Beyond Secret Handshakes: Affiliation-Hiding Authenticated Key Exchange

Stanisław Jarecki, Jihye Kim, and Gene Tsudik

Computer Science Department
University of California, Irvine
{stasio, jihyek, gts}@ics.uci.edu

Abstract. Public key based authentication and key exchange protocols are not usually designed with privacy in mind and thus involve cleartext exchanges of identities and certificates before actual authentication. In contrast, an Affiliation-Hiding Authentication Protocol, also called a *Secret Handshake*, allows two parties with certificates issued by the same organization to authenticate each other in a *private* way. Namely, one party can prove to the other that it has a valid organizational certificate, yet this proof hides the identity of the issuing organization unless the other party also has a valid certificate from the same organization.
We consider a very strong notion of Secret Handshakes, namely *Affiliation-Hiding Authenticated Key Exchange* protocols (AH-AKE), which guarantee security under arbitrary composition of protocol sessions, including man-in-the-middle attacks. The contribution of our paper is three-fold: First, we extend existing notions of AH-AKE security to Perfect Forward Secrecy (PFS), which guarantees session security even if its participants are later corrupted or any other sessions are compromised. Second, in parallel to PFS security, we specify the exact level of privacy protection, which we call *Linkable Affiliation-Hiding* (LAH), that an AH-AKE protocol can provide in the face of player corruptions and session compromises. Third, we show an AH-AKE protocol that achieves both PFS and LAH properties, under the RSA assumption in ROM, at minimal costs of 3 communication rounds and two (multi)exponentiations per player.

Keywords: secret handshakes, authenticated key exchange, privacy, privacy-preserving authentication.

1 Introduction

Affiliation-Hiding Authentication protocols, also known as *Secret Handshakes* (SH) [BDS+03], allow two members of the same group to authenticate each other in a way that hides their affiliation from all others. For example, two FBI agents, Alice and Bob, want to discover and communicate with other agents, but they don't want to reveal their affiliations to non-agents. Since the environment is potentially hostile, Alice wants to authenticate herself to Bob *only if* Bob is another FBI agent, and vice versa for Bob. Affiliation-hiding authentication scheme ensures that if only one of the two is a genuine agent, the other (the impostor) learns nothing about the counterpart's affiliation. More generally, a non-member adversary who stages an active attack against group members (even playing a man in the middle) should not determine whether any of the parties he interacts with is a member of the targeted group.

Prior Work on Secret Handshakes. Affiliation-hiding authentication schemes were introduced as *Secret Handshakes* by Balfanz et al. [BDS+03], together with a construction based on the security of the Bilinear Diffie-Hellman (BDH) problem in a group with a bilinear map. Subsequently, Castelluccia, et al. [CJT04] constructed a more efficient scheme secure under the Computational Diffie Hellman (CDH) assumption. (Recently [D.V05] proposed an RSA-based SH scheme, but the scheme fails to provide affiliation-hiding.[1]) However, the schemes of [BDS+03] and [CJT04] are only entity authentication schemes, and not authenticated key agreements (AKE). Secondly, both these papers consider only a weak notion of (affiliation-hiding) authentication scheme, which looks only at security of isolated protocol instances. In particular, their model excludes man in the middle attacks, and indeed their schemes are insecure against them.

This restricted notion of affiliation-hiding authentication schemes was strengthened to *Affiliation-Hiding Authenticated Key Exchange* protocols (AH-AKE) in [JKT07] and [JL07]. In [JKT07] this notion was defined for *group* key agreement protocols, which generalize two-party AKE's. In [JL07] the notion was strengthened to AKE's which are both affiliation-hiding and *unlinkable* (see a more on unlinkability below), but this in particular implies a two-party affiliation-hiding AKE protocol. The notion of affiliation-hiding authenticated key exchange strengthens the notion of (affiliation-hiding) entity authentication considered in [BDS+03,CJT04] in two ways: First, an authenticated key agreement is a more useful protocol tool because it outputs an authenticated key which can be used for any secure communication task, including entity authentication. Second, the AH-AKE notion of [JKT07,JL07] satisfies the standard requirements demanded of AKE protocols, as formalized by Bellare, Canetti, Krawczyk [MRH01,CK01] and Shoup [V.S99] (but without perfect forward secrecy). Essentially, each protocol session remains secure even if all protocol sessions are arbitrarily scheduled by the adversary, and even if the adversary compromises the key on any *other* protocol session. In particular, this implies security against a man in the middle attack. However, both [JKT07] and [JL07] consider only a simplified notion of *privacy* for AH-AKE protocols, where the adversary can arbitrarily interleave protocol sessions, but it cannot compromise any of them. Consequently, it is unclear how *any* information about the agreed session keys affects the privacy protection offered by such schemes. Note that in most applications even a passive adversary learns whether two sessions have succeeded, and produced the same

[1] Providing affiliation-privacy in RSA-based protocols requires extra care, because one must prevent any correlation of protocol messages with the RSA modulus n which is a part of the public key of a given group. The proposal for an RSA-based secret handshake scheme, [D.V05], based on the OSBE scheme of [LDB03], fails to achieve affiliation-hiding because it tries to prevent such correlation by obfuscating only the *size* of the protocol messages (and hence, in their intention, the RSA modulus), leaving intact other possibilities of correlation. In fact, instances of the protocol of [D.V05] can be correlated with the group public key by computing the Jacobian symbol of several protocol messages: If the protocol instance involves the given modulus n, the corresponding Jacobian symbols are related in predictable ways, thus providing a test whether the protocol session involves players affiliated with a group whose public key is the given modulus n. Our RSA-based protocol fixes that problem by requiring a safe RSA modulus n, and making sure that all the exchanges messages are random elements in the group Z_n^*, before applying a simple method that masks the modulus size.

session key, and indeed this information reveals something about the affiliations of the two interacting parties, namely that they are the same.

Efficiency-wise, the 2-party AH-AKE protocols implied by the (affiliation-hiding) *group* key agreement protocols of [JKT07] take three rounds, involve three exponentiations, and work under either RSA or CDH assumptions in ROM. The 2-party AH-AKE protocol implied by the affiliation-hiding *and unlinkable* AKE scheme of [JL07] works only assuming that the revocation lists of the two interacting players are no farther than some constant Δ apart, moreover the protocol requires $O(\Delta * \log n)$ exponentiations where n is the upper bound on the number of players affiliated with a single organization.

Several other papers appeared on secret handshakes [TX06,XY04], but they do not address security and privacy of affiliation-hiding AKE protocols under arbitrary protocol composition.

Our Contributions: **First,** We strengthen the AH-AKE security notion of [JKT07,JL07] to include *Perfect Forward Secrecy* (PFS), which ensures that each session remains secure even if its participants are eventually corrupted, revealing all their long-term secrets to the adversary. Note that since the adversary in the AH-AKE models of [JKT07,JL07] is not allowed to compromise any group members future corruption of the long-term secrets of any of them may endanger the secrecy of previous protocol sessions in which this group member was a participant. We upgrade the AH-AKE security definition to a more rubust and useful notion by modelling corruptions in the security model.

Second, in parallel to PFS security, we formalize the exact level of privacy protection, which we call *Linkable Affiliation-Hiding* (LAH), that an AH-AKE protocol might provide in the face of player corruptions and session compromises. Intuitively, a linkable affiliation-hiding AKE protocol can reveal only as much information about affiliations of the participating parties as is revealed in the following idealized process: The process assigns a random "pseudonym" value to each certificate in the system. Denote a value assigned to certificate $\mathsf{cert}_i^{(j)}$ which user P_i holds for group G_j as $id_{i,j}$. Every time player P_i runs an AH-AKE protocol using the public key of G_j, the session reveals the pseudonym $id_{i,j}$. Every time an adversary compromises a session, it learns only if this session failed or succeeded, and consequently it learns whether the pseudonyms $id_{i,j}$ and $id_{i',j'}$ of the two players participating in this session correspond to the same group (and thus it learns whether P_i and P_j have a shared affiliation), because otherwise these sessions are not supposed to succeed. Finally, whenever some player P_i is corrupted, the process reveals, for each pseudonym $id_{i,j}$ of player P_i, which group G_j this pseudonym corresponds to. We stress that the AH-AKE privacy models of [JKT07,JL07] modelled only the information which the adversary learns from protocol sessions themselves, and not the information learned from subsequent compromises of session keys and/or corruptions of their participants.

Third, we show an optimal-cost AH-AKE protocol, which satisfies our strengthened security definitions, PFS and LAH, in the Random Oracle Model (ROM) under the RSA assumption. The protocol takes 3 rounds, and it is *implicitly authenticated* versions of the Diffie-Hellman Key Exchange protocol. The cost of the protocol appears minimal because its computation costs are very similar to the cost of the unauthenticated Diffie-Hellman key exchange, namely one (off-line) exponentiation and one (on-line) multi-

exponentiation per participant. Moreover, three rounds of interaction again matches the round complexity of non-private PFS AKE protocols.

Linkability Disclaimer. As the name suggests, the privacy guaranteed by our notion of *linkable affiliation-hiding* does not include unlinkability, and in this aspect the new notion is similar to the (weaker) notions of affiliation-hiding considered in [BDS+03,CJT04,JKT07]. Indeed, since the ideal process we use to define the LAH privacy property reveals the same pseudonym every time a player uses the same certificate in an AH-AKE session, the adversary can potentially link two instances of the same player. Note, however, that that these pseudonyms do not leak the *affiliation* of this player, except if a player is corrupted (which can reveal a link between a pseudonym and a group) or when a session is compromised (which can reveal that two pseudonyms correspond to the same group). We stress that even though unlinkable and affiliation-hiding schemes exist [TX06,JL07], they have severe limitations (synchronization in revocation lists, expensive operation), while affiliation-hiding protocols, as we show here, can be achieved at seemingly minimal expense, and thus it is important to understand the exact privacy guarantees offered by the "merely" affiliation-hiding (but linkable) authentication protocols.

Moreover, it is worth pointing out that while our security and privacy models assume that every user has only a single certificate from any group, in which case any two instances involving the same group member are linkable, there are various heuristics which can ameliorate this issue in practice. For example, heuristic unlinkability can be achieved by users rotating through a small set of certificates, by setting strict time limits on usage of each certificate, or by associating different certificates with different locations or aspects of user's activity.

Organization. In Section 2 we define AH-AKE protocols with perfect-forward secrecy (PFS) and linkable affiliation-hiding (LAH), and show that linkable affiliation-hiding implies perfect forward secrecy. In section 3 we show an AH-AKE which satisfies the LAH property (and hence the PFS property), based on the RSA assumption and the proof of security of the protocol is given in detail.

2 Affiliation-Hiding Authenticated Key Exchange Protocols

Entities. Our AH-AKE model is based on the existing models for standard (i.e. non affiliation-hiding) authenticated key exchange protocols, e.g. [BPR00,CK01]. The main difference is that the standard model assumes a global PKI where each entity has a private/public key-pair and a certificate issued by a CA which is part of the PKI. The PKI involves a *certification hierarchy*, where the integrity of the association between entities and their public keys is vouched by a chain of certificates all leading to some commonly trusted CA-s. In this model, it is assumed that certificates (which in many applications contain information about owners' *affiliation*) are publicly available. In contrast, AH-AKE protocols aim to protect affiliation privacy of the participants and certificates are kept private. Another distinctive feature of our model is its "flat" certification structure, i.e., certification hierarchies and chains are not allowed. There are only CA-s and entities certified by CA-s; there are no intermediate CA-s and no delegation of certificates.

An AH-AKE *scheme* operates in an environment that includes a set of *users* $\mathcal{U}$ and a set of *groups* $\mathcal{G}$. Each group is administered by a CA responsible for creating the group, admitting entities as members and revoking membership. We assume upper bounds m and n, respectively, on the total number of groups and the number of members in any given group, i.e., $|\mathcal{G}| \leq m$ and $|\mathcal{U}| \leq n$. We assume that each user can be a member of many groups. We denote the fact that user $U \in \mathcal{U}$ is a member of group $G \in \mathcal{G}$ as $U \prec G$.

AH-AKE Protocol. The main part of an AH-AKE scheme is an AH-AKE *protocol*, which is executed by any pair of users. Player U_i, participating in an instance of the AH-AKE protocol executes the protocol instructions on inputs a public key of some group $G \in \mathcal{G}$ s.t. $U_i \prec G$, and U_i's certificate of membership in G. The purpose of the AH-AKE protocol is for a pair of players to establish an authenticated shared secret key as long as (1) both run the protocol on the public key of the same group G, and (2) it holds that $U_i \prec G$ and $U_j \prec G$.

To avoid any misunderstanding, we stress that such protocol does not in general imply an efficient solution for an (affiliation-hiding) *group discovery* problem, where two each player start a protocol on a *set* of its certificates, and the protocol succeeds, for example, as long as the sets contributed by the two players have a non-empty union. In contrast, our AH-AKE schemes are most practical in scenarios where each user is a member of at most one group. However, we stress that if a user is a member of many groups, this would affect execution efficiency, but it would not affect the *security* and the *affiliation-hiding* of our schemes. While the *protocols* we give are efficient only if each player is always a member of at most a few groups, the *security definitions* stated below without loss of generality assume that each user is a member of every group.

Public Information and Network Assumptions. We assume that all groups $G \in \mathcal{G}$ are publicly known. Their CA public keys and certificate revocation lists (CRL-s) maintained by CA-s are publicly accessible. Before any group can be created, a common security parameter must be publicly chosen, and a public Setup procedure is executed on that parameter. The Setup procedure creates common cryptographic parameters which are used as inputs in all subsequent protocols. We stress that the Setup procedure does not need to be executed by a trusted authority: It can be executed by anyone, for example by one of the CAs, and everyone can verify the validity of its outputs.

We assume that communication between users and CA-s, i.e. the certificate issuance process and the CRL retrieval, are conducted over anonymous and authenticated channels. In practice, a user might communicate with the CA, e.g., while retrieving the most recent CRL for its group, over an anonymous channel such as TOR [DMS04]. Alternatively, the CRL-s of all groups can be combined and stored at some highly-available site where they can be either retrieved in bulk (if small) or via a Private Information Retrieval (PIR) protocol, e.g., [CKGS98].

We assume that all communication within the AH-AKE protocol takes place over an unauthenticated channel. In our model, the adversary is assumed to have full control of the underlying network: it sees the messages sent by each participant in a given round, and decides which messages will be *delivered* to each participant in that round. The adversary can delete, modify or substitute any message and it can choose to deliver different messages to different participants.

AH-AKE Syntax. We define an AH-AKE scheme as a collection of the following algorithms:

- Setup: on input of security parameter κ, it generates public parameters params.
- KGen: executed by the group CA, on input params, it outputs the group public key $\mathcal{PK}$ and the corresponding secret key $\mathcal{SK}$ for this group, and an empty certificate revocation list $\mathcal{CRL}$. We denote the group corresponding to the public key $\mathcal{PK}$ as $\mathsf{Group}(\mathcal{PK})$.
- Add: executed by the CA of group G, on input $\mathcal{SK}$ and $U \in \mathcal{U}$, it adds U to G by generating a certificate for U, denoted cert. If cert is issued under a public key $\mathcal{PK}$, we say that $\mathsf{cert} \in \mathsf{Certs}(\mathcal{PK})$.
- Revoke: executed by the group CA, on input $U \in \mathcal{U}$, it retrieves the corresponding certificate cert issued for U, and revokes it by adding a new entry which uniquely identifies this certificate to the group CRL. If cert is revoked in list CRL, we say that $\mathsf{cert} \in \mathsf{RevokedCerts}(\mathcal{CRL})$.
- $\mathsf{Handshake}$: This is an interactive protocol executed by two users, e.g. U_i and U_j. The inputs of user U_i is a tuple $(\mathsf{cert}_i, \mathcal{PK}_i, \mathcal{CRL}_i, \mathsf{role}_i)$, where $\mathcal{PK}_i$ is the public key of the group with which U_i wants to establish an authenticated connection, $\mathcal{CRL}_i$ is U_i's current CRL for this group, cert_i is U_i's certificate in that group, assumed to be *globally unique* (see the exact assumption below), and $\mathsf{role}_i \in \{\mathsf{init}, \mathsf{resp}\}$. The inputs of user U_j is the corresponding tuple $(\mathsf{cert}_j, \mathcal{PK}_j, \mathcal{CRL}_j, s_j, \mathsf{role}_j)$. Each party either rejects, or outputs an authenticated secret key, respectively, K_i or K_j.

Instances and Session IDs. In line with prior work on AKE's, e.g. [BPR00,CK01], our model allows for multiple executions of the AH-AKE protocol scheduled in an arbitrary way. Namely, a player $U_i \in \mathcal{U}$ can have many *instances*, involved in distinct concurrent executions of AH-AKE protocol. We denote s-th instance of player U_i as Π_i^s. Each player instance can either reject or accept and output a key. We say that an instance Π_U^s runs a *protocol session*, and we use *player instance* and *protocol session* interchangeably, denoting both as Π_U^s. When referring to a specific user U_i we use Π_i^s as a short-hand for $\Pi_{U_i}^s$. Each instance Π_i^s keeps a state variable, sid_i^s called *session id*, which is always set in our protocols to a concatenation of all public inputs and all the messages sent and received by instance Π_i^s.

Matching Sessions, Partnered Sessions, and the Correctness Property of AH-AKEs. The intended execution of the secret handshake scheme is to allow two players running two *matching* instances to establish an authenticated (and secret) key K, where two instances Π_i^s and Π_j^t care called **matching** if the respective inputs used on these sessions satisfy the following conditions: $\mathcal{PK}_i^s = \mathcal{PK}_j^t$, $\mathsf{cert}_i^s \in \mathsf{Group}(\mathcal{PK}_i^s)$, $\mathsf{cert}_j^t \in \mathsf{Group}(\mathcal{PK}_j^t)$, $\mathsf{cert}_i^s \notin \mathsf{RevokedCerts}(CRL_j^s)$, $\mathsf{cert}_j^t \notin \mathsf{RevokedCerts}(CRL_i^t)$, and $\mathsf{role}_i^s \neq \mathsf{role}_j^t$. We call two protocol instances **partnered** to denote instances which communicate without adversary's interference. Namely, we say that two instances Π_i^s and Π_j^s are partnered if $\mathsf{sid}_i^s = \mathsf{sid}_j^t$, because this condition implies a complete agreement among these two instances with regard to the set of messages sent and delivered between them.

Finally, we say that an AH-AKE scheme is **correct** if, assuming that all keys, certificates and CRL-s are generated by following the Setup, KGen, Add and Revoke procedures,

the following holds that all *matching and partnered* pairs ofinstances Π_i^s, Π_j^t output the same key $K_i^s = K_j^t$.

Security with Perfect Forward Secrecy. We model the security of an AH-AKE scheme similarly to the way security has been defined for general AKE schemes (e.g. [BPR00,CK01]). Namely, we define security via a game between a challenger $\mathcal{C}$ playing the part of a network of m groups and n users, and an adversary $\mathcal{A}$ who starts any number of sessions between these users, and who is in complete control of the network over which they communicate, who is allowed to reveal any number of agreed-upon keys *and* corrupt any number of players, and yet he cannot distinguish from random a key of *any* un-revealed session of a currently uncorrupted player. This is modelled in a standard way, by having an adversary test some session executed by a currently uncorrupted player, which also has not been revealed before (also, the adversary is barred from revealing a session which is partnered with the tested session), at which point a random coin-toss b determines if the adversary sees a key computed on this session or a random value of the same length. The attacker can continue starting and revealing sessions and corrupting players (including the players on the tested session), and finally he outputs his guess b' as to bit b, i.e. as to whether the tested key was real or random.

Formally, security is defined via an interaction of an adversarial algorithm $\mathcal{A}$ and a challenger $\mathcal{C}$ on common inputs (κ, n, m). The interaction starts with $\mathcal{C}$ generating params via $\mathsf{Setup}(\kappa)$, and initializing m groups $G_1, ..., G_m$, by running the $\mathsf{KGen}(\mathsf{params})$ algorithm m times. $\mathcal{C}$ initializes all members in these groups, by running the $\mathsf{Add}(\mathcal{SK}_j)$ algorithm, for each $\mathcal{SK}_j$, $j = 1, ..., m$, for n times. This way, $\mathcal{C}$ generates m certificates for every $U \in \mathcal{U}$, thus making every user a member of every group. The adversary $\mathcal{A}$ gets all generated public keys $\mathcal{PK}_1, ..., \mathcal{PK}_m$. It then chooses any subset $\mathsf{Rev} \subseteq \mathcal{U}$ of initially corrupted players and gets the set of their certificates $\{\mathsf{cert}_i{}^{(j)} \mid U_i \in \mathsf{Rev}, j \in \{1, ..., m\}\}$. For each group G in $\mathcal{G}$, the challenger runs the Revoke algorithm to revoke all corrupted members $U \in \mathsf{Rev}$, and outputs the resulting CRL-s for each group, i.e., $\mathcal{CRL}_1, ..., \mathcal{CRL}_m$.

After this initialization, $\mathcal{A}$ schedules any number of $\mathsf{Handshake}$ protocols, arbitrarily manipulates their messages, requests the keys on any number of the (accepting) sessions, and corrupts any number of additional players, all of which can be modelled by $\mathcal{A}$ issuing any number of the commands listed below. Finally, $\mathcal{A}$ stops and outputs a single bit b'. The commands the adversary can issue, and the way the challenger $\mathcal{C}$ responds to them, are listed below. In all commands we assume that $U \in \mathcal{U} \setminus \mathsf{Rev}$.

- $\mathsf{Start}(U, G, s, \mathsf{role})$: If $U \in \mathcal{U} \setminus \mathsf{Rev}$ and $G \in \mathcal{G}$ and s was not used already in another Start query on the same user, the challenger retrieves key $\mathcal{PK}$ for group G, certificate *cert* issued to player U for group G, and the $\mathcal{CRL}$ corresponding to this group, initiates instance Π_U^s, and follows the the $\mathsf{Handshake}$ protocol on behalf of user U on inputs $(\mathsf{cert}, \mathcal{PK}, \mathcal{CRL}, \mathsf{role})$, forwarding any message generated by U to $\mathcal{A}$. The challenger keeps the state of all initiated instances Π_U^s. We denote the group upon which Π_U^s is initiated as $\mathsf{Group}(\Pi_U^s)$. If Π_U^s is triggered on G^* then $\mathsf{Group}(\Pi_U^s) = G^*$.
- $\mathsf{Send}(U, s, \mathcal{M})$: If instance Π_U^s has been initiated and is waiting for a message, $\mathcal{C}$ delivers M on this instance, and forwards to $\mathcal{A}$ any message U generates in response. If U outputs a key on this session, $\mathcal{C}$ stores it with the session state.

- $\mathsf{Reveal}(U, s)$: If instance Π_U^s has been initiated and has output a session key K, $\mathcal{C}$ delivers this key to $\mathcal{A}$. If the session has either not completed yet or has rejected, $\mathcal{C}$ sends a null value to $\mathcal{A}$. The challenger does not respond if $\mathcal{A}$ queries Reveal on instance Π_U^s for which $\mathcal{A}$ previously issued a Test query (see below) *or* s.t. instance Π_U^s *matches* and is *partnered* with some instance $\Pi_{U'}^{s'}$, $s' = s$, for which $\mathcal{A}$ has issued a Test query.
- $\mathsf{Test}(U, s)$: This query is allowed only once, at any time during the adversary's execution. If Π_U^s is *fresh* (see below) has output some key K, then $\mathcal{C}$ responds depending on its private input bit b. If $b = 1$ then $\mathcal{C}$ sends to $\mathcal{A}$ key K established on this session. If $b = 0$ then $\mathcal{C}$ sends to $\mathcal{A}$ a random κ-bit long value K' instead of K. If the session does not exist, has failed, or is still active, the challenger ignores this Test command.
- $\mathsf{Corrupt}(U_i)$: This query models adaptive corruptions and is essential for modelling perfect forward secrecy of AH-AKEs. The challenger adds U_i to the revocation list Rev, and replies with all long-term secrets $\mathsf{cert}_i^{(1)}, ..., \mathsf{cert}_i^{(m)}$ of user. In particular, the adversary can query $\mathsf{Corrupt}(U)$ even if it has *previously* issued $\mathsf{Test}(U, s)$ for some s.[2]

Freshness. Following [BPR00], we define a notion of *freshness* appropriate for the goal of forward secrecy. An instance Π_U^s is *fresh* unless one the following is true, for any session $\Pi_{U'}^s$ which *matches* and is *partnered* with Π_U^s: (1) The adversary has queried $\mathsf{Reveal}(U, s)$ or $\mathsf{Reveal}(U', s)$; or (2) The adversary has queried $\mathsf{Corrupt}(U)$ or $\mathsf{Corrupt}(U')$.

Definition 1. *Denote $\mathcal{A}$'s output in the above interaction with $\mathcal{C}$ on bit b and (κ, n, m) as $A^{\mathcal{C}(b)}(\kappa, n, m)$. Define the adversary's advantage as follows (the probability goes over the randomness of $\mathcal{A}$ and $\mathcal{C}$):*

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sec}}(\kappa, n, m) = |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{C}(1)}(\kappa, n, m)] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{C}(0)}(\kappa, n, m)]|$$

We call an AH-AKE scheme secure with perfect forward secrecy*, or* PFS*, if for any efficient probabilistic adversary $\mathcal{A}$, for parameters n an m polynomially related to κ, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{sec}}(\kappa, n, m)$ is negligible in κ.*

Linkable Affiliation-Hiding. We define the affiliation-hiding property, similarly as in the security definition, using a game between an adversary and a challenger $\mathcal{C}^{\mathsf{ah}}$. However, the adversary's goal in the affiliation-hiding game is not to violate semantic security of some session key but to learn more about the participants' affiliation by corrupting players, and by learning whether certain sessions were successful. Note that by revealing whether two partnered sessions were successful the adversary learns that the two instances match, and hence that the two players must belong to the same group. We model the property of the attacker's *inability* to learn more than what he is entitled to, by comparing two executions of the adversary: One where the challenger follows the

[2] On the other hand, $\mathcal{C}$ does refuse the $\mathsf{Test}(U, s)$ query if U is corrupted. (Compare with the notion of freshness.) This reflects the simple fact that we can protect security of a protocol run even if one or both of the two players involved is corrupted in the future, but it makes no sense to ask for security of sessions in which one of the participants is already corrupted.

protocols faithfully on behalf of all honest participants, and the other where the adversary interacts with a *simulator*, instead of the real users. The simulator attempts to follow adversary's instructions, except that it is never told the groups for which the (scheduled by the adversary) Handshake protocol instances are executed, i.e., if the adversary issues a $\mathsf{Start}(U, G, s, \mathsf{role})$ query, the simulator gets only an identifier id which is uniquely but arbitrarily assigned to the pair $(U, G) \in \mathcal{U} \times \mathcal{G}$.

Consequently, these inputs are also the only thing that the adversary can possibly learn from the messages produced by this simulator. In other words, the simulated protocol messages can reveal only whether or not two sessions involve *the same* (user,group) pair. However, the adversary does not learn which group, nor can he decide if two instances of two different users belong to the same group. Note that we allow the adversary to be able to *link* instances which involve the same (user,group) pair because the simulator gets the same id for such instances. Indeed, all AH-AKE schemes we propose in this paper are linkable in this sense.

Formally, we model the affiliation-hiding property using an interactive algorithm $\mathcal{SIM}$, function F indexed by the public parameters params of the scheme, and the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}^{\mathsf{ah}}$, on inputs κ, n, m: $\mathcal{C}^{\mathsf{ah}}$ runs $\mathsf{Setup}(\kappa) \rightarrow \mathsf{params}$, $\mathsf{KGen}(\mathsf{params}) \rightarrow (\mathcal{PK}_j, \mathcal{SK}_j)$, for $j \in [1..m]$, and $\mathsf{Add}(\mathcal{SK}_j) \rightarrow \mathsf{cert}_i{}^{(j)}$, for $(i, j) \in [1..n] \times [1..m]$, and gives $\{\mathcal{PK}_j\}_{j=1..m}$ to $\mathcal{A}$. After this initialization, $\mathcal{A}$ can issue any number of queries of the form $\mathsf{Start}(U, G, i, \mathsf{role})$, $\mathsf{Send}(U, s, \mathcal{M})$, $\mathsf{Reveal}(U, s)$, and $\mathsf{Corrupt}(U)$ to $\mathcal{C}^{\mathsf{ah}}$, as in the security game (except there's no Test query). The challenger $\mathcal{C}^{\mathsf{ah}}$ runs on an additional input of bit b, and it responds to $\mathcal{A}$'s commands depending on whether $b = 0$ or 1. If $b = 1$, $\mathcal{C}^{\mathsf{ah}}$ responds to all $\mathcal{A}$'s commands by following the corresponding protocol on behalf of the honest users. If $b = 0$ then $\mathcal{C}^{\mathsf{ah}}$ replies to $\mathcal{A}$'s commands using an *ideal affiliation-hiding* process and a *simulator*, an interactive machine $\mathcal{SIM}$ running on input params, as follows:

- (1) On Start and Send, $\mathcal{C}^{\mathsf{ah}}$ replies with messages produced by the simulator $\mathcal{SIM}$, which instead of $\mathsf{Start}(U_i, G_j, s, \mathsf{role})$ and $\mathsf{Send}(U_i, s, \mathcal{M})$ gets inputs$\mathsf{Start}(id_i^j, s, \mathsf{role})$ and $\mathsf{Send}(id_i^j, s, \mathcal{M})$, respectively, where $id_i^j = F_{\mathsf{params}}(\mathsf{cert}_i{}^{(j)})$.
- (2) On $\mathsf{Corrupt}(U_i)$, $\mathcal{C}^{\mathsf{ah}}$ gives to $\mathcal{A}$ all the long-term secrets of player U_i, i.e. $\{\mathsf{cert}_i^{(j)}\}_{j \in [1..m]}$.
- (3) On $\mathsf{Reveal}(U_i, s)$, $\mathcal{C}^{\mathsf{ah}}$ returns value $\bar{K}_i^s$ chosen as follows. If (a) Π_i^s is *matched and partnered* with some session Π_j^t (note that $\mathcal{C}^{\mathsf{ah}}$ knows this), (b) Π_i^s has received all messages to complete the protocol, (c) all the messages between Π_i^s and Π_j^t up to this point were correctly exchanged, and (d) $\bar{K}_i^s$ is not yet set, then $\mathcal{C}^{\mathsf{ah}}$ picks $\bar{K}_i^s$ at random in $\{0,1\}^\kappa$, sets $\bar{K}_j^t \leftarrow \bar{K}_i^s$, and returns $\bar{K}_i^s$ to $\mathcal{A}$. If (a),(b),(c) holds but not (d), i.e. if $\bar{K}_i^s$ is already set then $\mathcal{C}^{\mathsf{ah}}$ returns this $\bar{K}_i^s$ to $\mathcal{A}$. In every other case $\mathcal{C}^{\mathsf{ah}}$ returns $\bar{K}_i^s = \perp$.

Remark. Note that if an adversary $\mathcal{A}$ exchanges all messages between Π_i^s and Π_j^t, and then reveals whether or not Π_i^s established a key, then $\mathcal{A}$ learns whether or not sessions Π_i^s and Π_i^t are matching, and hence learns that these sessions relate to the same group. This is unavoidable, since a session is supposed to be successful only if it is partnered with a matching one, but this interaction implies that this is the only information divulged by revealing a session key. In particular, the adversary does not learn *which* group these two protocol instances share.

Definition 2. *Denote the output of adversary $\mathcal{A}$ in the above interaction with $\mathcal{C}^{\mathsf{ah}}$ on inputs (κ, n, m), $\mathcal{C}^{\mathsf{ah}}$'s private input b, and $\mathcal{C}^{\mathsf{ah}}$'s access to procedure $\mathcal{SIM}$ and function F, as $\mathcal{A}^{\mathcal{C}^{\mathsf{ah}}(b),\mathcal{SIM},F}(\kappa, n, m)$. Define $\mathcal{A}$'s advantage as follows (where the probabilities are taken over the randomness of $\mathcal{A}$ $\mathcal{C}^{\mathsf{ah}}$, and $\mathcal{SIM}$):*

$$\mathsf{Adv}^{\mathsf{ah}}_{\mathcal{A},\mathcal{SIM},F}(\kappa, n, m) = |\Pr[1 \leftarrow \mathcal{A}^{\mathcal{C}^{\mathsf{ah}}(1),\mathcal{SIM},F}(\kappa, n, m) - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{C}^{\mathsf{ah}}(0),\mathcal{SIM},F}(\kappa, n, m)|$$

We call an AH-AKE scheme linkable affiliation-hiding*, or* LAH*, if there exists a family of functions F and an efficient probabilistic algorithm $\mathcal{SIM}$ s.t.*

1. *For any efficient probabilistic algorithm $\mathcal{A}$ and any parameters n an m polynomially related to κ, the adversarial advantage $\mathsf{Adv}^{\mathsf{ah}}_{\mathcal{A},\mathcal{SIM},F}(\kappa, n, m)$ is a negligible function of κ.*
2. *There is a negligible function ϵ s.t. for any params output by $\mathsf{Setup}(\kappa)$, and any two keys pairs $(\mathcal{PK}_0, \mathcal{SK}_0)$ and $(\mathcal{PK}_1, \mathcal{SK}_1)$ output by $\mathsf{KGen}(\mathsf{params})$, the statistical distance between distribution D_0 and D_1 is bounded by $\epsilon(\kappa)$, where $D_b = \{F(\mathsf{cert}) \mid \mathsf{cert} \leftarrow \mathsf{Add}(\mathcal{SK}_b)\}$.*

Remark. Intuitively, requirement (2) implies that $F(\mathsf{cert}_i{}^{(j)})$ reveals no information about the group G_j that issued $\mathsf{cert}_i{}^{(j)}$. Therefore, by requirement (1), the only information that $\mathcal{A}$ learns when attacking a LAH scheme, is a "pseudonym" $id_{i,j} = F(\mathsf{cert}_i{}^{(j)})$ which corresponds to user U_i and group G_j, but which does not leak what group this pseudonym corresponds to.

By a simple hybrid argument we can show that LAH implies PFS. Intuitively, this is because the affiliation-hiding game compares the view of the real execution with a "fully-random" view, where all messages and keys are chosen by the challenger and a simulator, whereas the security game compares the real view with a view modified so that only the key of the tested session is chosen at random. It's not difficult to see that a significant difference between the views in the second pair implies a significant difference between the views in the first pair. The exact security in this reduction decreases by a small constant factor.

Lemma 1. *If AH-AKE scheme is* linkable affiliation-hiding *then it is secure with* perfect forward secrecy*.*

Proof. Let $\mathcal{A}$ be an (adaptive) adversary which attacks the security game. Denote by $\mathsf{p\text{-}sc}_{\mathcal{A},b}$ the probability that $\mathcal{A}$ outputs 1 on the interaction defined as in the security game with the challenger $\mathcal{C}$ running on but b. The construction of an adaptive adversary $\mathcal{A}'$ which attacks the (PFS-enabled) affiliation-hiding game is trivial: $\mathcal{A}'$ forwards the messages from its challenger $\mathcal{C}^{\mathsf{ah}}$ to $\mathcal{A}$, and it similarly forwards all the commands from $\mathcal{A}$ to $\mathcal{C}^{\mathsf{ah}}$, except the $\mathsf{Test}(U, s)$ command for which $\mathcal{A}'$ issues $\mathsf{Reveal}(U, s)$ to $\mathcal{C}^{\mathsf{ah}}$, and returns $\mathcal{C}^{\mathsf{ah}}$'s response to $\mathcal{A}$. When $\mathcal{A}$ stops and outputs a bit b', $\mathcal{A}'$ returns the same bit.

Let $\mathsf{p\text{-}sc}_b$ denote the probability that $\mathcal{A}$ outputs 1 on the interaction defined as in the security game with the challenger $\mathcal{C}(b)$, and $\mathsf{p\text{-}ah}_b$ be the probability that $\mathcal{A}'$ outputs 1 when interacting with $\mathcal{C}^{\mathsf{ah}}(b)$. Note that $\mathsf{p\text{-}ah}_1 = \mathsf{p\text{-}sc}_1$ because in both cases this is the interaction of $\mathcal{A}$ with the real protocol. Let $p \approx p'$ denote that $|p - p'|$ is a negligible function of the security parameter. Then by the assumption that the scheme

is linkable affiliation-hiding, $\mathsf{p\text{-}ah}_0 \approx \mathsf{p\text{-}ah}_1 = \mathsf{p\text{-}sc}_1$. We will argue that $\mathsf{p\text{-}sc}_0 \approx \mathsf{p\text{-}sc}_1$ as well. Let $p(b_1, b_2, b_3)$ denote the probability $\mathcal{A}$ outputs 1 on interaction with the challenger which computes all messages and keys as in the real protocol, except that (1) if $b_1 = 0$ then all players' messages are computed as $\mathcal{C}^{\mathsf{ah}}$ on $b = 0$, i.e. via the simulator $\mathcal{SIM}$, and (2) if $b_2 = 0$ then all the revealed keys are chosen independently at random for every session (unless Π_i^s matches some partnered session $\Pi_{i'}^{s'}$, in which case $K_i^s = K_{i'}^{s'}$), again as in the procedure for $\mathcal{C}^{\mathsf{ah}}$ on $b = 0$, and (3) if $b_3 = 0$ then also the key of the tested session is chosen at random. Using this notation we have $\mathsf{p\text{-}ah}_0 = p(0,0,0)$, $\mathsf{p\text{-}sc}_0 = p(1,1,0)$, $\mathsf{p\text{-}ah}_1 = \mathsf{p\text{-}sc}_1 = p(1,1,1)$. Our assumption is that $\mathsf{p\text{-}ah}_0 = p(0,0,0) \approx p(1,1,1) = \mathsf{p\text{-}ah}_1$, and so if we show that $p(0,0,0) \approx p(1,1,0)$, this will imply that $\mathsf{p\text{-}sc}_0 = p(1,1,0) \approx p(1,1,1) = \mathsf{p\text{-}sc}_1$.

Now, if $p(0,0,0) \not\approx p(1,0,0)$ then by a trivial reduction which substitutes the revealed and tested session keys with independently chosen random keys (except for partnered matching sessions, as above), it would follow that $p(0,0,0) \not\approx p(1,1,1)$. By contradiction, we get $p(0,0,0) \approx p(1,0,0)$. By a similar argument on just the revealed keys, if $p(0,0,0) \not\approx p(1,1,0)$ then we'd have $p(0,0,0) \not\approx p(1,0,0)$, and hence it follows that $p(0,0,0) \approx p(1,1,0)$ as needed.

3 Authenticated Key Agreement with PFS and LAH based on the RSA Assumption

- Setup: Given security parameter κ, the Set-up defines another security parameter κ' (polynomial in κ), s.t. the RSA assumption holds on $(2\kappa')$-long composites with security parameter κ. The setup also defines hash function $H_1 : \{0,1\}^* \to \{0,1\}^\kappa$.
- KGen: Generate a $2\kappa'$-bit safe RSA modulus $n = pq$, where $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are primes. Pick a random element g s.t. g generates a maximum subgroup in Z_n^*, i.e. $ord(g) = 2p'q'$, and s.t. $-1 \notin \langle g \rangle$. (This holds for about half of the elements in Z_n^*, and it is easily tested.) Note that in this case $Z_n^* \equiv \langle -1 \rangle \times \langle g \rangle$. Therefore, in particular, if $x \leftarrow Z_{2p'q'}$ and $b \leftarrow \{0,1\}$ then $(-1)^b g^x$ is distributed uniformly in Z_n^*. RSA exponents (e, d) are chosen in the standard way, as a small prime e and $d = e^{-1} \pmod{\phi(n)}$. The secret key is (p, q, d) and public key is (n, g, e). Key generation also fixes a hash function $H_n : \{0,1\}^* \to Z_n$, specific to the group modulus n.[3]
- Add: To add user U to the group, the manager picks a random string $id \leftarrow \{0,1\}^\kappa$ and computes a (full-domain hash) RSA signature on id, $\sigma = h^d \pmod{n}$, where $h = H_n(id)$. U's certificate is $\mathsf{cert} = (id, \sigma)$.
- Revoke: To remove user U from the group, the manager appends string id to the group $\mathcal{CRL}$, where (σ, id) is U's certificate in this group.
- Handshake: This is an AKE protocol for users U_A and U_B of the honest players, where player U_A's inputs a tuple $(\mathsf{cert}_A, \mathcal{PK}_A = (n_A, e_A, g_A), \mathcal{CRL}_A, \mathsf{init})$ and U_B's inputs $(\mathsf{cert}_B, \mathcal{PK}_B = (n_B, e_B, g_B), \mathcal{CRL}_B, \mathsf{resp})$ s.t. $\mathsf{cert}_A = (id_A, \sigma_A)$ is U_A's certificate for

[3] Selecting separate hash function H_n for every group is done purely for notational convenience. A family of hash functions $H_n : \{0,1\}^* \to Z_n$ s.t. each H_n is statistically close to a random function with range Z_n, can be easily implemented in the random oracle model with a single hash function with range $2^{2\kappa'+\kappa}$. E.g., $H_n(m) = H(n, m) \bmod n$.

the public key $\mathcal{PK}_A = (n_A, e_A, g_A)$, i.e. $\mathsf{cert}_A \in \mathsf{Certs}(PK_A)$, $\mathcal{CRL}_A$ is the (hopefully recent) CRL for group $\mathsf{Group}(n_A, e_A, g_A)$, and similarly $\mathsf{cert}_B = (id_B, \sigma_B)$, $\mathcal{PK}_B = (n_B, e_B, g_B)$, and $\mathcal{CRL}_B$ are defined for U_B. In the protocol, player U_A picks $b_A \leftarrow \{0,1\}$ and $x_A \leftarrow Z_{n_A}$, and computes $\theta'_A = (-1)^{b_A}\sigma_A g^{x_A} \bmod n_A$ and $\theta_A = \theta'_A + k_A n_A$ where $k_A \in [0, ..., \lfloor 2^{2\kappa'+\kappa}/n_A \rfloor]$. [4] U_B follows the same process to compute θ_B with random b_B and x_B. The protocol runs as in Figure 1 below.

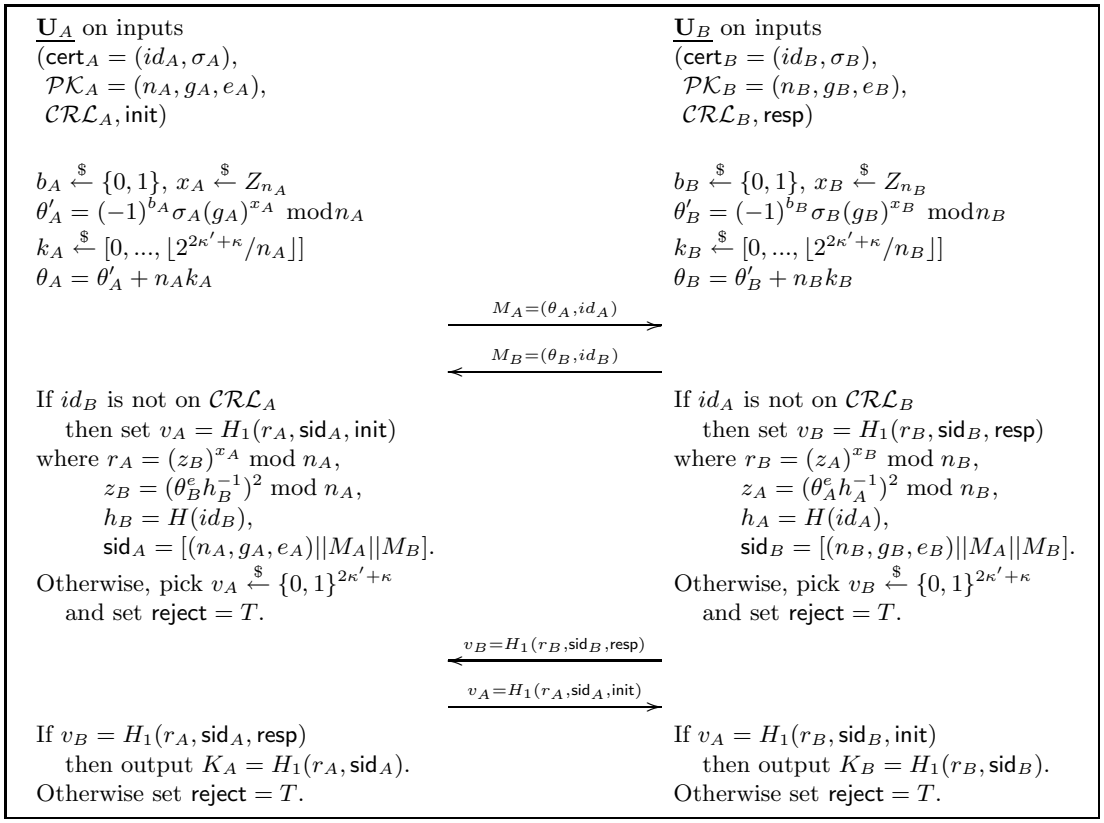

Fig. 1. AH-AKE protocol with LAH and PFS based on RSA assumption

To check that this scheme is correct, observe that because $z_A = ((\theta_A)^e (h_A)^{-1})^2 = g^{2ex_A}$ and $z_B = ((\theta_B)^e (h_B)^{-1})^2 = g^{2ex_B}$. Therefore both $r_A = (z_B)^{x_A} = g^{2ex_A x_B}$ and

[4] Note that if θ'_A is uniform in Z_{n_A} then the distribution of $\theta_A = \theta'_A + k_A n_A$, where k_A is picked as above, is statistically close to $U_{2^{2\kappa'+\kappa}}$. There's an alternative way to hide the range of θ_A, which does not take the κ bandwidth overhead, is to repeat picking θ'_A until $\theta'_A \in \{0,1\}^{2\kappa'-1}$. However, the expected running time of such procedure is at most twice that of our procedure, and this alternative procedure could also be subject to timing attacks. Note that the overhead of κ bits we incur is small compared to $|\theta'_A| = |n_A| = 2\kappa'$.

$r_B = (z_A)^{x_B} = g^{2ex_Ax_B}$. We show that the scheme is linkable affiliation hiding (LAH), and, hence, it is also secure with perfect forward secrecy (PFS). Note that the exact security shown by the reduction below is tight, and the exact bounds can be easily derived from the proof.

Definition 3. *Let* $\mathsf{S\text{-}RSA\text{-}IG}(\kappa)$ *be an algorithm that outputs so-called safe RSA instances, i.e. pairs* (n, e) *where* $n = pq$*,* e *is a small prime that satisfies* $gcd(e, \phi(n)) = 1$*, and* p, q *are randomly generated* κ*-bit primes subject to the constraint that* $p = 2p' + 1$*,* $q = 2q' + 1$ *for prime* p', q'*,* $p' \neq q'$*.*

We say that the RSA problem is (ϵ, t)*-hard on* 2κ*-bit safe RSA moduli, if for every algorithm* $\mathcal{A}$ *that runs in time* t *we have*

$$\Pr[(n, e) \leftarrow \mathsf{S\text{-}RSA\text{-}IG}(\kappa), g \leftarrow \mathbb{Z}_n^* : \mathcal{A}(n, e, g) = z \text{ s.t. } z^e = g \pmod n] \leq \epsilon.$$

Theorem 1. *Under the RSA assumption on safe RSA moduli, the above AH-AKE scheme is secure with Perfect Forward Secrecy and with Linkable Affiliation-Hiding, in the Random Oracle Model.*

Proof. By lemma 1, we only need to argue LAH, i.e. we need to show that $\mathcal{A}$'s view of the interaction with the challenger $\mathcal{C}^{\mathsf{ah}}$ on bit $b = 1$ is indistinguishable from the view of the interaction with $\mathcal{C}^{\mathsf{ah}}$ on bit $b = 0$. Let Game0 represent a real execution, i.e., interaction of $\mathcal{A}$ with challenger $\mathcal{C}^{\mathsf{ah}}$ on bit $b = 1$, while Game2 represents a simulation, i.e., interaction of $\mathcal{A}$ with challenger $\mathcal{C}^{\mathsf{ah}}$ on bit $b = 0$. Thus, our goal is to demonstrate that $\mathcal{A}$'s view in Game0 is indistinguishable from $\mathcal{A}$'s view in Game2. Consider Game1, which is like Game0, except that it stops if there is ever a collision in sid values of any two instances. Since θ_i sent by U_i is indistinguishable from $(2\kappa' + \kappa)$-bit string (see below), probability that there is a collision in polynomially many executions is negligible. Therefore Game0 and Game1 are indistinguishable.

Simulation. To describe Game2, i.e. the simulation, we need to define the simulator $\mathcal{SIM}$ and a function family F. Note that $\mathsf{params} = (\kappa, \kappa')$ and that certificates cert are pairs of the form (id, σ) where $id \in \{0,1\}^\kappa$. We will set $F_{(\kappa,\kappa')}(\mathsf{cert}) = id$. Note that this function satisfies requirement (2) in the LAH definition because the id part of any certificate cert is a random κ-bit string independent of the group's key. Now we describe the simulator $\mathcal{SIM}$, and at the same time we recall how $\mathcal{C}^{\mathsf{ah}}$ interacts with $\mathcal{A}$ using this simulator and function F. Note that the simulator is only involved in the Start and Send queries. Since the protocols of the initiator and the responder are symmetric, below we only describe the initiator's part.

- First $\mathcal{C}^{\mathsf{ah}}$ initializes all the groups and all the users in these groups using Setup, KGen, and Add algorithms as in the real protocol, and gives all the group public keys to $\mathcal{A}$.
- On $\mathsf{Start}(U_i, G_j, s, \mathsf{init})$ from $\mathcal{A}$, $\mathcal{C}^{\mathsf{ah}}$ responds with the output of $\mathcal{SIM}(id_{i,j}, s, \mathsf{init})$, where $id_{i,j} = F_{(\kappa,\kappa')}(\mathsf{cert}_i{}^{(j)})$. On inputs $\mathcal{SIM}(id_{i,j}, s, \mathsf{init})$, $\mathcal{SIM}$ returns $(\theta_i^s, id_{i,j})$ where $\theta_i^s \xleftarrow{\$} \{0,1\}^{2\kappa'+\kappa}$.
- On $\mathsf{Send}(U_i, s, \mathcal{M})$ from $\mathcal{A}$, $\mathcal{C}^{\mathsf{ah}}$ responds with the output of $\mathcal{SIM}(id_{i,j}, s)$ where $id_{i,j}$ is the id corresponding to Π_i^s. The simulator, regardless of $\mathcal{M}$, returns v_i^s where $v_i^s \xleftarrow{\$} \{0,1\}^{2\kappa'+\kappa}$.

– On $\mathsf{Corrupt}(U_i)$, $\mathcal{C}^{\mathsf{ah}}$ gives to $\mathcal{A}$ all the long-term secrets of player U_i, i.e. set $\{\mathsf{cert}_i^{(j)}\}_{j\in[1..m]}$.
– On $\mathsf{Reveal}(U_i, s)$, $\mathcal{C}^{\mathsf{ah}}$ returns value $\bar{K}_i^s$ chosen as described in the LAH definition. Briefly, it's a random κ-bit string if the sessions are matched, partnered, and all messages were exchanged up to this point, and otherwise it's a $\perp$ symbol. The only exception is that $\bar{K}_j^t = \bar{K}_i^s$ on the two matched sessions on which all messages were exchanged properly.

Let HQuery be an event that $\mathcal{A}$ ever queries H_1 on arguments $(r_i^s, \mathsf{sid}_i^s)$, or $(r_i^s, \mathsf{sid}_i^s, \mathsf{init})$, or $(r_i^s, \mathsf{sid}_i^s, \mathsf{resp})$, for any Π_i^s that $\mathcal{A}$ starts, where r_i^s is defined via the combination of the message (θ_i^s, id_i, s) which instance Π_i^s of an honest player U_i sent on that session, and message $M = (\hat{\theta}, \hat{id}, s)$ which $\mathcal{A}$ sent to Π_i^s in his $\mathsf{Send}(U_i, s, M)$ command, as follows:

$$r_i^s = (\hat{z})^{x_i^s} \bmod n \quad \text{where} \quad \hat{z} = g^{2e\hat{x}} = (\hat{\theta})^{2e}(\hat{h})^{-2} \quad \text{and} \quad z_i^s = g^{2ex_i^s} = (\theta_i^s)^{2e}(h_i)^{-2} \qquad (1)$$

where $\hat{h} = H_n(\hat{id})$ and $h_i = H_n(id_i)$. In other words, HQuery is an event that $\mathcal{A}$ computes (and enters into hash function H_1) the key-material r_i^s for *any* instance Π_i^s run by an honest player.

Claim 1. Unless HQuery happens, $\mathcal{A}$'s view of the interaction with the challenger in Game1 is indistinguishable from the view of the interaction with the challenger in Game2.

Note that if HQuery does not happen then all the challenge/response values v_i^s and keys K_i^s that $\mathcal{A}$ gets in Game1 are distributed the same as in the real protocol, i.e. as independently chosen random κ-bit strings. Moreover, all the messages (θ_i^s, id_i) the adversary sees are also statistically close to the corresponding values in the real execution. The reason is that in both cases, the simulation and the execution, each value θ_i^s is distributed statistically close to a uniform bit-string of length $2\kappa' + \kappa$, and it is independent of id_i (and σ_i). Note that since $Z_n^* \equiv \langle -1\rangle \times \langle g\rangle$ value $\theta' = (-1)^b g^x \sigma \pmod n$ is random in Z_n^* if b is a random bit and if x is random in $Z_{2p'q'}$. Since $n - (4p'q')$ is on the order of $\sqrt{n}$, which is negligible compared to n, the distribution of θ' for x chosen in Z_n is still statistically close to uniform in Z_n^*. Similarly, since there are only $O(\sqrt{n})$ elements in Z_n/Z_n^*, this random variable is also statistically close to uniform in Z_n. Finally, for any $s > n2^\kappa$, value $\theta = \theta' + k * n$ (over integers) for random θ' in Z_n and $k \xleftarrow{\$} [0, ..., \lfloor s/n \rfloor]$, is statistically indistinguishable from a random value in Z_s.

Claim 2. If event HQuery happens with non-negligible probability on input $(r_i^s, \mathsf{sid}_i^s)$, or $(r_i^s, \mathsf{sid}_i^s, \mathsf{init})$, or $(r_i^s, \mathsf{sid}_i^s, \mathsf{resp})$, for any Π_i^s, then $\mathcal{A}$ can be used to break the RSA assumption.

We divide the adversary into three types classes, depending on the message $M = (\hat{id}, \hat{\theta})$ which is involved in computation of r_i^s. Type I adversary makes HQuery s.t. the related $\hat{id}$ and $\hat{\theta}$ are rerouted from another honest player's instance. Type II adversary makes HQuery s.t. the related $\hat{id}$ is created by the adversary. Type III adversary makes HQuery s.t. the related $\hat{id}$ is rerouted but $\hat{\theta}_j$ is created by the adversary.

We describe each reduction algorithm using a modified challenger algorithm called $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$. Let G^* be the group s.t. the probability that $\mathcal{A}$ queries H_1 on r_i^s corresponding to Π_i^s where $\mathsf{Group}(\Pi_i^s) = G^*$, is at least $1/m$. For each type of adversaries, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ takes

an RSA public key (n, e) of one of the groups denoted by G^* as an input and picks the private/public keys for all the remaining groups. The $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ issues users' certificates for all groups except of G^* correctly as in the real execution. However, for users in G^* the modified challenger will need to simulate the signatures on each id_i by setting $H_n(id_i)$ as $a^e \bmod n$ for some random value a. This way $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ can present the certificate of player id_i in G^* as a. (The exact way that values a are chosen is described in adversary type I-III below.) The modified challenger can fail if $\mathcal{A}$ has made a query to H_n on the randomly chosen id_i value, for any i, but this happens with a negligible probability of at most $nq_H/2^\kappa$, and otherwise the certificates are distributed as in the execution. In each case, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ responds to Send commands as in Simulation above, additionally storing $[j, \Pi_i^s, \mathsf{sid}_i^s]$ in table denoted T_{H_1}, which is used by a reduction algorithm every time $\mathcal{A}$ makes a query to H_1 (see below).

Type I Adversary.

- **Setup and Initialization.** On the RSA challenge (n, e, z), $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ sets the public key of G^* as (n, e, g) where $g = h^{\alpha e^2}$ for $h \xleftarrow{\$} Z_n^*$ and $\alpha \xleftarrow{\$} Z_n$. Note that given a a safe RSA modulus n, with probability about $1/2$ we have that $Z_n^* \equiv \langle -1 \rangle \times \langle g \rangle$. The rest part of initialization is the same as in the real protocol.
- **Hash queries to H_n and H_1.** $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ sets $H_n(id_i) = (-1)^{d_i}/g^{a_i e} \pmod n$ for random $(d_i, a_i) \in Z_2 \times Z_n$ for each U_i. For the queries to H_1, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ simply passes these queries to H_1. However, for each query (r, sid) and $(r, \mathsf{sid}, \mathsf{role})$ to H_1, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ also tries to solve the RSA challenge as we describe below.
- **Corrupt queries.** $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ responds to $\mathsf{Corrupt}(U_i)$ with $(id_i, (-1)^{d_i}/g^{a_i})$.
- **Start queries.** On $\mathsf{Start}(U_i, G^*, s, \mathsf{init})$ from $\mathcal{A}$, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ responds with the output of $\mathcal{SIM}(id_i, s, \mathsf{init})$, where $id_i = F_{(\kappa,\kappa')}(\mathsf{cert}_i{}^{(*)})$. On inputs $\mathcal{SIM}(id_i, s, \mathsf{init})$, $\mathcal{SIM}$ returns (θ_i^s, id_i): $\mathcal{SIM}$ sets θ_i^s as either $hg^{c_i^s}(-1)^{b_i^s} \bmod n$ or $zg^{c_i^s}(-1)^{b_i^s} \bmod n$, plus the random kn shift, with probability $1/2$ each, for random $(c_i^s, b_i^s) \in Z_n \times Z_2$. Notice that all these values are distributed indistinguishably from the distribution produced by the real execution.
- **Reduction algorithm from HQuery event.** With probability of at least $\epsilon/16m$, for the Π_i^s and Π_j^t instances involved in $\mathcal{A}$'s query on r_i^s, we have $\theta_i^s = hg^{c_i^s}(-1)^{b_i^s} \bmod n$ and $\theta_j^t = zg^{c_j^t}(-1)^{b_j^t} \bmod n$. We replace RSA challenge z by h^k for some unknown k. Then it's easy to see that if $(\hat{\theta}, \hat{id}) = (\theta_j^t, id_j)$ then in equation (1) we get $z_i^s = g^{2ex_i^s} = g^{2e(1+\alpha e^2(a_i+c_i^s))(\alpha e^2)^{-1}}$ and $\hat{z} = g^{2e(k+\alpha e^2(a_j+c_j^t))(\alpha e^2)^{-1}}$. Therefore,

$$r_i^s = g^{2e[k(\alpha e^2)^{-1}+a_j+c_j^t][(\alpha e^2)^{-1}+a_i+c_i^s]}$$

  Since $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ knows α, e, a_j, c_j^t, a_i and c_i^s, $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ can extract $g^{2k(\alpha^2 e^3)^{-1}}$, from which $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ can compute $z^{2d\alpha^{-1}}$ since $g = h^{\alpha e^2}$ and $z = h^k$. Thus, z^{2d} can be extracted. Since $gcd(2, e) = 1$, therefore, computing z^{2d} leads to computing z^d. This reduction algorithm is executed whenever $\mathcal{A}$ makes a query (r, sid) (or $(r, \mathsf{sid}, \mathsf{role})$) to H_1 for each entry $[j, \Pi_i^s, \mathsf{sid}_i^s]$ in table T_{H_1} s.t. $\mathsf{sid}_i^s = \mathsf{sid}$. $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ can verify which entry is related to the given value r, since after computing $w = z^{2d}$ as above $\mathsf{F}\text{-}\mathcal{C}^{\mathsf{ah}}$ can test if $w^e = z^2$.

Type II Adversary.

- **Setup and Initialization.** On the RSA challenge (n, e, z), F-$\mathcal{C}^{\mathsf{ah}}$ sets the public key of G^* as (n, e, g) where $g = \alpha^e$ for $\alpha \leftarrow Z_n^*$. (Note that a random g in Z_n^* matches that chosen by a real key generation with probability about 1/2). The rest part of initialization is the same as in the real protocol.
- **Hash queries to H_n and H_1.** F-$\mathcal{C}^{\mathsf{ah}}$ answers a query to H_n on x depending on the source of x. Namely, F-$\mathcal{C}^{\mathsf{ah}}$ responds with $H_n(x) = a_x^e/g \bmod n$ for a randomly chosen $a_x \in Z_n^*$ if x corresponds to some id of an honest player generated by the simulator. Let $H_n(id_i) = a_i^e g^{-1}$. If x does not match with any ids created by the simulator, F-$\mathcal{C}^{\mathsf{ah}}$ replies with $H_n(x) = a_x^e/z \bmod n$ for a randomly chosen $a_x \in Z_n^*$. For the queries to H_1, F-$\mathcal{C}^{\mathsf{ah}}$ simply passes these queries to H_1. However, for each query (r, sid) and $(r, \mathsf{sid}, \mathsf{role})$ to H_1, F-$\mathcal{C}^{\mathsf{ah}}$ also tries to solve the RSA challenge as we describe below.
- **Corrupt queries.** F-$\mathcal{C}^{\mathsf{ah}}$ responds to $\mathsf{Corrupt}(U_i)$ with $(id_i, a_i/\alpha)$.
- **Start queries.** On $\mathsf{Start}(U_i, G^*, s, \mathsf{init})$ from $\mathcal{A}$, F-$\mathcal{C}^{\mathsf{ah}}$ responds with the output of $\mathcal{SIM}(id_i, s, \mathsf{init})$, where $id_i = F_{(\kappa,\kappa')}(\mathsf{cert}_i{}^{(*)})$. On inputs $\mathcal{SIM}(id_i, s, \mathsf{init})$, $\mathcal{SIM}$ returns (θ_i^s, id_i) where $\theta_i^s = (-1)^{b_i^s} a_i g^{\gamma_i^s}$ plus the random kn shift, for random $(\gamma_i^s, b_i^s) \in Z_n \times Z_2$.
- **Reduction algorithm from HQuery event.** With probability of (almost) at least $\epsilon/4m$, the query r corresponds to session Π_i^s on which $\mathcal{A}$ sends $\hat{id} \neq id_j$ for any honest U_j, i.e., $r = r_i^s = (\hat{z})^{x_i^s}$. Since $\theta_i^s = (-1)^{b_i^s}(H(id_i))^d g^{d+\gamma_i^s}$ and $\hat{z} = (\hat{\theta}/a)^{2e} z^2$ where $H_n(\hat{id}) = a^e z^{-1}$, we get $z_i^s = g^{2ex_i^s} = g^{2e(d+\gamma_i^s)}$ from equation (1). Therefore $(\hat{z})^{x_i^s} = (\hat{\theta}/a)^{2e(d+\gamma_i)} z^{2(d+\gamma_i)} = (\hat{\theta}/a)^{(2+2e\gamma_i)} z^{2\gamma_i} z^{2d}$ and F-$\mathcal{C}^{\mathsf{ah}}$ can extract z^{2d}. Since $gcd(2, e) = 1$, F-$\mathcal{C}^{\mathsf{ah}}$ can compute z^d from $(\hat{z})^{x_i^r}$.

Type III Adversary. In the last case, we assume that with probability at least $\epsilon/4m$ the *first such* $(\hat{z})^{x_i^s}$ which makes event HQuery true corresponds to session Π_i^s on which $\mathcal{A}$ sends $\hat{id} = id_j$ for some session Π_j^t matching Π_i^s for some *currently uncorrupted* U_j, but $\hat{\theta} \neq \theta_j^t$. We show that F-$\mathcal{C}^{\mathsf{ah}}$ can solve the RSA problem in this case. The view that it will present to $\mathcal{A}$ will match what $\mathcal{A}$ expects *until* the above query $(\hat{z})^{x_i^s}$, i.e. HQuery, is done, in which case F-$\mathcal{C}^{\mathsf{ah}}$ will solve the RSA problem. Note that it is unimportant whether F-$\mathcal{C}^{\mathsf{ah}}$ can continue presenting $\mathcal{A}$ with the correct view afterwards. Let U_j be a player s.t. the probability that this query is done *and* that it involves $\hat{id} = id_j$ is at least $\epsilon/4mn$. Since Setup, Start, and the reduction algorithm are the same as the ones for Type II adversary, in the following we describe the rest algorithms only.

- **Hash queries to H_n and H_1.** F-$\mathcal{C}^{\mathsf{ah}}$ responds with $H_n(x) = a^e/g$ for a randomly chosen $a \in Z_n^*$ if x corresponds to some id of an honest player $U_i \neq U_j$. If x match with the id of U_j, F-$\mathcal{C}^{\mathsf{ah}}$ replies with $H_n(x) = a^e/z \bmod n$ for a randomly chosen $a \in Z_n^*$. Let $H_n(id_i) = a_i^e g^{-1}$.
- **Corrupt queries.** F-$\mathcal{C}^{\mathsf{ah}}$ responds to $\mathsf{Corrupt}(U_i)$ with $(id_i, a_i/\alpha)$ if $U_i \neq U_j$. On $\mathsf{Corrupt}(U_j)$, F-$\mathcal{C}^{\mathsf{ah}}$ stops. As we argued above, it does not matter that this reduction cannot open the state of player U_j with a valid-looking signature on id_j, since at the time $\mathcal{A}$ makes the crucial r_i^s query the player U_j must be still uncorrupted.

References

[BDS+03] 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.

[BPR00] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *proceedings of Eurocrypt*, 2000.

[CJT04] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from CA-oblivious encryption. In *proceedings of Asiacrypt*, 2004.

[CK01] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Proceedings of CRYPTO'2001*, 2001.

[CKGS98] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45:965–981, 1998.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, 2004.

[D.V05] D.Vergnaud. Rsa-based secret handshakes. In *International Workshop on Coding and Cryptography*, 2005.

[JKT07] S. Jarecki, J. Kim, and G. Tsudik. Group secret handshakes, or affiliation-hiding authenticated group key agreement. In *CT-RSA'07*, 2007.

[JL07] S. Jarecki and X. Liu. Unlinkable secret handshakes and key-private group key management schemes. In *proceedings of ACNS*, 2007.

[LDB03] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *Proceedings of 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, 2003.

[MRH01] M.Bellare, R.Canetti, and H.Krawczyk. A modular approach to the design and analysis of authentication and key-exchange protocols. In *30th STOC*, 2001.

[TX06] G. Tsudik and S. Xu. A flexible framework for secret handshakes. In *rivacy-Enhancing Technologies Workshop (PET'06)*, 2006.

[V.S99] V.Shoup. On formal models for secure key exchange. In *Theory of Cryptography Library*, 1999.

[XY04] S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 158–167, New York, NY, USA, 2004. ACM Press.