.

Specifically, we construct an algorithm \mathcal{B} that simulates the SHS environment by having oracle access to a challenge group signature scheme GSIG. The execution of SHS is exactly the same as in a real-life system, except that whenever \mathcal{A} presents a query against the GSIG, the query is forwarded by \mathcal{B} to the challenge GSIG environment. Clearly, $\text{AdvTraceUser}_{\text{SHS},\mathcal{A}}$ is non-negligible means that \mathcal{A} can provide a group signature in the $\text{TraceUser}_{\text{SHS},\mathcal{A}}$ experiment such that the signer is an innocent legitimate member. This is \mathcal{B} ’s successful attack against the challenge GSIG environment.

No-misattribution. If an adversary can frame an honest member (link to a handshake instance) we claim that **no-misattribution** of group signature scheme is violated. Specifically, we construct an algorithm \mathcal{B} which simulates a SHS environment by having oracle access to a challenge GSIG scheme. The simulated SHS environment executes exactly as in a real system, except that an oracle query provided by \mathcal{A} (which cannot be directly answered by \mathcal{B}) is answered after \mathcal{B} forwards the same query to the challenge GSIG environment. In particular, the group manager in the challenge GSIG environment can be corrupted. Clearly, if $\text{AdvMisattribution}_{\text{SHS},\mathcal{A}}$ is non-negligible, then \mathcal{B} violates **no-misattribution** of the challenge GSIG environment with the same probability.

⁹ We notice that here it is reinforced why we need the newly introduced strong security property of [34]; otherwise, an adversary could obtain the past group key and thus tell whether a handshake it participated involves some legitimate members.

F Proof Sketch for Theorem 2

Proof. (SKETCH) The proof is the same as in the proof of Theorem 1, except for the proof of property **unlinkability**. So we only focus on it.

Unlinkability. This property needs to be proved from scratch because the underlying group signature scheme only possesses **anonymity** rather than **full-anonymity**. Suppose an adversary can link two handshake sessions involving a same honest party. Without loss of generality, we assume that both sessions involve the same malicious party. There are two cases:

- **CASE 1:** Both sessions are successful handshakes. First, recall that no information in the transcript, except θ_i , contains any information specific to the identity of the honest party. If the adversary can assert that the same party is involved in the two instances, then we claim that **anonymity** of **GSIG** is violated. This is so because the adversary’s decision can only be based upon the two group signatures produced by the same party. Specifically, we can construct a polynomial-time algorithm \mathcal{B} , which simulates the **SHS** environment as in the real life, except that \mathcal{B} only has oracle access to a challenge group signature scheme **GSIG**. Whenever \mathcal{A} makes a query to the **SHS** environment, if the query cannot be answered by \mathcal{B} based on its information regarding the **CGKD** and **DGKA**, \mathcal{B} makes the same queries to the challenge **GSIG** scheme. Clearly, the simulated **SHS** environment is perfectly the same as in a real-life system. Finally, when \mathcal{A} outputs the two handshake sessions involving the same person, \mathcal{B} decrypt the corresponding θ ’s and output the pair of group signatures.
- **CASE 2:** At least one of them is an unsuccessful handshake. This is impossible since, in **Phase III** of the unsuccessful session, the honest party’s output is wholly based on random simulations (i.e., independent of the honest party’s secrets).

G Proof Sketch for Theorem 3

Proof. (SKETCH) The proof is the same as in the proof of Theorem 2, except that we further achieve **self-distinction**. So we only focus on it.

Self-distinction. We observe that T_7 is the base common to all the participating parties and $T_6 = T_7^{x'}$, where x' is a member’s secret. Suppose $\text{AdvSelfDist}_{\text{SHS}, \mathcal{A}}$ is non-negligible, then \mathcal{A} must be able to provide two group signatures with different x' ’s. Without loss of generality, suppose \mathcal{A} knows (A, e, x, x') such that $A^e = a_0 a^x b^{x'}$. Denote by X the event that \mathcal{A} holds (A, e, x, x') and $(A_{\mathcal{A}}, e, x_{\mathcal{A}}, x'_{\mathcal{A}})$ such that $A^e = a_0 a^x b^{x'} = (A_{\mathcal{A}})^e = a_0 a^{x_{\mathcal{A}}} b^{x'_{\mathcal{A}}}$. Note that

$$\begin{aligned} & \Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"}] \\ &= \Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | X] \cdot \Pr[X] + \Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \neg X] \cdot \Pr[\neg X] \\ &\leq \Pr[X] + \Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \neg X] \end{aligned}$$

Since $\Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"}]$ is non-negligible, so is $\Pr[X]$ or $\Pr[\text{SelfDist}_{\text{SHS}, \mathcal{A}}(1^\kappa) \text{ returns "1"} | \neg X]$. We claim that the **full-traceability** of the **GSIG** scheme is broken.

Specifically, we construct an algorithm \mathcal{B} , which simulates a **SHS** environment by having oracle access to a challenge **GSIG** scheme. The simulated **SHS** environment execute exactly the same as in a real-life system, except that an oracle query provided by \mathcal{A} , which cannot be directly answered by \mathcal{B} , is answered after \mathcal{B} forwards the same query to the challenge **GSIG** environment. Note that the group manager in the challenge **GSIG** environment is never corrupted. Therefore, if \mathcal{A} can provide two group signatures with $A^e = a_0 a^x b^{x'} = (A_{\mathcal{A}})^e = a_0 a^{x_{\mathcal{A}}} b^{x'_{\mathcal{A}}}$, \mathcal{B} breaks the **full-traceability** of the challenge **GSIG** environment.

H Overview of the Kiayias/Yung Group Signature Scheme

For completeness, we now briefly overview the signature scheme of [22], which is an extension of the [1] group signature scheme. We summarize the parameters as:

- Two primes p', q' with $p = 2p' + 1$, $q = 2q' + 1$ also primes. The modulus is set to $n = pq$.
- The group manager selects $a, a_0, b, g, h \in_R QR(n)$.
- The group manager secret key is set to $p, q, \theta \in_R \mathbb{Z}_{p'q'}$.
- The system public key is n, a, a_0, b, y, g, h , where $y = g^\theta \pmod n$.
- A member’s private key is (A, e, x, x') such that $A^e = a_0 a^x b^{x'} \pmod n$, where e is an appropriate prime number, x is an appropriate random number known also to the member, and x' is an appropriate random number known only to the member.

In order to sign a message, a group member computes:

$$T_1 = Ay^r, T_2 = g^r, T_3 = g^e h^r, T_4 = g^{xk}, T_5 = g^k, T_6 = g^{x'k'}, T_7 = g^{k'}$$

where r, k, k' are randomly selected from an appropriate interval. The signature is derived from the following proof of knowledge of appropriate x, x', e, r, h' :

$$T_2 = g^r, T_3 = g^e h^r, T_2^e = g^{h'}, T_5^x = T_4, T_7^{x'} = T_6, a_0 a^x b^{x'} y^{h'} = T_1^e.$$

Notice that (T_1, T_2) allows the group manager to open a group signature because of A , (T_4, T_5) allows one who knows x to trace the member's signatures, and (T_6, T_7) allows one to claim its signatures.