Taming Big Brother Ambitions: More Privacy for Secret Handshakes

Mark Manulis[1], Bertram Poettering[1], and Gene Tsudik[2]

[1] Cryptographic Protocols Group, TU Darmstadt & CASED, Germany
mark@manulis.eu, bertram.poettering@cased.de
[2] Computer Science Department, UC Irvine, USA
gts@ics.uci.edu

Abstract. In Secret Handshakes (SH) and Affiliation-Hiding Authenticated Key Exchange (AH-AKE) schemes, users become group members by registering with Group Authorities (GAs) and obtaining membership credentials. Group members then use their membership credentials to privately authenticate each other and communicate securely. The distinguishing privacy property of SH and AH-AKE is that parties learn each other's groups affiliations and compute common session keys only if their groups match. Current SH and AH-AKE schemes consider GAs to be fully trusted, especially, with regard to (i) security of the registration phase (no phantom members), (ii) secrecy of established session keys, and (iii) privacy. The impact of possible "big brother" ambitions of malicious GAs has not been investigated so far. In this paper, we discuss implications on group members' privacy and security of their communication in the presence of possible GA corruptions. We demonstrate problems arising from relaxed GA trust assumptions and propose an efficient — yet provably secure — AH-AKE protocol with enhanced privacy properties.

1 Introduction

Affiliation-Hiding Key Exchange In the public-key setting, traditional Authenticated Key Exchange (AKE) protocols offer secure key establishment while usually revealing the identities and certificates of participants. Affiliation-Hiding AKE (AH-AKE) schemes [11, 12] that combine Secret Handshakes (SH) [3, 8, 21, 20, 19, 2, 13, 14] with secure key establishment offer stronger privacy guarantees. In both SH and AH-AKE, users are members of groups administrated by Group Authorities (GAs). Prior to participation, users register with the GA to obtain their membership credentials. The goal of SH and AH-AKE is to ensure private matching (e.g. exact, or dynamic matching [2]) between the affiliations (groups) of participants. Privacy stems from the requirement to hide these affiliations from outsiders or members of non-matching groups. AH-AKE protocols provide stronger privacy than Secret Handshakes, since they guarantee the affiliation-hiding property, even if established session keys are disclosed. Additionally, AH-AKE protocols provide traditional AKE-security goals [7] for the established keys. SH and AH-AKE schemes come in two flavors: linkable and unlinkable.

Linkable protocols [3, 8, 11, 12], which are useful if participants wish to be recognized across different sessions, employ re-usable pseudonyms that members obtain during the registration process with the GA. In contrast, unlinkable protocols [21, 13, 2, 14] aim at preventing any correlation among multiple sessions of the same participant.

The GA Role We assume that each GA manages one group. It is responsible for the registration of new members and for any subsequent membership revocation [3, 8, 21, 12, 14]. While in linkable AH-AKE members can be efficiently revoked by black-listing their pseudonyms on public revocation lists, unlinkable AH-AKE supports revocation either by restricting the number of unlinkable sessions of users, e.g. [21], or by regularly updating unrevoked membership credentials, e.g. [13].

Current protocols assume that GAs are fully trusted. This becomes clear by inspecting the underlying security models [13, 12, 14] where GA corruptions are not among the adversary's options. We first discuss what exactly the GA is trusted with, and whether this trust is justifiable and/or offers space for meaningful relaxations.

Security of Registration/Revocation Among GA duties is the registration and revocation of group members. Clearly, if the GA misbehaves and introduces phantom members, then security of session keys computed by honest participants in sessions with phantom members can no longer represent a meaningful requirement. Therefore, GA must be trusted with regard to security of the registration (and revocation) in that it does not enroll phantom (or revoke honest) members. This is similar to the usual trust placed into the Certification Authority (CA) in public-key based AKE schemes.

However, we believe that possible GA misbehavior during registration of new (honest) members must be taken into account. Note that the registration process is the only step where GA interacts directly with users. Therefore, information obtained or issued by the GA during registration may be later misused to the detriment of members' privacy.

Security of Session Keys An AH-AKE instance between two honest members should result in a secure session key. For obvious reasons, it is desirable for this key to be kept secret from the GA. This requirement subsumes forward secrecy of session keys with respect to any future GA corruptions. Although this issue has not been formally addressed so far, we note that some recent results [11, 12] seem to satisfy this extended form of forward secrecy. Whereas, in many other protocols, e.g. [3, 8, 2], private secrets of the GA can be used to immediately recover session keys.

Privacy of Group Members The central privacy requirement of AH-AKE protocols is to hide the group membership (affiliation) of participants from outsiders. However, it is also meaningful to extend this requirement towards the GA. As

long as GA is trusted with the security of the registration, it makes sense for transcripts of sessions among honest members not to reveal their affiliations to the GA. This requirement is of particular importance for linkable schemes where participants communicate via pseudonyms. Current linkable schemes, such as [3, 8, 20, 12], do not provide this stronger privacy notion, since the GA learns (or even specifies) the pseudonyms of its members during the registration process.

Finally, there is an even more significant threat to privacy of group members, that stems from the fact that, during the registration process, GA learns users' real identities. In particular, we believe that handshake sessions involving honest members should not reveal any information about their real identities to the GA (even though the GA knows the real identities of all group members). In other words, users should remain *untraceable* throughout their communication sessions.

Untraceability is a new privacy requirement that does not appear in current AH-AKE security models; in fact, all current linkable protocols that we are aware of do not provide it. We observe that untraceability is an individual goal of members, while affiliation-hiding is a goal shared by all members of the same group. Therefore, untraceability is desirable even if the GA deliberately engages in sessions with genuine group members.

The above discussion shows that unconditional trust in GA by group members, as imposed by current AH-AKE security models, can be problematic and needs to be re-examined. In particular, mitigation of potential GA misbehavior (aimed to undermine privacy of group members and security of their communication) is an important goal which motivates our present work.

Contributions and Organization Our work makes two contributions. First, in Section 2, we present three intuitive security goals for AH-AKE schemes (key secrecy, affiliation-hiding and user untraceability) that explicitly consider GA misbehavior. These goals can be viewed as strengthening those of the recent model of [12] where GA corruptions are not considered. We summarize two current protocols [12, 8] and discuss why they are not strong enough to achieve these new goals. Based on this observation, as our second contribution, in Section 3 we propose a new AH-AKE scheme that operates in the Discrete Logarithm (DL) setting. Although some central design ideas are similar, the new protocol is fairly different from the one proposed by Castellucia, et al. [8]. In short, one novel factor is the decoupling of the registration phase, where pseudonyms are generated, from later protocol sessions by adopting blinding techniques for Schnorr signatures [18, 17]. We show that an anonymized registration process is in fact necessary to preserve affiliation-hiding and untraceability against GAs. We also show that the latter can be achieved unconditionally. Efficiency and key security of our technique stem from a key establishment process where session keys are derived similarly to the Unified Model [4], thus achieving forward secrecy (not provided by [8]). Then, in Section 4, we prove security and privacy of our protocol in the random oracle model (ROM) using the computational variant of the Oracle Diffie-Hellman assumption from [1].

Related Work Linkable Secret Handshake (LSH) schemes [3, 8, 20] provide group members with credentials composed of a pseudonym and additional secrets. These schemes have been designed with authentication in mind, and, although some of them offer session key establishment, no formal security treatment of the latter has been provided. Aforementioned schemes provide efficient revocation using certificate/pseudonym revocation lists. An extension of LSH to Linkable AH-AKE (LAH-AKE) schemes has been formally modeled and analyzed in [12]. The scheme in [12] works in the safe RSA setting and offers forward secrecy as well as revocation, under the trusted GA assumption.

In unlinkable Secret Handshakes [2, 13, 19, 14] credentials are reused while still precluding the correlation of multiple sessions involving the same participant. The challenging part is the process of revocation of protocol participants, which is completely disregarded in [2], handled via synchronization of revocation epochs in [13], and addressed in [19] with group signatures and broadcast encryption. Jarecki and Liu [14] recently constructed a scheme that supports more efficient revocation and unlinkable reusable credentials using group signature-related techniques. We remark that unlinkability generally can be obtained from linkable protocols by using one-time pseudonyms; however, this is clearly impractical.

There are also a couple of Linkable Group Secret Handshake schemes [10, 11] that extend the security model from two-party to multi-party authentication and key establishment scenarios. The approach in [10] uses credentials that employ Schnorr signatures issued by the GA (this is, in some sense, related to [8]), whereas, the scheme in [11] applies similar ideas to the RSA setting. Both approaches achieve session group key establishment based on a variant of the well-known Burmester-Desmedt [5] technique.

The first result on privacy protection against misbehaving GAs is due to Kawai, et al. [15]. It deviates from the traditional setting by splitting the GA role among the issue authority (responsible for registration and certificate issuance) and the tracing authority (responsible for tracing users based on their handshake transcripts). Since we treat the GA as a single instance, the setting of our work is more consistent with earlier results.

2 Malicious GAs: Impact and Challenges

After describing LAH-AKE syntax, we illustrate challenges stemming from malicious GAs using, as our running example, a concrete LAH-AKE scheme from [12]. We also highlight techniques necessary to protect against malicious GAs that are later used in our own construction.

2.1 SH and AH-AKE

The main syntactical difference between AH-AKE and SH schemes is the session key computation during protocol execution. Although many SH schemes provide participants with a session key, doing so is not mandatory for the purpose of pure authentication. An LAH-AKE scheme includes four components:

$\mathsf{CreateGroup}(1^\kappa)$ a probabilistic algorithm that sets up a new group G. It is executed by the corresponding GA. On input of the security parameter 1^κ, it generates a public/private group key-pair $(G.\mathsf{pk}, G.\mathsf{sk})$, initializes the group's pseudonym revocation list $G.\mathsf{prl}$ to $\emptyset$ and outputs public group parameters $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$ along with the private key $G.\mathsf{sk}$.

$\mathsf{AddUser}(U, G)$ a protocol executed between a prospective group member U and the GA of G. The algorithm on U's side is denoted $\mathsf{AddUserU}(U, G.\mathsf{par})$, and on GA's side by $\mathsf{AddUserG}(U, G.\mathsf{sk})$. Let π be a session of AddUserU or AddUserG. The *state* of π is represented with a variable $\pi.\mathsf{state}$ and can take running or accepted values. Initially $\pi.\mathsf{state} = \mathsf{running}$. Once AddUserU session π reaches $\pi.\mathsf{state} = \mathsf{accepted}$ its variable $\pi.\mathsf{result}$ contains a pair $(\mathsf{id}, \mathsf{id.cred})$ where id is a *pseudonym* and id.cred is a *membership credential* enabling U to authenticate as id in group G in future Handshake sessions. A user can have several registered pseudonyms in the same group.

$\mathsf{Handshake}(\mathsf{params}_1, \mathsf{params}_2)$ a protocol (handshake) executed between two users, U_1 and U_2, on inputs $\mathsf{params}_i = ((\mathsf{id}_i, \mathsf{id}_i.\mathsf{cred}), G_i.\mathsf{par}, r_i), i \in \{1, 2\}$, with $G_i.\mathsf{par} = (G_i.\mathsf{pk}, G_i.\mathsf{prl})$, $r_1 = \mathsf{init}$ and $r_2 = \mathsf{resp}$. Each U_i executes its own part $\mathsf{Handshake}'(\mathsf{param}_i)$. Note that id_i is the pseudonym previously registered to group G_i using the AddUser algorithm. The protocol verifies that both users are members of the same group (i.e. $G_1 = G_2$) and possess valid membership credentials. If so, the protocol accepts with an established shared session key. Otherwise, it rejects. Users keep track of the state of created Handshake protocols π through session variables that are initialized as follows: $\pi.\mathsf{state} \leftarrow \mathsf{running}$, $\pi.\mathsf{key} \leftarrow \perp$, $\pi.\mathsf{id} \leftarrow \mathsf{id}$ (where id is the own pseudonym) and $\pi.\mathsf{partner} \leftarrow \perp$. At some point, the protocol completes and $\pi.\mathsf{state}$ is updated to either rejected or accepted. In the latter case, $\pi.\mathsf{key}$ is set to the established session key (of length κ) and the pseudonym of the handshake partner is assigned to $\pi.\mathsf{partner}$. State accepted cannot be reached if the protocol partner is revoked ($\pi.\mathsf{partner} \in G.\mathsf{prl}$).

$\mathsf{Revoke}(G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id})$ a revocation algorithm executed by the GA of G. It outputs the updated pseudonym revocation list $G.\mathsf{prl} \leftarrow G.\mathsf{prl} \cup \{\mathsf{id}\}$.

Definition 1 (Correctness). *Suppose that two users, U_1 and U_2, register as members of groups G_1 and G_2, and obtain their credentials* $(\mathsf{id}_1, \mathsf{id}_1.\mathsf{cred})$ *and* $(\mathsf{id}_2, \mathsf{id}_2.\mathsf{cred})$*, respectively, via corresponding* AddUser *executions. Further suppose that U_1 and U_2 participate in a* Handshake *protocol and let π_1 and π_2 denote their corresponding sessions. The LAH-AKE scheme is called* correct *if (a) π_1 and π_2 complete in the same state:* accepted *iff $G_1 = G_2$ and $\mathsf{id}_1 \notin G_2.\mathsf{prl}$ and $\mathsf{id}_2 \notin G_1.\mathsf{prl}$ and $r_1 \neq r_2$, and (b) if both sessions accept, then* $(\pi_1.\mathsf{key}, \pi_1.\mathsf{partner}, \pi_1.\mathsf{id}) = (\pi_2.\mathsf{key}, \pi_2.\mathsf{id}, \pi_2.\mathsf{partner})$.

2.2 Impact of GA Corruptions

LAH-AKE with Honest GAs One state-of-the-art LAH-AKE scheme is due to Jarecki, et al. [12]. It is very efficient and offers a number of valuable security

properties. In particular, it satisfies the following standard requirements, which we state here informally.

Authenticated Key Exchange (AKE) Security with Forward Secrecy. It should be infeasible for an active PPT adversary to distinguish the session key computed in some test session from a random key with a probability non-negligibly exceeding $\frac{1}{2}$. AKE-security has been modeled in [12] following the general approach for key exchange protocols (e.g. [7]) via an indistinguishability game, that precludes all "trivial" attacks via which the adversary could obtain the key computed in the test session. The (sub)requirement of forward secrecy is typically modeled by allowing user corruptions, while preventing active participation of the adversary in the test session.

Linkable Affiliation-Hiding (LAH). It should be infeasible for an active PPT adversary to decide the group membership of an uncorrupted user from its handshake sessions or from knowledge of computed session keys. This requirement has been modeled in [12] via the simulation approach where the simulator executes handshake sessions without knowing the affiliation (and secret membership credentials) of participants.

We now briefly overview the LAH-AKE scheme from [12]. During setup, the GA creates public group parameters (n, g, e), where n is a safe RSA modulus of length $2\kappa''$, i.e., an RSA modulus $n = pq$ where p, q are safe κ'' bit primes, and $e \in \mathbb{Z}_{\varphi(n)}$ is an RSA exponent satisfying $\gcd(e, \varphi(n)) = 1$. Element $g \in \mathbb{Z}_n^*$ is chosen such that $\mathbb{Z}_n^* = \langle -1 \rangle \times \langle g \rangle$ (and hence $\mathrm{ord}(g) \approx n/2$). In addition, for each group, a specific hash function $H_n : \{0,1\}^* \to \mathbb{Z}_n^*$ is specified. When a user registers with the group, the pseudonym id it obtains is just a random string in $\{0,1\}^\kappa$. The corresponding credential $\mathsf{id.cred} = \sigma_{\mathsf{id}}$ issued by the GA is the RSA signature on the full-domain hash of id: $\sigma_{\mathsf{id}} = H_n(\mathsf{id})^d \bmod n$ (where $d = e^{-1} \bmod \varphi(n)$). The handshake protocol is sketched in Figure 1. Note that H_1 is a hash function $\{0,1\}^* \to \{0,1\}^\kappa$, and pad is a probabilistic function that maps its first argument θ' to a random element θ within a certain interval such that $\theta \equiv \theta' \pmod n$. This padding function is necessary to hide the RSA modulus sent in protocol messages. Correctness follows from $r_A = g^{2ex_Ax_B} = r_B$ which holds iff both participants employ valid credentials and consistent group parameters (n, g, e).

This protocol satisfies AKE- and LAH-security in the appropriate formal model, assuming random oracles and the hardness of the RSA problem with safe moduli. In our context, it is more important that the [12] model does not allow the adversary to corrupt relevant GAs. Our goal is to illustrate the impact of corrupted GAs on protocol sessions of honest users. We stress that our discussion does not mean that the original scheme is insecure. In our description, we distinguish between GAs malicious from the beginning (which is important if one considers that group parameters might be generated in some rogue way) and GAs that generate group parameters honestly but misbehave later.

Impact of GA corruptions on AKE-Security Suppose that the GA is malicious during setup. In particular, it might choose RSA modulus n or generator g

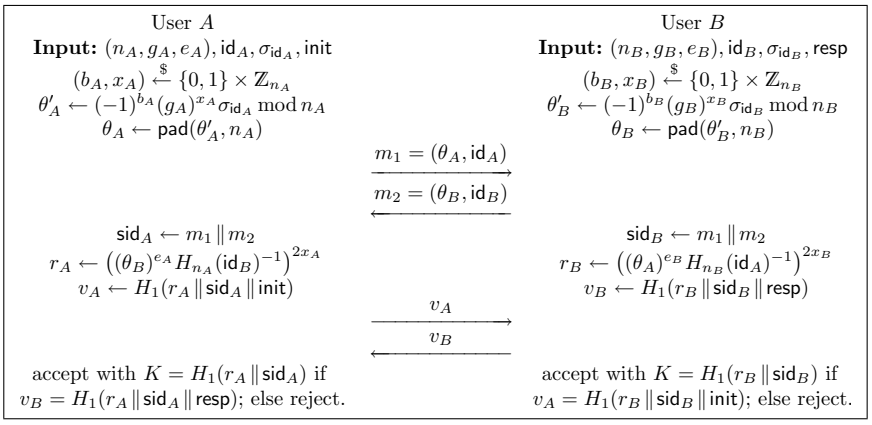

Fig. 1. RSA-based Handshake protocol from [12]

in a way that later facilitates computing session keys exchanged between honest members. For instance, if g is chosen to have small order (this is trivially feasible if n has more than two factors, or, if the factors are not safe primes), then the CDH-analog of computing r_A from θ'_A and θ'_B becomes tractable. Therefore, in general, public verifiability of group parameters is desirable for the registration of new members. In the RSA-based setting of [12], this is achievable using slight modifications of the zero-knowledge proofs from [6], as presented in [16]. However, this makes the registration process less efficient and, since our construction uses a DL-based setting, we refrain from investigating this idea further. As mentioned in Section 1, a malicious GA can always create phantom members and compute session keys exchanged between them and honest members. This is unavoidable since GA is typically trusted not to introduce new users to the system (similar to the CA in the classical PKI-setting). However, GA may be interested in learning the communication contents between two honest members and, in this sense, the protocol should ensure security of session keys in the presence of a passive GA that can corrupt group members. This requirement also implies forward secrecy with respect to GA corruptions. We observe that [12] provides this protection since the session key is derived from the ephemeral secret $g^{2ex_Ax_B}$ (assuming that public group parameters have been generated correctly). Nevertheless, forward secrecy against GA corruptions is an important security goal and should be considered in the design of AH-AKE schemes. For example, the SH scheme of [8], which we modify to obtain our solution, is not forward secure if the GA is malicious.

Impact of GA Corruptions on LAH-Security It seems impossible for affiliation of honest members to remain secret if these members are involved in handshake sessions with phantom members created by the malicious GA. In fact, this is similar to the case where the adversary corrupts a member and communicates with other users on behalf of that member. This case is typically excluded

from the definition of LAH-security. On the other hand, it is still desirable for sessions between two honest users not to reveal their affiliations to the GA, i.e., no information about the affiliation of a handshake participant should be derivable from a session transcript. Considering the registration process in [12], we observe that, since the GA learns the pseudonym id of each new member U, it can always decide whether some honest handshake participant is a member of its group by simply observing the communication and the transmission of pseudonyms in the clear. A possible remedy is to prevent the GA from learning pseudonyms of group members upon their registration. This can be achieved by *blinding* the registration process. One natural approach (in the context of [12]) it to combine blind RSA signatures coupled with a full-domain hash [9, 17]. However, the adversary could then register *any* pseudonym with any group. In particular, the adversary would be able to obtain membership credentials for some pseudonym id that is already in use by an honest group member, without explicitly corrupting any parties. This, in turn, would allow the adversary to mount (active) attacks on LAH-security and AKE-security of honest group members (since the adversary would be able to impersonate honest users without corrupting them or the GA). We stress that, in [12], this problem would arise not because of the blind registration process, but due to the specific construction of pseudonyms. In fact, our approach includes a blind registration process where no such problems occur, due to certain differences in pseudonym generation.

Impact of GA Corruptions on Traceability As noted in Section 1, consideration of malicious GAs motivates a new privacy requirement — member untraceability, which we define informally as follows:

> *Member Untraceability:* It is infeasible for an active PPT adversary to learn the real identity U of an honest group member from handshake sessions involving that member. Note that untraceability is an individual privacy goal motivated by the fact that the GA learns members' real identities during their registration processes.

In [12], untraceability is not provided for the same reason that handshake transcripts reveal the participants' affiliation to the GA: the link between a member's real identity U and its pseudonym id is known by the GA from the registration process. We believe that this is avoidable by adopting a blinded registration process. However, it requires us to further examine group membership revocation. In LAH-AKE schemes, revocation is attained by adding members' pseudonyms to the revocation list maintained by the GA. In schemes like [12] where the GA knows the link between U and id anyway, there is no difference between revoking members and revoking their pseudonyms. The consequence of untraceability is that revoking a specific member U is no longer possible (since neither the GA nor a protocol participant can link U to id). However, it is still possible for the GA to revoke pseudonyms. Since members participate in handshakes using pseudonyms and revocation can be seen as a tool to prevent misbehavior of participants, it is still sufficient for the GA to revoke "misbehaving" pseudonyms, effectively

preventing further participation of the member who "owns" them. This works only if the scheme ensures uniqueness of pseudonyms. However, if the scheme of [12] is amended with blind signatures in the registration process, uniqueness of pseudonyms can no longer be guaranteed. Although some workaround might be possible [16] (commensurate with lower efficiency), we do not investigate this direction, since our approach (which builds upon [8]) does not have such problems.

2.3 Challenges and Design Rationale

Based on our discussion above, we identify some issues and sketch potential solutions. The first issue is how to avoid possible attacks resulting from rogue generation of group parameters by the GA. Since, in the RSA setting (e.g., [12]), a suitable solution would have to involve inefficient zero-knowledge proofs, it seems that moving to the DL-based setting would be more advantageous. The second issue is how to blind the registration process, while ensuring uniqueness of pseudonyms. An intuitive solution based on blind signatures works only if the registration process prevents the prospective member from choosing its pseudonyms freely. For reasons alluded to above, the scheme in [12] does not yield a straightforward solution to these issues. On the other hand, we observe that the SH scheme in [8] is amenable to modifications that do not introduce significant overhead. Below, we briefly describe this scheme and the design rationale for our modifications, introduced in the subsequent section.

Let $\mathcal{G} = \langle g \rangle$ denote a cyclic group of prime order q. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $H_1 : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ denote hash functions. Upon group initialization, the GA picks a private key $x \in \mathbb{Z}_q \setminus \{0\}$ and publishes its public key $y_G = g^x$. To issue a credential for pseudonym id the GA computes a Schnorr signature (ω, t) on id, i.e. $(\omega, t) = (g^r, r + xH(\omega \parallel \mathsf{id}))$ for some $r \in_R \mathbb{Z}_q \setminus \{0\}$, and hands it out to the corresponding user. Note that $g^t = \omega(y_G)^{H(\omega \parallel \mathsf{id})}$. In [8], element ω is considered as a public value associated with id from which g^t can be computed as described above, while t acts as a trapdoor for this value and is only known to the owner of id. The handshake protocol shown in Figure 2 treats (g^t, t) as public/private key pair for ElGamal encryption. In essence, it is the protocol proposed in [8], expanded from a four-move to a six-move protocol, for the sake of better readability.

Since its goal is Secret Handshakes, this scheme has not been analyzed with regard to session key security. We note that this scheme does not provide forward secrecy, since the session key is derived from encrypted nonces, which can be decrypted later upon corruption of participants. Similar to [12], it does not provide affiliation-hiding and member untraceability in the face of GA corruptions.

Section 3 describes our LAH-AKE protocol which incorporates two important modifications to [8] that address aforementioned challenges stemming from GA corruptions. First, we introduce a blinded registration process using blind Schnorr signatures [18, 17]. One nice property of blind Schnorr signatures is that both the signer and the verifier (i.e., the GA and the new member) contribute to the values (ω, t) of the resulting signature (see Figure 3). It follows that ω

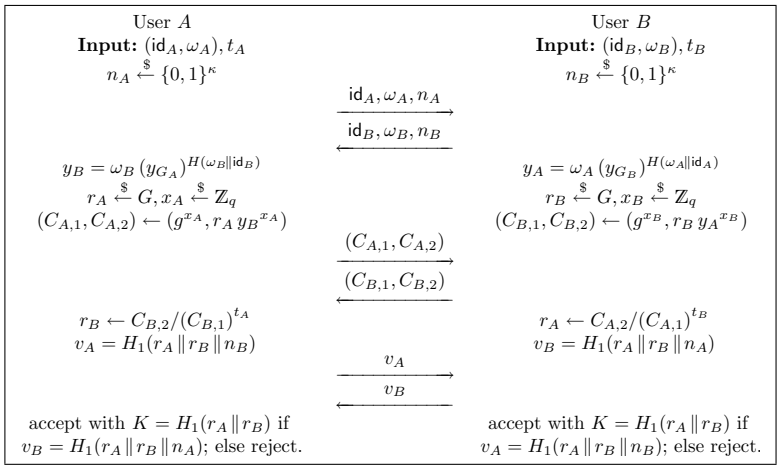

Fig. 2. DL-based Handshake protocol from [8]

can serve as the unique identifier of a group member. Therefore, we consider as user pseudonyms only the ω part of the signature, excluding all other identifiers. In other words: member pseudonyms together with secret user credentials form Schnorr signatures on the empty string. Our second tweak concerns the way session keys are computed. In our protocol they are derived from ephemeral Diffie-Hellman keys and only their authentication is performed using group credentials. The construction is similar to the Unified Model [4] and ensures forward secrecy with regard to later corruptions of both the GA and group members.

3 Untraceable LAH-AKE Protocol with Untrusted GAs

Our untraceable LAH-AKE scheme is inspired by the Secret Handshake protocol from [8] in which membership credentials are defined via Schnorr signatures. In order to meet stronger security and privacy requirements we make several substantial changes to the registration and key exchange procedures (for differences and design rationale see Section 2). We proceed with the description of algorithms and protocols.

Algorithm $\mathsf{Setup}(1^\kappa)$ This algorithm selects and publishes global parameters that are common to all users and group authorities. This is done by selecting security parameter κ' (polynomially dependent on κ), and by specifying a prime order cyclic group $(\mathcal{G}, g, q) \leftarrow \mathsf{GGen}(1^{\kappa'})$ and two hash functions $H^* : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $H : \{0,1\}^* \rightarrow \{0,1\}^{3\kappa}$.

Algorithm $\mathsf{CreateGroup}()$ The group authority GA picks a random secret key $G.\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q \setminus \{0\}$ and calculates the public key as $G.\mathsf{pk} = g^{G.\mathsf{sk}}$. The algorithm initializes the group's revocation list $G.\mathsf{prl}$ to $\emptyset$ and outputs $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$ as public group parameters, and $G.\mathsf{sk}$ as private group key.

Protocol $\mathsf{AddUser}(U, G)$ This protocol admits an user U to group G. The protocol as specified in Figure 3 is basically the blind variant of the Schnorr signature scheme [18, 17] where the empty message is signed by the group authority's secret key $G.\mathsf{sk}$. The communication between U and G is assumed to be authentic. Due to the blinding factors α and β the resulting signature (r', s') remains unknown to both eavesdroppers and the group authority. The output of this algorithm is the pair $(\mathsf{id}, \mathsf{id.cred}) = (r', s') \in \mathcal{G} \times \mathbb{Z}_q$ where id will be used as U's pseudonym in group G and $\mathsf{id.cred}$ as his secret credential. Note from inspection of the protocol that from $r' = g^{k+\alpha+\beta G.\mathsf{sk}}$ and $s' = k + (H^*(r') + \beta)G.\mathsf{sk} + \alpha$ it follows that

$$\mathsf{id}(G.\mathsf{pk})^{H^*(\mathsf{id})} = r'(G.\mathsf{pk})^{H^*(r')} = g^{s'} = g^{\mathsf{id.cred}}.$$

Note that neither U nor GA have exclusive control over the resulting values for id and $\mathsf{id.cred}$.

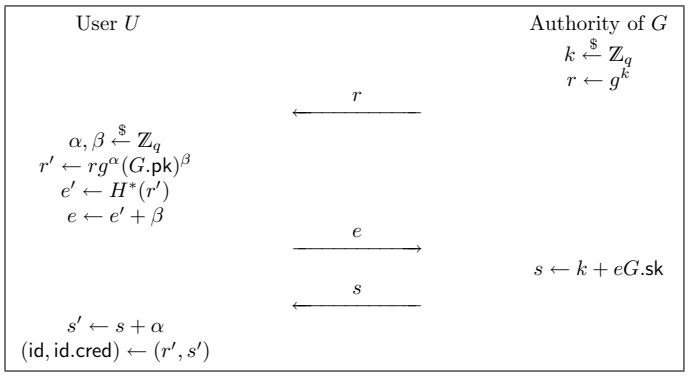

Fig. 3. Specification of our $\mathsf{AddUser}(U, G)$ protocol

Protocol $\mathsf{Handshake}((\mathsf{id}_A, \mathsf{id}_A.\mathsf{cred}, G_A.\mathsf{par}, \mathsf{init}), (\mathsf{id}_B, \mathsf{id}_B.\mathsf{cred}, G_B.\mathsf{par}, \mathsf{resp}))$ The handshake protocol is executed between two users A and B, holding pseudonyms id_A and id_B, private credentials $\mathsf{id}_A.\mathsf{cred}$ and $\mathsf{id}_B.\mathsf{cred}$ and public group parameters $G_A.\mathsf{par} = (G_A.\mathsf{pk}, G_A.\mathsf{prl})$ and $G_B.\mathsf{par} = (G_B.\mathsf{pk}, G_B.\mathsf{prl})$, respectively. The protocol is specified in Figure 4. Observe that the equality $L_A = g^{\mathsf{id}_A.\mathsf{cred} \cdot \mathsf{id}_B.\mathsf{cred}} = L_B$ is implied by property $\mathsf{id}(G.\mathsf{pk})^{H^*(\mathsf{id})} = g^{\mathsf{id.cred}}$, which is inherent for the correctness of the protocol.

Algorithm $\mathsf{Revoke}(G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id})$ The revocation of a pseudonym id from the group is handled by the particular group authority by including id in the corresponding pseudonym revocation list $G.\mathsf{prl}$. It is assumed that this list is distributed authentically.

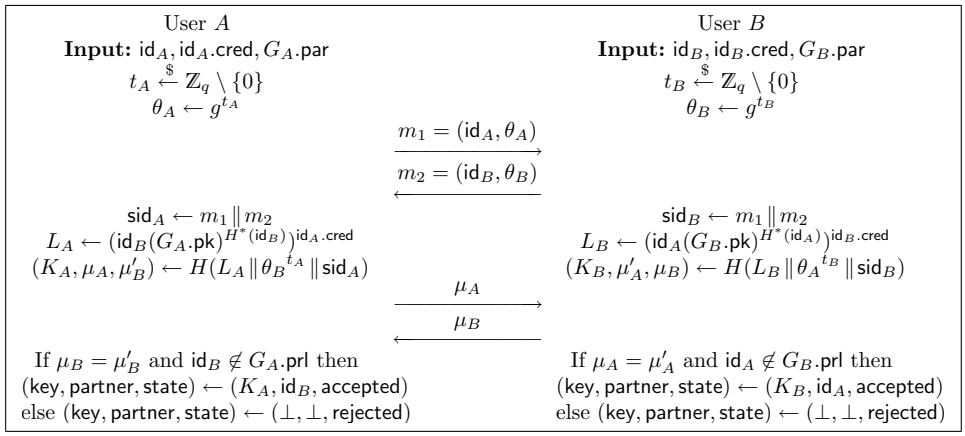

Fig. 4. Specification of our $\mathsf{Handshake}(U_A, U_B)$ protocol

4 Security Analysis and Performance Comparison

Security Analysis of our Protocol In order to prove security of our protocol, we need an extension of the classical CDH assumption where we provide CDH-adversary $\mathcal{A}$ with two exponentiation oracles H'_a and H'_b (that will be realized with a hash function H' and modeled as random oracles). The stronger decisional version of this assumption was named Oracle Diffie-Hellman Assumption in [1], for which it has been shown that, in the random oracle model, such hash oracles do not provide additional advantage for distinguishing the challenge. Therefore, it seems reasonable to assume that these oracles are not helpful for breaking the CDH challenge either.

Definition 2 (Oracle CDH (OCDH) Assumption). *Let* $\mathsf{GGen}(1^{\kappa'})$ *denote an algorithm that outputs the specification of a group* $(\mathcal{G}, \cdot) = \langle g \rangle$ *of prime order* $q \geq 2^{\kappa'}$*, let* $H' : \mathcal{G} \to \{0,1\}^{\kappa}$ *denote a public hash function where* κ *is polynomial in* κ'*. We define* $\mathsf{Succ}^{\mathsf{ocdh}}_{\mathsf{GGen},H'}(\kappa') =$

$$\max_{\mathcal{A}} \Pr\Big[(\mathcal{G}, g, q) \leftarrow \mathsf{GGen}(1^{\kappa'});\ a, b \xleftarrow{\$} \mathbb{Z}_q;\ h \leftarrow \mathcal{A}_{\mathcal{G}}^{H'_a, H'_b}(g^a, g^b) \text{ with } h = g^{ab}\Big]$$

where $H'_a : x \mapsto H'(x^a)$ *and* $H'_b : x \mapsto H'(x^b)$ *are oracles available to* $\mathcal{A}$*. The* OCDH assumption *states that there exist* GGen *and* H' *such that* $\mathsf{Succ}^{\mathsf{ocdh}}_{\mathsf{GGen},H'}(\kappa')$ *is negligible in* κ'*.*

Presuming the hardness of the OCDH problem, our construction presented in Section 3 satisfies the AKE-/LAH security and Untraceability goals introduced in Section 2 and formalized in the full version of this paper. The proof of Theorem 3 (with estimated attack probability) is given in Appendix A.3. For proofs of Theorems 1 and 2 we again refer to the full version.

Theorem 1 (AKE-Security). *Our LAH-AKE scheme is AKE-secure in the random oracle model under the OCDH assumption.*

Theorem 2 (LAH-Security). *Our LAH-AKE scheme is LAH-secure in the random oracle model under the OCDH assumption.*

Theorem 3 (Untraceability). *Our LAH-AKE scheme is unconditionally untraceable.*

Efficiency of our Protocol Our protocol offers strong security and privacy for users and remains very efficient at the same time. Some computations can be even further optimized. The registration protocol is performed only once per user and takes three exponentiations in $\mathcal{G}$. The handshake protocol requires two exponentiations for the computation of the Diffie-Hellman value $g^{t_A t_B}$ plus two additional exponentiations for the computation of the shared long-term authentication key $L_A = g^{\mathsf{id}_A.\mathsf{cred}\cdot\mathsf{id}_B.\mathsf{cred}} = L_B$. The latter two exponentiations can be omitted in future sessions with the same partner by caching long-term keys. Furthermore, if user pseudonyms are publicly listed then long-term keys can be pre-computed.

Comparison with [8] and [12] Table 1 compares security, privacy, and efficiency of the three protocols treated in this paper. We see that in respect to key security forward secrecy (FS) is provided only by [12] and our protocol, presuming honest behavior of the GA — denoted by hGA — for the former (otherwise, small group order attacks would be possible, see Section 2.2). In contrast, our protocol offers AKE security with forward secrecy even in the presence of corrupted GAs — denoted by cGA. As the user registration process in [8] and [12] is not blinded both protocols cannot provide LAH security if GAs are corrupted (as malicious GAs could record AddUser transcripts and later recognize affiliated pseudonyms). The same holds for user untraceability. The converse is correct for our protocol, which offers both properties even in the presence of corrupted GAs. As pointed out in Section 2.2, through the deployment of the blinding process revocation can be only performed based on pseudonyms.

Protocol	Security & Privacy				Revocation of	Complexity		
	AKE[1]	FS[2]	LAH[3]	UT[4]		Transf. bits[5]	# passes[6]	# exps[7]
CJT [8]	hGA	✗	hGA	✗	users,pseudonyms	$2(3\kappa' + 3\kappa)$	4	3 short
JKT [12]	hGA	✓	hGA	✗	users,pseudonyms	$2(\kappa'' + 3\kappa)$	3	2 long
Ours	cGA	✓	cGA	✓	pseudonyms	$2(2\kappa' + \kappa)$	3	2 short

[1]AKE-Security; [2]Forward Secrecy; [3]LAH-Security; [4]Untraceability; [5]Total number of transferred bits per handshake; [6]Number of message passes per protocol run; [7]Number of exponentiations (with short ($\approx 2\kappa$ bit) or long ($\approx \kappa''$ bit) exponents)

Table 1. Security and Performance Comparison with [8] and [12]

The security advantage of our protocol is gained very efficiently: our protocol has best message and computational complexity (using optimal arrangement of

messages and not counting cacheable computations). In practice security parameters $\kappa = 80$, $\kappa'' = 1024$ and $\kappa' = 1024$ (standard group setting) or $\kappa' = 2\kappa = 160$ (ECC group setting) would be chosen. In the latter case our protocol has the smallest bandwidth complexity of all named protocols (of about $10\kappa = 800$ bits per full handshake).

5 Conclusion

SH and AH-AKE schemes provide useful privacy-preserving authentication mechanisms coupled with the establishment of secure session keys. These schemes are becoming more important due to the increasing popularity of multi-user collaborative and group-based applications. Existing approaches and security models assume unconditional trust in Group Authorities. In this paper, we demonstrated that such trust assumptions might become problematic. We illustrated that these assumptions can be relaxed in a meaningful way resulting in more secure and private (yet efficient and practical) AH-AKE schemes. Our work opens a new research direction: Consideration of untrusted Group Authorities in unlinkable [2, 13, 14] and multi-party AH-AKE schemes [10, 11].

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, 2001.
2. G. Ateniese, J. Kirsch, and M. Blanton. Secret Handshakes with Dynamic and Fuzzy Matching. In *Network and Distributed System Security Symposium (NDSS 2007)*. The Internet Society, 2007.
3. D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong. Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy 2003*, pages 180–196. IEEE CS, 2003.
4. S. Blake-Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. In *6th IMA Intl. Conference on Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer, 1997.
5. M. Burmester and Y. Desmedt. A Secure and Efficient Conference Key Distribution System. In *EUROCRYPT 1994*, volume 950 of *LNCS*, pages 275–286. Springer, 1994.
6. J. Camenisch and M. Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 107–122. Springer, 1999.
7. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and their Use for Building Secure Channels. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.
8. C. Castelluccia, S. Jarecki, and G. Tsudik. Secret Handshakes from CA-Oblivious Encryption. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 293–307. Springer, 2004.
9. D. Chaum. Blind Signatures for Untraceable Payments. In *CRYPTO 1982*, pages 199–203. Plenum Press, 1983.

10. S. Jarecki, J. Kim, and G. Tsudik. Authentication for Paranoids: Multi-Party Secret Handshakes. In *4th Intl. Conference on Applied Cryptography and Network Security (ACNS 2006)*, volume 3989 of *LNCS*, pages 325–339. Springer, 2006.
11. S. Jarecki, J. Kim, and G. Tsudik. Group Secret Handshakes or Affiliation-Hiding Authenticated Group Key Agreement. In *CT-RSA 2007*, volume 4377 of *LNCS*, pages 287–308. Springer, 2007.
12. S. Jarecki, J. Kim, and G. Tsudik. Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In *CT-RSA 2008*, volume 4964 of *LNCS*, pages 352–369. Springer, 2008.
13. S. Jarecki and X. Liu. Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In *5th Intl. Conference on Applied Cryptography and Network Security (ACNS 2007)*, volume 4521 of *LNCS*, pages 270–287. Springer, 2007.
14. S. Jarecki and X. Liu. Private Mutual Authentication and Conditional Oblivious Transfer. In *CRYPTO 2009*, volume 5677 of *LNCS*, pages 90–107. Springer, 2009.
15. Y. Kawai, K. Yoneyama, and K. Ohta. Secret Handshake: Strong Anonymity Definition and Construction. In *5th Intl. Conference on Information Security Practice and Experience (ISPEC 2009)*, volume 5451 of *LNCS*, pages 219–229. Springer, 2009.
16. M. Manulis, B. Poettering, and G. Tsudik. Affiliation-Hiding Key Exchange with Untrusted Group Authorities. In *8th Intl. Conference on Applied Cryptography and Network Security (ACNS 2010)*, volume 6123 of *LNCS*, pages 402–419. Springer, 2010.
17. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
18. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.
19. G. Tsudik and S. Xu. A Flexible Framework for Secret Handshakes. In *6th Intl. Workshop on Privacy Enhancing Technologies (PET 2006)*, volume 4258 of *LNCS*, pages 295–315. Springer, 2006.
20. D. Vergnaud. RSA-Based Secret Handshakes. In *Intl. Workshop on Coding and Cryptography (WCC 2005)*, volume 3969 of *LNCS*, pages 252–274. Springer, 2005.
21. S. Xu and M. Yung. k-Anonymous Secret Handshakes with Reusable Credentials. In *11th ACM Conference on Computer and Communications Security (CCS 2004)*, pages 158–167. ACM, 2004.

A Model and Proof for Untraceability

In the appendix of the full version of this paper formal security models and proofs for AKE-security, LAH-security and Untraceability are given. While, in respect to the two former models, we extend the work of [12], the property of untraceability is newly introduced here. Due to space limitations, in this version, we restrict the focus on model and proof of the latter.

A.1 Adversary Model

The adversary $\mathcal{A}$ is modeled as a PPT machine that interacts with parties via the set of the following basic queries. Unless explicitly noted, we assume that $\mathcal{A}$

always has access to up-to-date exhaustive (system-wide) lists of groups GLi and pseudonyms IDLi (these lists do not disclose the mapping between pseudonyms and groups).

CreateGroup() This query sets up a new group G and publishes its public parameters $G.\mathsf{par}$. The group is added to GLi.

AddUserU($U, G.\mathsf{par}$) This query models the actions of U initiating the AddUser protocol with given target group G. A new protocol session π is started. Optionally, a first protocol message M is output. G is also added to GLi if it is a new group; this allows $\mathcal{A}$ to create its own groups with arbitrary (possibly malicious) public parameters.

AddUserG(G, U) This query differs from AddUserU in that it models GA's actions on the AddUser protocol. We require that G has been already established through the CreateGroup query.

Handshake($\mathsf{id}, G.\mathsf{par}, r$) This query lets pseudonym id start a new session π of the Handshake protocol. It receives as input the public parameters of the group G wherein the handshake shall take place (given that id has credentials for that group) and a role identifier $r \in \{\mathsf{init}, \mathsf{resp}\}$ that determines whether the session will act as protocol initiator or responder. Optionally, this query returns a first protocol message M.

Send(π, M) Message M is delivered to session π. After processing M, the eventual output is given to $\mathcal{A}$. This query is ignored if π is not waiting for input. Note that π is either an AddUserU, an AddUserG or a Handshake protocol session. If π is an AddUserU session and accepts after processing M then id from $\pi.\mathsf{result}$ is added to IDLi.

Reveal(π) This query is defined only if π is a handshake session. Then, if $\pi.\mathsf{state} \neq \mathsf{running}$ it returns $\pi.\mathsf{state}$ and $\pi.\mathsf{key}$; otherwise the query is ignored.

Corrupt($*$) The input is either a pseudonym id or a group identifier G:

Corrupt(id): If $\mathsf{id} \in$ IDLi then, for any group G where id is registered, the corresponding credential $\mathsf{id.cred}$ is given to $\mathcal{A}$.

Corrupt(G): For a group G created by CreateGroup() this query hands G's long term secret $G.\mathsf{sk}$ and control over G's revocation list $G.\mathsf{prl}$ over to $\mathcal{A}$.

Revoke(G, id) This query lets the GA of G include $\mathsf{id} \in$ IDLi in its pseudonym revocation list $G.\mathsf{prl}$.

A.2 Definition of Untraceability

The idea behind untraceability is that, even in the presence of a malicious GA, any member remains untraceable throughout its AH-AKE sessions. As discussed in Section 1, this is a new (individual) privacy requirement, distinct from AKE- and LAH-security. We formalize it using the indistinguishability approach: we let $\mathcal{A}$ specify group parameters for a group G and pick two users U_0 and U_1 that are then enrolled into G by the challenger that obtains their respective pseudonyms id_0 and id_1. Untraceability means the inability of $\mathcal{A}$ to trace id_b where $b \in_R \{0, 1\}$.

Definition 3 (Untraceability). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*, b a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Reveal}, \mathsf{Corrupt}, \mathsf{Revoke}\}$ *the set of queries available to $\mathcal{A}$. By* $\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *we denote the following interaction of $\mathcal{A}$ with participants, where, for obvious reasons, we prevent $\mathcal{A}$ from accessing the up-to-date pseudonym list* IDLi*:*

- $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ *interacts with all participants using the queries in $\mathcal{Q}$ and outputs a triple $(G.\mathsf{par}, U_0, U_1)$ where $G.\mathsf{par}$ are public parameters of a group G and U_0 and U_1 are two distinct users.*
- *U_0 and U_1 are admitted to G through the execution of $\mathsf{AddUser}(U_0, G)$ and $\mathsf{AddUser}(U_1, G)$ protocols in which the corresponding pseudonyms id_0 and id_1 are generated. Note that, during this process, protocol sessions on behalf of G can be executed by $\mathcal{A}$, however, the game does not proceed until the corresponding protocol sessions executed on behalf of U_0 and U_1 accept.*
- *$\mathcal{A}$ is given id_b and continues to interact with all participants via queries until it terminates and outputs bit b', which is also the output of the game.*

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

and denote by $\mathsf{Adv}^{\mathsf{trace}}_{\mathsf{LAH\text{-}AKE}}(\kappa)$ *the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that* $\mathsf{LAH\text{-}AKE}$ *is* untraceable *if this advantage is negligible (in κ).*

A.3 Proof of Untraceability (Theorem 3)

It is well known that blind Schnorr signatures (see Figure 3) offer unconditional blinding [18, 17]. In fact the two blinding values α and β as used in the $\mathsf{AddUser}$ protocol act as one-time-pad encryption in $\mathbb{Z}_q$ and therefore offer perfect secrecy. It follows directly that the group authority of G cannot learn any information about id or $\mathsf{id.cred}$ as established by an $\mathsf{AddUser}$ protocol session. Therefore, the probability that for a given pseudonym id_b in $\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ an Untraceability adversary $\mathcal{A}$ can output $b = b'$ is strictly the probability of a random guess, i.e. $1/2$. Hence our LAH-AKE protocol offers unconditional untraceability, i.e. we have

$$\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa) = 0 \qquad \text{(for all } \kappa\text{)}$$

Observe that $\mathsf{Handshake}$ sessions are completely independent of the user running them and depend solely on the deployed pseudonym id and membership credential $\mathsf{id.cred}$. Even a $\mathsf{Corrupt}(\mathsf{id})$ query does not reveal the owning user of the given id.