Affiliation-Hiding Key Exchange with Untrusted Group Authorities

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. Privacy-preserving techniques are increasingly important in our highly computerized society where privacy is both precious and elusive. Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols offer an appealing service: authenticated key agreement coupled with privacy of group memberships of protocol participants. This type of service is essential in privacy-conscious p2p systems, mobile ad hoc networks and social networking applications. Prior work has succeeded in constructing a number of secure and efficient AH-AKE protocols which all assume full trust in the Group Authority (GA) — the entity that sets up the group as well as registers and (optionally) revokes members. In this paper, we argue that, for many anticipated application scenarios, the trusted GA model should be relaxed to allow for certain types of malicious behavior. We examine the consequences of malicious GAs and explore the design of stronger AH-AKE protocols that withstand GA attacks. Our results demonstrate that such protocols are both feasible and practical.

1 Introduction

Privacy-Preserving Authentication. Secret Handshakes (SH) [2,8,23,22,21, 1,16,17] and Affiliation-Hiding Authenticated Key Exchange (AH-AKE) protocols [14,15] offer privacy-preserving authentication among members of the same group. A user joins a group and obtains its membership credential by registering with the Group Authority (GA). In some schemes, the GA can later revoke these credentials [2, 8, 23, 15, 17]. Possession of valid credentials by both participants is a requirement for a successful handshake. Privacy of the authentication process is defined as the requirement to hide the affiliations (group memberships) of protocol participants from outsiders, as well as from each other, unless their respective affiliations match. Since the authentication process is usually followed by a communication session, AH-AKE protocols [14, 15] combine secret handshakes with session key establishment. Although session keys are also provided by many secret handshake schemes, the distinguishing feature of AH-AKE protocols is ensuring that session key leakage does not reveal any information about affiliations of session participants. Additionally, AH-AKE protocols guarantee

J. Zhou and M. Yung (Eds.): ACNS 2010, LNCS 6123, pp. 402–419, 2010.

the usual security requirements, i.e., authenticated key exchange security with forward secrecy [7].

Both secret handshakes and AH-AKE protocols can be linkable or unlinkable, depending on whether sessions of the same group member can be related. Linkable secret handshakes and AH-AKE protocols [2, 8, 14, 15] are useful if participants wish to be recognized across different sessions, while keeping their affiliations hidden. This property is typically realized via reusable pseudonyms obtained by members during the registration process. Unlinkable secret handshakes and AH-AKE protocols [23,16,1,17] prevent any correlation among multiple sessions.

Untrusted Group Authorities. Current secret handshakes and AH-AKE protocols assume that group authorities are trusted. This assumption is apparent in the security models of [16,15,17] where corruptions of GAs are not considered. We believe that there are two main reasons for the trusted GA assumption: (a) anticipated applications of secret handshakes were in the domain of homeland security (e.g. intelligence agencies, police) where GA trustworthiness is not questioned, and (b) the GA's role has been considered as being similar to that of a certification authority (CA) in classical PKI-based scenarios where such corruptions are counter-intuitive as they would allow the CA to certify new users at will. However, we consider a more general context of open and commercial applications, such as p2p systems or social networks that allow for the creation of multiple groups, each administrated by its own GA.

In such settings, unconditional trust in the GA is problematic because, unlike a CA only trusted with security, a GA is trusted with both security and privacy. We argue that it is reasonable to trust a GA not to register members frivolously or fraudulently, whereas trusting a GA to maintain and respect privacy of members is a matter that should be treated separately. We now informally discuss the impact of untrusted GAs on security of AH-AKE protocols.

In general, a malicious GA might attempt any of the following: generate public group parameters in a rogue way, create phantom group members, misbehave during the registration process of honest members, and mount active attacks on sessions involving honest members. The resulting challenge is: Which security requirements of AH-AKE protocols can be preserved in a meaningful way? For example, if a malicious GA creates a phantom member and participates on its behalf in an AH-AKE protocol with another party, then this GA would be able to check whether that someone is a member of its group. Moreover, if the protocol succeeds, the GA would also compute the corresponding session key. However, this is an unavoidable consequence of trusting the GA with the security of registration. In fact, it is conceivable that all registration processes are logged and are auditable by some higher authority, thus dis-incentivizing the GA from registering phantom members.

Therefore, we claim that it is meaningful to restrict GA misbehavior to attacks on affiliation-hiding and key secrecy for sessions involving honest members. To this end, our goal is to explore approaches to preserve both of these properties even in the face of a misbehaving GA. In linkable AH-AKE protocols, hiding

affiliations of participants appears to be especially challenging due to the use of pseudonyms created during the registration phase. Indeed, none of the linkable AH-AKE protocols we are aware of can attain this goal. There are also some protocols, e.g. [2,8,1], where knowledge of GA secrets would immediately reveal session keys. Additionally, consideration of malicious GAs leads us to a new privacy goal, which we call *untraceability* — the infeasibility for a malicious GA to learn real identities of honest members through their AH-AKE sessions. This requirement is beneficial even for sessions executed with phantom members introduced by the GA. Intuitively, untraceability is related to the GA's ability to obtain information during the registration phase of an honest member that allows it to later link AH-AKE sessions of that member to the registration process, and thus, to the real identity. We observe that many current linkable protocols [2, 8, 22, 15] do not provide this property. This may have serious impact on user privacy. Consider for example an anti-government social network that operates in an oppressive regime. The social network controls the GA and issues credentials to its members. Members can then identify each other and hold discrete meetings and conversations. However, if the government raids the social network and confiscates the GA, then all parameters and records become exposed and the government can thereafter trace and identify all members.

Contributions and Organization. In this paper, we address the challenge posed by untrusted GAs in linkable AH-AKE (LAH-AKE) protocols. This research direction is particularly interesting, since it is counter-intuitive to tolerate linkability of AH-AKE sessions and provide meaningful support for revocation, while offering untraceability against malicious GAs. Our work models untrusted GAs in LAH-AKE protocols and yields stronger LAH-AKE schemes with meaningful security guarantees. Specifically, we extend work by Jarecki, et al. [15], which constructed a LAH-AKE protocol in the trusted GA model.

In Section 2, we propose some updates to the syntax of LAH-AKE protocols and to their security model in order to accommodate corruptions of GAs. More precisely, we extend the adversary model to allow adversarial control of GAs, and define the aforementioned untraceability property. We also update former definitions of authenticated key exchange (AKE) security and linkable affiliation-hiding (LAH) security from [15] to take into account malicious GAs.

In Section 3, we show how the protocol from [15], with security proven under the RSA assumption on safe moduli [13, 12] in the random oracle model (ROM) [4], can be fortified to provide new security properties without increasing the round complexity of the original protocol. Our modifications involve several "tricks". First, the initial generation of group parameters is extended with slightly modified (non-interactive) zero-knowledge proofs from [6] to prevent rogue values from being chosen by the GA. Second, we construct an anonymized registration process using the blind RSA signature scheme [9, 3] to ensure privacy of obtained pseudonyms and membership credentials. This modification is essential to achieve untraceability. In order to prevent impersonation attacks in AH-AKE sessions we link pseudonyms to public keys of some existentially-unforgeable signature scheme. The idea is to let new members choose their corresponding private

signature keys during the registration process and use them during each AH-AKE session to sign key confirmation messages. This allows us to ensure that each member proves ownership of the pseudonym used during protocol execution. By linking pseudonyms to public signature keys we also allow members to use the same pseudonym in different groups. Similar to [15], we can still support revocation of pseudonyms via revocation lists.

The security and efficiency analysis of our protocol is given in Section 4. We stress that security of our scheme no longer relies solely on the RSA problem (since GA knows the factors of the modulus n) but also on the Decisional Diffie-Hellman assumption [5] in subgroups of $\mathbb{Z}_n^*$ of maximal order.

Related Work. Early secret handshake schemes [2,8,22] provided group members with pseudonyms and secret credentials, as part of the registration process. Such schemes are linkable since the same pseudonym is used in multiple handshakes. Unlinkability can be trivially obtained by using one-time pseudonyms; however, this is clearly unscalable. In [2], a credential is a secret element of a bilinear group; [8] uses special CA-oblivious PKI-enabled encryption realized via Schnorr signatures; and [22, 15] use blinded verification of RSA signatures.[1] As the next step, [15] introduced an LAH-AKE protocol that offers both linkable affiliation-hiding and key exchange security with forward secrecy.

Unlinkable secret handshake schemes [1,16,21,17] support reusable credentials while precluding correlation of multiple sessions involving the same participant. However, [1] does not support revocation; [16] requires users to synchronize revocation epochs, and [21] is a complex scheme based on group signatures and broadcast encryption, which complicates revocation. Another flavor of unlinkability (*k-anonymity*) was explored in [23]. Based on some group signature-related techniques, Jarecki and Liu [17] recently proposed another construction with revocation and unlinkable reusable credentials. However, this scheme is hardly practical as it is based on pairings and has linear computation complexity in the number of revoked users.

The only current work on privacy-protection against group authorities is due to Kawai, et al. [18], who deviate from the classical setting and split the role of the group authority among two entities: the issue authority responsible for the registration of users and issue of certificates, and the tracing authority that can trace users from their handshakes. Of particular interest is the new notion of *co-traceability*, which is supposed to prevent authorities from identifying users upon their handshake sessions. The crucial assumption for this property in [18] is that the issue and tracing authorities do not collude. In contrast, our setting is more consistent with prior work, since we treat the group authority as a monolithic entity. Also, our model considers security of session keys and their impact on user privacy, whereas, [18] builds upon [8] where these issues are not modeled. [18] requires group signatures with message recovery and its operation is based on pairings, which significantly decreases the efficiency of the scheme (note that no performance analysis is given in [18]).

[1] Note that the protocol in [22] was shown to be insecure in [15].

2 Untrusted GA Model for Linkable AH-AKE Protocols

We now define and model the security of LAH-AKE protocols while considering malicious GA behavior.

2.1 Linkable Affiliation-Hiding Key Exchange Syntax

An LAH-AKE scheme is a four-tuple $\{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$ with components defined as follows:

$\mathsf{CreateGroup}(1^\kappa)$**.** This probabilistic algorithm sets up a new group G and 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 $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl})$ and private key $G.\mathsf{sk}$.

$\mathsf{AddUser}(U, G)$**.** This protocol is executed between the prospective group member U and the GA of G. The algorithm on U's side is denoted $\mathsf{AddUserU}(U, G.\mathsf{par})$, the algorithm on GA's side by $\mathsf{AddUserG}(U, G.\mathsf{sk})$. Let π be a session of either the AddUserU or the AddUserG algorithm. The *state* of π is defined through the session variable $\pi.\mathsf{state}$ and can take running, accepted, or rejected values. For both algorithms 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, and the same pseudonym may be registered in different groups.

$\mathsf{Handshake}(\mathsf{params}_1, \mathsf{params}_2)$**.** This is 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 $r_2 = \mathsf{resp}$. We assume that each U_i executes the corresponding interactive algorithm $\mathsf{Handshake}'(\mathsf{param}_i)$. Note that id_i is the pseudonym previously registered to group G_i using the AddUser algorithm. The protocol verifies that both users are affiliated to 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 pseudonym used) and $\pi.\mathsf{partner} \leftarrow \perp$. At some point, the protocol will complete and $\pi.\mathsf{state}$ is then 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 $\pi.\mathsf{partner}$. The accepted state cannot be reached if the protocol partner is revoked from the group ($\pi.\mathsf{partner} \in G.\mathsf{prl}$).

$\mathsf{Revoke}(G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id})$**.** This algorithm is executed by the GA of G and results in the update of G's pseudonym revocation list: $G.\mathsf{prl} \leftarrow G.\mathsf{prl} \cup \{\mathsf{id}\}$.

Definition 1 (Correctness of LAH-AKE). *Assume that two users,* U_1 *and* U_2*, register as members of groups* G_1 *and* G_2*, and obtain their credentials* (id_1, $\mathsf{id}_1.\mathsf{cred}$) *and* ($\mathsf{id}_2, \mathsf{id}_2.\mathsf{cred}$)*, respectively, through corresponding* AddUser *executions. Assume that* U_1 *and* U_2 *participate in a* Handshake *protocol and let* π_1 *and* π_2 *denote the corresponding sessions of* U_1 *and* U_2*. The LAH-AKE scheme is called* correct *if (a)* π_1 *and* π_2 *complete in the same state, which is* 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 Security Model and Extended Goals

We now present our security model that takes into account malicious GAs. After describing adversarial queries we define three security properties: authenticated key exchange (AKE) security (with forward secrecy), linkable affiliation-hiding (LAH) security, and untraceability. Our model and definitions build upon [15], which considered the first two properties in the trusted GA model.

Adversary Model. The adversary $\mathcal{A}$ is modeled as a PPT machine that interacts with protocol participants 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.

$\mathsf{Corrupt}(*)$**.** The input is either a pseudonym id or a group identifier G:
$\mathsf{Corrupt}(\mathsf{id})$: If $\mathsf{id} \in \mathsf{IDLi}$ then, for any group G where id is registered, the corresponding credential $\mathsf{id.cred}$ is given to $\mathcal{A}$.
$\mathsf{Corrupt}(G)$: For a group G created by $\mathsf{CreateGroup}()$ this query hands G's long term secret $G.\mathsf{sk}$ and control over G's revocation list $G.\mathsf{prl}$ over to $\mathcal{A}$.

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

Definition 2 (Honest Generation of Pseudonyms and Groups). *A* pseudonym id *is called* honestly generated *if it was established through an* $\mathsf{AddUserU}$ *query. It is called* honest *if thereafter no* $\mathsf{Corrupt}(\mathsf{id})$ *query has been asked. Similarly,* group G *is called* honestly generated *if it was established through a* $\mathsf{CreateGroup}$ *query. It is called* honest *if thereafter no* $\mathsf{Corrupt}(G)$ *query has been asked.*

Definition 3 (Session IDs and Partnered Sessions). Session id $\pi.\mathsf{sid}$ *of a* $\mathsf{Handshake}$ *session* π *that was initiated by pseudonym* id *and is in state* $\mathsf{accepted}$ *is a value that uniquely identifies* π *in the set of all protocol sessions of* id*. Two* $\mathsf{Handshake}$ *sessions* π, π' *are called* partnered *if* $\pi.\mathsf{state} = \pi'.\mathsf{state} = \mathsf{accepted}$ *and* $(\pi.\mathsf{sid}, \pi.\mathsf{id}, \pi.\mathsf{partner}) = (\pi'.\mathsf{sid}, \pi'.\mathsf{partner}, \pi'.\mathsf{id})$.

Authenticated Key Exchange Security. AKE-security of LAH-AKE protocols is determined by analyzing the statistical distribution of resulting session keys: $\mathcal{A}$'s task is to distinguish a key established in a protocol run from a randomly generated value of the same length.

In order to formalize the corresponding indistinguishability game we first introduce two new flags $\pi.\mathsf{revealed}$ and $\pi.\mathsf{tested}$, that are initially set to false, and define the Reveal^* query (as a slightly modified version of Reveal) and the auxiliary Test query with secret parameter $b \in \{0, 1\}$.

$\mathsf{Reveal}^*(\pi)$**.** This query is answered as the regular $\mathsf{Reveal}(\pi)$ query (and $\pi.\mathsf{revealed}$ is set to true) unless $\pi.\mathsf{tested} = \mathsf{true}$ or $\pi'.\mathsf{tested} = \mathsf{true}$ for any session π' that is partnered with π. In the latter case the query is ignored.

$\mathsf{Test}(\pi)$**.** This query is ignored if π is not fresh (see Definition 4). Otherwise $\pi.\mathsf{tested}$ is set to true and a key is returned, obeying the following rule: Let $b \in \{0, 1\}$ denote a bit chosen in advance. In case $b = 1$ $\pi.\mathsf{key}$ is returned. In case $b = 0$ a random element drawn uniformly from $\{0, 1\}^\kappa$ is returned instead. The Test query may be invoked at most once.

The following notion of freshness is useful to exclude trivial attacks and simplify the definition of AKE-security.

Definition 4 (Session Freshness). *A session* π *that is invoked in response to* $\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$ *for an honestly generated* id *is called* fresh *if all of the following hold:*

(a) $\pi.\mathsf{state} = \mathsf{accepted}$ *and* $\pi.\mathsf{revealed} = \mathsf{false}$*;*
(b) for any existing session π' *that is partnered with* π*,* $\pi'.\mathsf{revealed} = \mathsf{false}$ *and* $\mathsf{Corrupt}(\pi'.\mathsf{id})$ *was not invoked prior to setting* $\pi'.\mathsf{state} = \mathsf{accepted}$*. Note that in this case* $\pi'.\mathsf{id} = \pi.\mathsf{partner}$*;*
(c) $\pi.\mathsf{partner}$ *was honestly generated and* $\mathsf{Corrupt}(\pi.\mathsf{partner})$ *was not invoked prior to setting* $\pi.\mathsf{state} = \mathsf{accepted}$ *OR* $\pi.\mathsf{partner}$ *was not honestly generated and prior to setting* $\pi.\mathsf{state} = \mathsf{accepted}$ *both* G *was honest and no* $\mathsf{AddUserG}(G, \cdot)$ *has been asked;*

We now provide some rationale: Conditions (a) and (b) prevent $\mathcal{A}$ from revealing the session key computed by π or its partnered session π' and also model forward secrecy by allowing $\mathcal{A}$ to corrupt the corresponding members after the computation of the session key. Condition (c) states meaningful requirements on $\pi.\mathsf{partner}$: on the one hand, it prevents the corruption of honestly generated $\pi.\mathsf{partner}$ during the execution of the handshake (otherwise $\mathcal{A}$ can trivially act in the session on behalf of $\pi.\mathsf{partner}$ and compute the key); on the other hand, it prevents the trivial attack where $\pi.\mathsf{partner}$ was introduced either by the malicious GA of G or as a consequence of the $\mathsf{AddUserG}$ query with which $\mathcal{A}$ could otherwise obtain regular credentials for G. Note that we allow $\mathcal{A}$ to corrupt $\pi.\mathsf{id}$ at any time (even before protocol acceptance) without considering π as not fresh. Essentially this also models resilience against Key Compromise Impersonation (KCI) attacks [19]. We are now ready to formally define AKE-security.

Definition 5 (AKE-Security with Forward Secrecy). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*,* b *be a bit chosen at random, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Reveal}^*, \mathsf{Test}, \mathsf{Corrupt}, \mathsf{Revoke}\}$ *denote the set of queries available to* $\mathcal{A}$*. By* $\mathsf{Game}^{\mathsf{ake},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *we denote the following game:*

- $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ *interacts with all participants using the queries in* $\mathcal{Q}$*;*
- *at some point* $\mathcal{A}^{\mathcal{Q}}$ *asks* $\mathsf{Test}(\pi^*)$ *to a session* π^* *which is* fresh*;*
- $\mathcal{A}^{\mathcal{Q}}$ *continues interacting via queries until it terminates and outputs bit* b'*, which is the output of the game.*

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

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

In contrast to [15] we do not assume that GAs remain uncorrupted. We do not even restrict $\mathcal{A}$ from setting up a group on its own (presumably by choosing some 'odd' parameters) or from letting honest users register in such groups. Thus, we allow $\mathcal{A}$ to invoke Test query to a session of an honest user that registered with some malicious GA (as long as that session satisfies the freshness conditions).

Linkable Affiliation-Hiding Security. In order to define linkable affiliation-hiding (LAH) security we adopt the simulation-based approach from [15]. The idea is to require that the real protocol execution remains indistinguishable from an idealized one performed by a simulator $\mathcal{SIM}$ that simulates handshake executions without knowing participants' affiliation. This indistinguishability can be defined through a game played between $\mathcal{A}$ and the challenger $\mathcal{C}^b$ initialized with a secret bit $b \in_R \{0,1\}$. $\mathcal{C}^1$ answers all queries of $\mathcal{A}$ honestly following the real protocol specification. $\mathcal{C}^0$ answers the queries of $\mathcal{A}$ with help of $\mathcal{SIM}$ as shown below.

We call a group G *trivially intrudable* if G was not setup honestly through a $\mathsf{CreateGroup}(\,)$ query or if an $\mathsf{AddUserG}(G,\cdot)$ query has been posed by $\mathcal{A}$ or if $\mathcal{A}$ corrupted some pseudonym id' generated in response to some $\mathsf{AddUserU}(\cdot, G.\mathsf{par})$ query. This means that for all trivially intrudable groups $\mathcal{A}$ can obtain valid membership credentials in a trivial way. Therefore, the idea is to let $\mathcal{SIM}$ simulate sessions on behalf of honest pseudonyms only if they belong to groups that are not trivially intrudable.

$\mathsf{CreateGroup}()$, $\mathsf{AddUserU}(U, G.\mathsf{par})$, $\mathsf{Revoke}(G, \mathsf{id})$. These queries are answered honestly without involving $\mathcal{SIM}$.

$\mathsf{AddUserG}(G, U)$,$\mathsf{Corrupt}()$. These queries are answered honestly without involving $\mathcal{SIM}$ unless there exists some still running handshake session π invoked for a group G which is not trivially intrudable. In this latter case $\mathsf{AddUserG}$ and $\mathsf{Corrupt}$ queries are ignored if their input is such that after processing these queries the group G would become trivially intrudable.

$\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$. We distinguish between two cases:
Case 1 if G is trivially intrudable then $\mathcal{C}^0$ correctly answers the query. We call the invoked session a Case 1-session.
Case 2 if G is not trivially intrudable then $\mathcal{C}^0$ invokes $\mathcal{SIM}.\mathsf{Handshake}(\mathsf{id}, r)$ and relays its reply. Note that, in this case, $\mathcal{SIM}$ doesn't learn the group parameters $G.\mathsf{par}$ from this query. We call the invoked session a Case 2-session.

$\mathsf{Send}(\pi, M)$. If π is an $\mathsf{AddUserU}$ or $\mathsf{AddUserG}$ session then $\mathcal{C}^0$ answers the query itself. If π is a Case 1-session then $\mathcal{C}^0$ correctly answers the query, whereby, if after processing M session π accepts then $\mathcal{C}^0$ sets the corresponding $\pi.\mathsf{key}$ as follows: If there exists a session π' partnered to π then $\pi.\mathsf{key} \leftarrow \pi'.\mathsf{key}$ is set; Otherwise, if "π is fresh", i.e., if $\pi.\mathsf{partner}$ was honestly generated and $\mathsf{Corrupt}(\pi.\mathsf{partner})$ was not asked OR $\pi.\mathsf{partner}$ was not honestly generated and both G is honest and $\mathsf{AddUserG}(G,\cdot)$ was not asked, then pick $\pi.\mathsf{key} \in_R \{0,1\}^\kappa$; Otherwise, set $\pi.\mathsf{key}$ according to the protocol specification. If π is a Case 2-session then $\mathcal{C}^0$ invokes $\mathcal{SIM}.\mathsf{Send}(\pi, M)$ and relays its reply.

We let $\mathcal{C}^0$ create sessions π and process corresponding $\mathsf{Send}(\pi,\cdot)$ queries correctly, without involving $\mathcal{SIM}$ (Case 1-sessions), in all cases for which $\mathcal{A}$ would be able to break the secrecy of $\pi.\mathsf{key}$ in the real protocol execution (during the interaction with $\mathcal{C}^1$) anyway, by the means of some trivial attack. In cases where $\mathcal{A}$ is not supposed to break the secrecy of $\pi.\mathsf{key}$ in the real protocol (the conditions are equivalent to those for session freshness in Definition 4) $\pi.\mathsf{key}$ is chosen randomly.

Before describing how $\mathcal{C}^0$ answers the $\mathsf{Reveal}(\pi)$ queries of $\mathcal{A}$ we define the auxiliary notion of compatible sessions.

Definition 6 (Compatible Sessions). *Two protocol sessions π, π' initiated by* Handshake *queries are called* compatible *if the groups associated with π, π' are identical, and the concatenation of the messages received by π is a prefix of the concatenation of messages sent by π' and vice versa. If π is a session that received enough messages to complete and there exists a compatible session π' then $\pi.\mathsf{id}$ and $\pi.\mathsf{partner}$ are set to the pseudonyms of the initiators of π and π'.*

$\mathsf{Reveal}(\pi)$**.** If π is a CASE 1-session and $\pi.\mathsf{state} = \mathsf{accepted}$ then $\mathcal{C}^0$ replies with $(\mathsf{accepted}, \pi.\mathsf{key})$, however if $\pi.\mathsf{state} = \mathsf{rejected}$ then $\mathcal{C}^0$ replies with $(\mathsf{rejected}, \perp)$.

If π is a CASE 2-session then $\mathcal{C}^0$ first checks whether π received all messages to complete the protocol ($\mathcal{C}^0$ knows this since it observes the messages passed to $\mathcal{SIM}$). If not, $\mathcal{C}^0$ ignores the query. Otherwise, let $\mathsf{Handshake}(\mathsf{id}, G.\mathsf{par}, r)$ be the query which invoked π. $\mathcal{C}^0$ checks whether there exists a session π' which is *compatible* to π. If no such session π' exists then $\mathcal{C}^0$ replies with $(\mathsf{rejected}, \perp)$. Otherwise, $\mathcal{C}^0$ replies with $(\mathsf{accepted}, \pi.\mathsf{key})$ according to the following rules: If $\pi.\mathsf{key}$ is not set but $\pi'.\mathsf{key}$ is then $\pi.\mathsf{key} \leftarrow \pi'.\mathsf{key}$. If both $\pi.\mathsf{key}$ and $\pi'.\mathsf{key}$ are not set then $\pi.\mathsf{key} \in_R \{0,1\}^\kappa$ is chosen randomly.

Compatibility of sessions π and π' means that the corresponding members id and id' satisfy all requirements for the acceptance in the handshake protocol. In this case, it is clear that, by revealing the session key, $\mathcal{A}$ will learn that id and id' belong to the same group. As noticed in [15], this is unavoidable and, even in this case, $\mathcal{A}$ is not supposed to learn the affiliation of these members. Now we are ready to formally define LAH-security.

Definition 7 (LAH-Security). *Let* $\mathsf{LAH\text{-}AKE} = \{\mathsf{CreateGroup}, \mathsf{AddUser}, \mathsf{Handshake}, \mathsf{Revoke}\}$*, b a randomly chosen bit, and* $\mathcal{Q} = \{\mathsf{CreateGroup}, \mathsf{AddUserU}, \mathsf{AddUserG}, \mathsf{Handshake}, \mathsf{Send}, \mathsf{Corrupt}, \mathsf{Reveal}, \mathsf{Revoke}\}$*. Let* $\mathsf{Game}^{\mathsf{lah},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ *denote the interaction of $\mathcal{A}^{\mathcal{Q}}(1^\kappa)$ with the challenger $\mathcal{C}^b$ via queries until $\mathcal{A}^{\mathcal{Q}}$ outputs bit b' which is the output of the game.*

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

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

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 $\mathcal{A}$, given id_b for $b \in_R \{0,1\}$, to identify user U_b.

Definition 8 (Untraceability). *Let* LAH-AKE $= \{$CreateGroup, AddUser, Handshake, Revoke$\}$*, b a randomly chosen bit, and $\mathcal{Q} = \{$*CreateGroup, AddUserU, AddUserG, Handshake, Send, Reveal, Corrupt, 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 such that the corresponding pseudonyms* id_0 *and* id_1 *are generated. Note that, during this process, the protocol sessions on behalf of G (*AddUserG*) 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 (*AddUserU*) 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* LAH-AKE *is* untraceable *if this advantage is negligible (in κ).*

In game $\mathsf{Game}^{\mathsf{trace},b}_{\mathcal{A},\mathsf{LAH\text{-}AKE}}(\kappa)$ the corruption of id_b is not forbidden. Therefore, untraceable LAH-AKE schemes hide the real identity of group members even if their membership credentials are leaked.

Revocation and its Relationship to Untraceability and Affiliation-Hiding. Our model raises some concerns about the relationship between the revocation procedure and the untraceability property. We notice that in linkable AH-AKE and SH protocols such as [15, 8], where GA learns the pseudonym id of a user U during the registration phase, revocation can be understood in two ways: The first approach is what we call *revocation of users*, i.e., GA may want to revoke some particular user U. The second approach is what we call *revocation of pseudonyms*, i.e., GA may want to revoke some pseudonym id. In the trusted GA model [15], i.e. without untraceability, there is no difference between these two approaches, since GA knows the mapping between U and its id, and can add id in both cases to $G.\mathsf{prl}$. In contrast, our model with untraceability ensures that, during the registration of U, GA does not get any information about the pseudonym id. Therefore, revocation of users is no longer possible. However, users participate in group applications via pseudonyms. Therefore, if some misbehavior is noticed, the responsible pseudonym can be identified and revoked. This type of revocation is still meaningful, since, if GA revokes some pseudonym id that is owned by some user U, then U cannot communicate in that group anymore, unless it obtains a new pseudonym. To do so, U would have to

re-register to the same group, which might be forbidden by the admission policy of the GA.

3 LAH-AKE Protocol Secure against Malicious GAs

We now describe our LAH-AKE scheme that provides security against malicious GAs. Our construction is based on the scheme from [15]. The modifications apply to the generation of group parameters, the registration process, and the actual key exchange protocol. Revocation of pseudonyms is handled via revocation lists similar to [15].

3.1 Number-Theoretic Assumptions and Building Blocks

Definition 9 (RSA Assumption on Safe Moduli). *Let* $\mathsf{RSA\text{-}G}(\kappa')$ *be a probabilistic algorithm that outputs pairs* (n, e) *where (a)* $n = pq$ *for random* κ'*-bit (safe) primes* $p \neq q$*, (b)* $p = 2p'+1, q = 2q'+1$ *for primes* p', q'*, and (c)* $e \in \mathbb{Z}_{\varphi(n)}$ *is coprime to* $\varphi(n)$*. The RSA-success probability of a PPT solver* $\mathcal{A}$ *is defined as*

$$\mathsf{Succ}^{\mathsf{rsa}}_{\mathcal{A}}(\kappa') := \Pr\big[(n,e) \leftarrow \mathsf{RSA\text{-}G}(\kappa');\ z \leftarrow_R \mathbb{Z}_n^*;\ m \leftarrow \mathcal{A}(n,e,z)\ \textit{with}\ m^e = z\big].$$

The RSA assumption on safe moduli *states that the maximum RSA-success probability* $\mathsf{Succ}^{\mathsf{rsa}}(\kappa')$ *(defined over all PPT solvers* $\mathcal{A}$*) is negligible in* κ'*.*

Definition 10 (CDH Assumption in $QR(p)$). *Let* $QR(p)$ *denote the group of quadratic residues modulo a safe prime* $p = 2p'+1$ *of length* κ'*. The CDH-success probability of a PPT adversary* $\mathcal{A}$ *is defined as*

$$\mathsf{Succ}^{\mathsf{cdh}}_{\mathcal{A}}(\kappa') := \max_{\substack{\text{safe prime } p \\ |p|=\kappa', \langle g\rangle = QR(p)}} \Pr\big[x, y \leftarrow_R \mathbb{Z}_{p'}; h \leftarrow \mathcal{A}(p, g, g^x, g^y)\ \textit{with}\ h = g^{xy}\big].$$

The CDH assumption in $QR(p)$ *states that the maximum CDH-success probability* $\mathsf{Succ}^{\mathsf{cdh}}(\kappa')$ *(defined over all PPT solvers* $\mathcal{A}$*) is negligible in* κ'*.*

Our scheme uses the following additional building blocks. Let κ, κ' be security parameters. We use cryptographic hash functions modeled as random oracles: $H_1 : \{0,1\}^* \to \{0,1\}^{2\kappa}$ and, for any $n \in \mathbb{N}$ with $|n| = 2\kappa'$, a specific hash function $H_n : \{0,1\}^* \to \mathbb{Z}_n$. Note that H_n can be constructed as $H_n(x) := H(n \,\|\, x) \bmod n$ using some hash function $H : \{0,1\}^* \to \{0,1\}^{2\kappa'+\kappa}$. By $\Sigma := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ we denote a digital signature scheme which is assumed to be existentially unforgeable under chosen message attacks (EUF-CMA).

Camenisch and Michels [6] show how to prove in zero-knowledge (ZK) the correct generation of an RSA modulus $n = pq$ for some safe primes p and q, including the necessary primality tests and without revealing any further information about the factors. We use an extended version of these ZK proofs: In Appendix A we show how to additionally prove in ZK that an element $g \in \mathbb{Z}_n^*$ has maximum order in $\mathbb{Z}_n^*$.

3.2 New LAH-AKE Scheme

We now proceed with the description of algorithms and protocols.

Algorithm CreateGroup(1^κ). This algorithm generates parameters for a new group G as follows: it picks two κ'-bit primes p, q with $p = 2p' + 1$ and $q = 2q' + 1$ for prime numbers p' and q', sets $n = pq$, picks an exponent $e \in \mathbb{Z}_{\varphi(n)}$ which is coprime to $\varphi(n) = (p-1)(q-1) = 4p'q'$, and computes $d = e^{-1} \pmod{\varphi(n)}$. Note that n is a Blum integer, i.e., $p \equiv q \equiv 3 \pmod 4$.

As $\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ the largest element order in $\mathbb{Z}_n^*$ is $\operatorname{lcm}(\varphi(p), \varphi(q)) = 2p'q' = \varphi(n)/2$, and hence $\mathbb{Z}_n^*$ is not cyclic. For elements $g \in \mathbb{Z}_n^*$ with $\operatorname{ord}(g) \geq p'q'$ and $g^{p'q'} \neq \pm 1$ it follows that $\operatorname{ord}(g) = 2p'q'$ and that $-1 \notin \langle g \rangle$. In this case, we have $\mathbb{Z}_n^* \cong \langle -1 \rangle \times \langle g \rangle$. Let the CreateGroup algorithm pick such an element g. Since about a half of the elements in $\mathbb{Z}_n^*$ has the desired properties [15], g can easily be found by just random sampling and testing.

Our security model treats GAs as untrusted parties. This even includes lack of trust in the honest generation of group parameters. Appendix A sketches a technique based on [6] that constructs a ZK proof for (n, g) showing that n is a safe RSA modulus and that $\mathbb{Z}_n^* = \langle -1 \rangle \times \langle g \rangle$. By $\Pi_{n,g}$ we denote its non-interactive version that can be obtained via the classical Fiat-Shamir transformation [11].

Finally, the algorithm sets $G.\mathsf{prl} = \emptyset$ and outputs $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl}, \Pi_{n,g})$ with $G.\mathsf{pk} = (n, e, g)$ and the private key $G.\mathsf{sk} = d$.

Protocol AddUser(U, G). Member admission is implemented using a protocol between U and GA, as specified in Figure 1. Communication between U and GA is assumed to be authentic, yet it does not need to be confidential as in [15], mainly because membership credentials in our registration process are not transported from GA to U but computed through interaction, which need not be private. Some details follow. It is assumed that U obtains public group parameters $G.\mathsf{par} = (G.\mathsf{pk}, G.\mathsf{prl}, \Pi_{n,g})$ prior to the protocol execution. Then, in a first step, U examines the validity of the group parameters (n, e, g) by checking the NIZK proof $\Pi_{n,g}$. Then, U generates a signature key pair $(pk, sk) \leftarrow \Sigma.\mathsf{KGen}(\kappa)$. The verification key pk is hereafter used by U as its pseudonym id in group G, i.e., we set $\mathsf{id} := pk$. Using the standard blind RSA signature scheme [9, 3] U obtains an RSA signature $\sigma_{\mathsf{id}} = H_n(\mathsf{id})^d$ on $H_n(\mathsf{id})$, as depicted in Figure 1 (all computations are $\bmod n$). Note that the blinding factor r^e effectively hides id and $H_n(\mathsf{id})$ from G. The output of U is $(\mathsf{id}, \mathsf{id.cred})$ with $\mathsf{id.cred} = (\sigma_{\mathsf{id}}, sk)$.

Protocol Handshake($(\mathsf{id}_A, \mathsf{id}_A.\mathsf{cred}, G_A, \mathsf{init}), (\mathsf{id}_B, \mathsf{id}_B.\mathsf{cred}, G_B, \mathsf{resp})$)
The handshake protocol is executed between two users, say A and B, holding pseudonyms id_A and id_B, public group keys $G_A.\mathsf{pk} = (n_A, e_A, g_A)$ and $G_B.\mathsf{pk} = (n_B, e_B, g_B)$, and credentials $\mathsf{id}_A.\mathsf{cred} = (\sigma_{\mathsf{id}_A}, sk_A)$ and $\mathsf{id}_B.\mathsf{cred} = (\sigma_{\mathsf{id}_B}, sk_B)$, respectively. The protocol is specified in Figure 2. We assume that all computations are performed $\bmod n_A$ on the left and $\bmod n_B$ on the right, except the assignments containing the padding function pad (line 5). The aim of

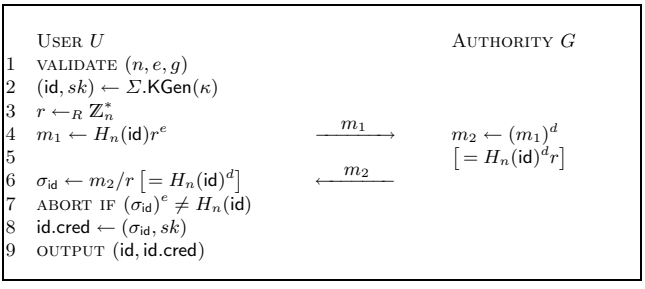

Fig. 1. Specification of $\mathsf{AddUser}(U, G)$

the latter is to hide the moduli n_A and n_B from observers. The technique dates back to [10]. A leakage could allow $\mathcal{A}$ to make conclusions about the players' affiliations. The padding function pad is a probabilistic mapping that transforms $\theta' \in \mathbb{Z}_n$ into an integer θ in the interval $[0, 2^{2\kappa'+\kappa} - 1]$ by choosing a random $k \leftarrow_R [0, \lfloor 2^{2\kappa'+\kappa}/n \rfloor - 1]$ and returning $\theta \leftarrow \theta' + kn$. Note that $\theta \equiv \theta' \bmod n$ and that $\{(-1)^b g^t \mid (b, t) \in \{0, 1\} \times \mathbb{Z}_{n/2}\} = \mathbb{Z}_n$.

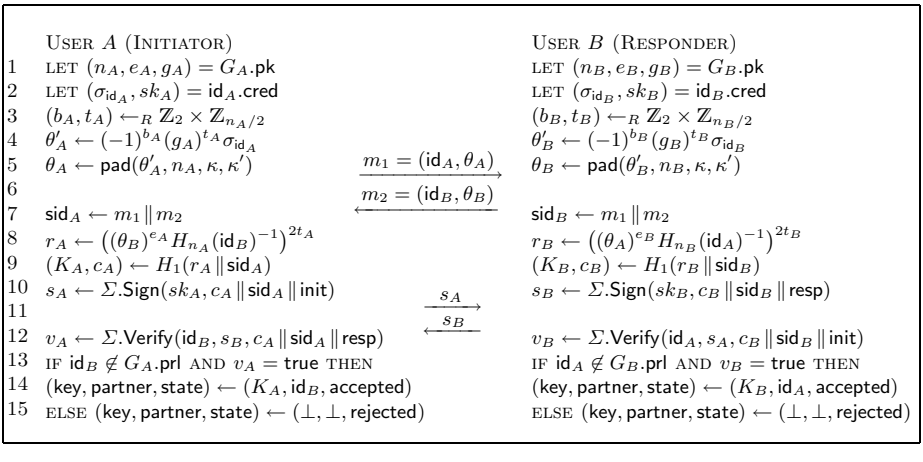

Fig. 2. Specification of $\mathsf{Handshake}((\mathsf{id}_A, \mathsf{id}_A.\mathsf{cred}, G_A, \mathsf{init}), (\mathsf{id}_B, \mathsf{id}_B.\mathsf{cred}, G_B, \mathsf{resp}))$

Remark 1. There is an important difference between our Handshake protocol and that from [15]: The key confirmation message sent in the second round of our protocol is a signature; its validity confirms not only the equality of the session key but also serves as a proof for the ownership of claimed id by the participant (i.e. through the knowledge of the corresponding secret key). This way we thwart active impersonation attacks where $\mathcal{A}$ exploits the blinding feature of the AddUser protocol and obtains credentials for ids of honest users. Note that the possibility of such pseudonym impersonation by insiders (group members) would

violate AKE security as defined in Section 2.2 (see part (c) of Definition 4). In contrast, the handshake protocol in [15] computes key confirmation messages as hash values computed using the established session key.

Algorithm Revoke($G.\mathsf{sk}, G.\mathsf{prl}, \mathsf{id}$). Revocation of pseudonyms is handled by the GA of G by placing the pseudonym into the corresponding pseudonym revocation list $G.\mathsf{prl}$. We assume that this list is distributed using authenticated channels.

4 Security and Efficiency

Correctness of the described LAH-AKE protocol follows by inspection. Note that the intermediate values r_A and r_B have the form

$$\begin{aligned} r_A &= \big((\theta'_B)^{2e_A} H_n(\mathsf{id}_B)^{-2}\big)^{t_A} = \big((g_B)^{2e_A t_B} (\sigma_{\mathsf{id}_B})^{2e_A} H_n(\mathsf{id}_B)^{-2}\big)^{t_A} \\ &= \big((g_B)^{2e_A t_B} H_n(\mathsf{id}_B)^{2e_A d_A} H_n(\mathsf{id}_B)^{-2}\big)^{t_A} = (g_B)^{2e_A t_A t_B} \qquad (\mathrm{mod}\, n_A) \end{aligned}$$

and, analogously, $r_B = (g_A)^{2e_B t_B t_A}$. Presuming that $(n_A, e_A, g_A) = (n_B, e_B, g_B)$, i.e. that users A and B are members of the same group, r_A and r_B evaluate to the same value, and $K_A = K_B$ follows.

Remark 2. Protocol correctness requires that the $H_n(\mathsf{id})$ values are indeed invertible $\mathrm{mod}\, n$. In fact, this is not the case for $H_n(\mathsf{id}) \in \mathbb{Z}_n \setminus \mathbb{Z}_n^*$ and then the protocol will fail. However, this occurs with negligible probability. We stress that this remark also applies to the original protocol in [15].

We now state that our LAH-AKE construction satisfies the AKE- and LAH-security and untraceability goals defined in Section 2.2. The corresponding proofs[2] (with estimated attack probabilities) are provided in the full version of this paper.

Theorem 1 (AKE-Security). *Our* LAH-AKE *scheme is AKE-secure (Def. 5) in the random oracle model under the RSA (Def. 9) and CDH (Def. 10) assumptions if $\Pi_{n,g}$ is sound and zero-knowledge, and Σ is EUF-CMA secure.*

Theorem 2 (LAH-Security). *Our* LAH-AKE *scheme is LAH-secure (Def. 7) in the random oracle model under the RSA (Def. 9) and CDH (Def. 10) assumptions if $\Pi_{n,g}$ is sound and zero-knowledge, and Σ is EUF-CMA secure.*

Theorem 3 (Untraceability). *Our* LAH-AKE *scheme is untraceable (Def. 8) in the random oracle model if $\Pi_{n,g}$ is sound.*

[2] It might initially seem, that, due to the utilization of blind signatures in the AddUser protocol, security of LAH-AKE cannot be shown without relying on the hardness of some One-More RSA-Inversion problem [3]. However, careful examination of the constraints for session freshness in Definition 4 shows that the AddUserG query (i.e. the adversary's access to the blind signature oracle) is available only in cases where the corresponding GA may be corrupted anyway. Hence, the RSA assumption suffices to prove the protocol's security.

Efficiency. The cost of our handshake protocol is dominated by the computations of $\theta'_{A/B}$, $r_{A/B}$, generation of $s_{A/B}$ and verification of $s_{B/A}$. The first two involve exponentiations (of size $\log n = 2\kappa'$) and the cost of the last two depends on the balance between $\Sigma.\mathsf{Sign}$ and $\Sigma.\mathsf{Verify}$. Many current signature schemes involve either low verification and high generation costs (e.g, RSA) or vice versa (e.g., DSA). In any case, suffice it to say that, for each participant, the overall computation cost amounts to approximately 3 full-blown exponentiations. Considering the high degree of security offered by our scheme, the overhead is very low.

The NIZK proof $\Pi_{n,g}$ in the $\mathsf{AddUser}$ protocol is the most expensive operation. In fact, the verifier would have to compute about $24\kappa t \log n$ (multi-) exponentiations, where 2^{-t} is the error-probability for the primality tests (see [6], Sections 4.3 and 5.1). Note that [6] suggests two optimizations on the protocol: the first one in [6, Section 5.2] that effectively removes factor t from the above equation; and the second one [6, Section 2.2] that is applicable only to interactive ZK proofs and eliminates factor κ. Nevertheless, the complexity for verifying the correctness of the group parameters remains relatively high. However, this proof is necessary (in theory) for the security in our model. In *practice*, it is conceivable to completely omit the verification of $\Pi_{n,g}$, since the set of public parameters of a group is fixed once, upon the initialization. Therefore, its verification by a single trusted auditing authority would suffice. An appropriate (weaker) security model is easily derived from that given in Section 2.2 by modifying the $\mathsf{AddUserU}$ query such that only group parameters $G.\mathsf{par}$ are accepted that were established by a $\mathsf{CreateGroup}$ query before. Note that, in this relaxed model, the untraceability of our $\mathsf{LAH\text{-}AKE}$ scheme becomes *unconditional* (since there is no longer any need to assume soundness of $\Pi_{n,g}$ in Theorem 3).

5 Conclusion

AH-AKE protocols are powerful privacy-preserving authentication mechanisms usable in various collaborative and group-based applications, such as p2p systems, mobile ad-hoc groups, and social networks. Prior protocols require a high degree of trust in the GA. In this work, we considered untrusted GAs in linkable AH-AKE protocols. We developed a model containing meaningful definitions of security and designed a concrete protocol for mitigating GA misbehavior, guaranteeing the valuable privacy goals for the users.

Acknowledgement

The authors wish to thank Marc Joye for the discussion about the hardness of the CDH assumption in the safe RSA setting, and Andreas Peter for general discussions on number-theoretic assumptions used in this paper and relationships among them.

References

1. Ateniese, G., Kirsch, J., Blanton, M.: Secret Handshakes with Dynamic and Fuzzy Matching. In: Network and Distributed System Security Symposium, NDSS 2007 (2007)
2. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret Handshakes from Pairing-Based Key Agreements. In: IEEE Symposium on Security and Privacy 2003, pp. 180–196. IEEE CS, Los Alamitos (2003)
3. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 309–328. Springer, Heidelberg (2002)
4. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: 1st ACM Conference on Computer and Communications Security (CCS 1993), pp. 62–73. ACM, New York (1993)
5. Boneh, D.: The Decision Diffie-Hellman Problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
6. Camenisch, J., Michels, M.: Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
7. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
8. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret Handshakes from CA-Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
9. Chaum, D.: Blind Signatures for Untraceable Payments. In: CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)
10. Desmedt, Y.: Securing Traceability of Ciphertexts — Towards a Secure Software Key Escrow System (Extended Abstract). In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 147–157. Springer, Heidelberg (1995)
11. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
12. Gennaro, R., Rabin, T., Jarecki, S., Krawczyk, H.: Robust and Efficient Sharing of RSA Functions. J. Cryptology 20(3), 393 (2007)
13. Gennaro, R., Rabin, T., Krawczyk, H.: RSA-Based Undeniable Signatures. J. Cryptology 20(3), 394 (2007)
14. Jarecki, S., Kim, J., Tsudik, G.: Group Secret Handshakes or Affiliation-Hiding Authenticated Group Key Agreement. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 287–308. Springer, Heidelberg (2006)
15. Jarecki, S., Kim, J., Tsudik, G.: Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
16. Jarecki, S., Liu, X.: Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
17. Jarecki, S., Liu, X.: Private Mutual Authentication and Conditional Oblivious Transfer. In: Halevi, S. (ed.) Advances in Cryptology — CRYPTO 2009. LNCS, vol. 5677, pp. 90–107. Springer, Heidelberg (2009)

18. Kawai, Y., Yoneyama, K., Ohta, K.: Secret Handshake: Strong Anonymity Definition and Construction. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 219–229. Springer, Heidelberg (2009)
19. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger Security of Authenticated Key Exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
20. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Tsudik, G., Xu, S.: A Flexible Framework for Secret Handshakes. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 295–315. Springer, Heidelberg (2006)
22. Vergnaud, D.: RSA-Based Secret Handshakes. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 252–274. Springer, Heidelberg (2006)
23. Xu, S., Yung, M.: k-Anonymous Secret Handshakes with Reusable Credentials. In: 11th ACM Conference on Computer and Communications Security (CCS 2004), pp. 158–167. ACM, New York (2004)

A About the Construction of $\Pi_{n,g}$

Camenisch and Michels [6] describe a cryptographic protocol where a prover P holding the factorization $n = pq$ of a public safe RSA modulus n proves in Zero-Knowledge to a verifier V that n is indeed a safe RSA modulus. This is done by sending randomized Pedersen commitments [20] $C(p), C(q), \ldots$ of p, q and some intermediate variables to V, and by proving in ZK arithmetical relationships between them. Within others, probabilistic primality tests for p and q are run and ZK-verified step by step by V. In the full version of this paper we describe the deployed techniques in more detail.

The protocol given in [6] is not directly applicable to the CreateGroup/AddUser algorithms of our AH-AKE scheme (see Section 3.2) as not only the fact that n is a safe RSA modulus has to be proven, but in addition also the equality $\mathbb{Z}_n^* = \langle -1 \rangle \times \langle g \rangle$. A necessary condition for the latter is that $\mathrm{ord}(g) = \varphi(n)/2 = 2p'q'$ and $g^{p'q'} \neq \pm 1$. As n is a Blum integer there are exactly four elements $a \in \mathbb{Z}_n^*$ with $a^2 = 1$, namely ± 1 and $\pm\omega$ for some $\omega \in \mathbb{Z}_n^*$. It is hence sufficient if P proves to V that $g^{p'q'} = \pm\omega$. In the following, we sketch how this can be done.

The Euclidean algorithm finds integers x, y satisfying $px + qy = 1$. For $\omega = px - qy \pmod{n}$ the Chinese Remainder Theorem shows that $\omega^2 = 1 \pmod{n}$. In a first step, P publishes commitments $C(x), C(y)$ for x and y, respectively, and proves (in ZK) $C(1) = C(p) \cdot C(x) + C(q) \cdot C(y)$. It then computes ω, publishes $C(\omega)$ and proves $C(\omega) = C(p) \cdot C(x) - C(q) \cdot C(y)$. After computing $g' = g^{p'q'}$, it publishes $C(g')$ and proves $C(g') = C(g)^{C(p') \cdot C(q')}$, concluding the proof by revealing either $C(g') = C(\omega)$ or $C(g') = C(-\omega)$.