Unlinkable Secret Handshakes and Key-Private Group Key Management Schemes

Stanisław Jarecki and Xiaomin Liu

University of California, Irvine

Abstract. We present the first practical *unlinkable secret handshake* scheme. An unlinkable secret handshake is a two-way authentication protocol in a PKI setting which protects privacy and anonymity of *all* information about the participants to *everyone* except of their intended authentication partners. Namely, if entity A certified by organization CA^A wants to authenticate itself only to other entities certified by CA^A, and, symmetrically, entity B certified by CA^B wants to authenticate itself only to entities also certified by CA^B, then a secret handshake protocol authenticates these parties and establishes a fresh shared key between them if and only if $CA^A = CA^B$ and the two parties entered valid certificates for this CA into the protocol. If, however $CA^A \neq CA^B$, or $CA^A = CA^B$ but either A or B is not certified by this CA, the secret handshake protocol reveals *no information* to the participants except of the bare fact that their inputs do not match. In other words, an Unlinkable Secret Handshake scheme is a perfectly private authentication method in the PKI setting: One can establish authenticated communication with parties that possess the credentials required by one's policy, and at the same time one's affiliation *and* identity remain perfectly secret to everyone except of the parties to whom one wants to authenticate.

Efficient secret handshake schemes, i.e. authentication protocols which protect the privacy of participants' affiliations, were proposed before, but participants in these schemes remained *linkable*. Namely, an attacker could recognize all the instances of the protocol executed by the same entity. Secondly, the previous schemes surrendered user's privacy if the certificates of this user were revoked, and our scheme alleviates this problem as well. Unlinkable schemes were proposed as well, but they either relied on single-use certificates, or did not support revocation, or required instantaneous propagation of revocation information.

Crucial ingredients in our construction of unlinkable secret handshakes are chosen-ciphertext secure key-private encryption and multi-encryption schemes, and the first efficient construction of a key-private group key management scheme, which is a stateful analogue of (key-private) public key broadcast encryption.

1 Introduction

Privacy of Authentication in the PKI Model. "Unlinkable Secret Handshake" is a name we give to an Authenticated Key Exchange [AKE] scheme which, in

J. Katz and M. Yung (Eds.): ACNS 2007, LNCS 4521, pp. 270–287, 2007.

addition to the standard security properties needed of an AKE scheme provides privacy properties of *affiliation hiding*, *policy hiding*, and *unlinkability*. In order to explain these privacy properties we need to recall how authentication (and authenticated key exchange) works in the public-key infrastructure [PKI] model. In the PKI model each party holds a certificate *cert* on its public key issued by some *Certification Authority* [CA], the fact which we denote as $cert \in CA$. (In the general PKI model, certificates can be put together in arbitrarily long chains, but here we consider only a "flattened PKI model" which does not support certificate chains.) We call the CA an *affiliation* of that party and we use the terms "certified by" and "affiliated with" interchangeably. We refer to the parties affiliated with a given CA as its *group*, and we call the CA a *group manager*.

In addition to a list of certificates, each party also holds an authentication *policy*, represented by a list of CA's, which specifies that this party wants to establish authenticated communication only with entities affiliated with these CA's. In some applications parties affiliated with some CA might have a policy to authenticate only to other entities affiliated with the same CA, e.g. with the employees of the same company, or with the members of the same organization. In a common case a player's policy will *include* all the CA's this player is affiliated with, but in general the two lists might have nothing in common.

Two-sided authentication in the PKI model is successful depending on whether there is a match between the affiliations and the policies of the two interacting parties. For simplicity of discussion, let each player hold only a single certificate, and let its authentication policy consist also of only a single "target" CA. In a secure PKI-based authentication scheme, if party A enters into the computation a certificate $cert^A$ and policy CA^A, and party B enters certificate $cert^B$ and policy CA^B, both players accept (and output a fresh authenticated session key) only if $cert^A \in CA^B$ and $cert^B \in CA^A$.

The standard way in which players A and B discover if their affiliations and policies match is to announce for one party to announce its affiliation and policies to the other. This match-discovery process is then followed by the cryptographic protocol, consisting of the players' verifying each other's certificates and running an AKE protocol on the public keys in these certificates. However, this means that some player's affiliations and/or policies are effectively available to anyone who requests it. This unrestricted leakage of authentication policy and/or affiliation information of a party is a privacy threat, because in many applications both one's affiliation and one's policy is a sensitive information that should be protected from unnecessary exposure. (Note also that most commonly one's policy immediately reveals one's affiliation.) The only entities that have the right to know that player A is affiliated with CA_1 and by policy wants to communicate securely with entities affiliated with CA_2, are entities that are indeed affiliated with CA_2. No one else *needs* to know anything about A's affiliation and policy. In fact, an un-authorized observer should not be able to *link* any two instances of the AKE protocol executed by any party. Note that linkability has been recognized as a privacy threat in the context of many applications, and this motivated research into identity escrow [KP98], electronic-cash, e.g. [CFN88], or unlinkable credentials [CL01], among other applications.

In contrast, an *(Unlinkable) Secret Handshake* [SH] scheme is an AKE protocol with the following privacy property: An adversary playing the role of player B, who does not hold a valid certificate $cert^B \in CA^A$ where CA^A is a policy specified by A in this AKE protocol instance, does not learn *anything* about party A. More specifically, the protocol provides (1) *affiliation/policy hiding*, in the sense that without authorization one cannot tell the affiliation and the authentication policy of any party, and (2) *unlinkability*, in the sense that an un-authorized adversary cannot link any two instances of the SH protocol executed by the same player. (Of course, B's privacy against malicious player A is protected in the analogous way.) Note that it makes no sense to require that affiliation/policy hiding holds against "insiders", i.e. parties which *do* satisfy A's policy. However, in some applications it might make sense to require unlinkability from insiders, and even though we do not model such insider-unlinkability formally, our scheme can be modified to support it, although the on-line computation cost would then grow from $2 \log n$ to $O(\Delta \log n)$ exponentiations.

Prior Work on Secret Handshakes. *Linkable* versions of a Secret Handshake scheme were given before, based on bilinear maps [BDS+03, BHS04], computational Diffie-Hellman [CJT04], or RSA [JKT07a, JKT07b]. (As shown in [JKT07b], the RSA-based secret handshake scheme proposed in [Ver05] is insecure because it fails to protect players' affiliations.) All these solutions are efficient and practical, but all of them display two privacy vulnerabilities: First, even though the affiliations and policies of the participants in these schemes are protected in the sense of affiliation/policy hiding, these schemes do not meet the unlinkability property. In fact, instances of the SH protocol executed by a single party can be efficiently linked by any observer. Secondly, these schemes do not protect affiliation privacy of players whose certificates need to be revoked, e.g. in case of key corruption or loss. There have been several proposed solutions to the unlinkable secret handshake problem, but neither of them solves the problem in a satisfactory way. First, all the above schemes have trivially unlinkable variants if players use single-use certificates, but such single-use certificates require too much storage and make revocation impractical. The scheme of Tsudik and Xu [TX05] relies on a group secret shared by all the group members, and thus it requires perfect synchrony in revocation information between the participating players, or otherwise the players fail to authenticate one another. The scheme of Xu and Yung, [XY04] is not based on any shared secrets, but it only offers a weak version of the privacy property called *k-anonymity*. This notion allows the attacker to learn that the participants' affiliation is contained in the set of k publicly revealed CA's. Moreover, since the real affiliation of player A belongs to an intersection of the k-element sets released each time A runs the protocol, protocol instances can be linked with significant probability. Finally, [AB07] proposed an unlinkable scheme but their scheme does support revocation.

Our Contributions: **(1)** The primary contribution of this paper is the first construction of an efficient *unlinkable* Secret Handshake scheme, with no information leakage due to certificate revocation, with no reliance on single-use certificates, with support of revocation, and without the requirement that the both players

assume the same revocation information. The scheme takes only a few communication rounds and its bandwidth is $\Delta \cdot \log n$ standard public key encryption ciphertexts, where n is the upper bound on the number of group members and Δ is a parameter equal to the maximum tolerated *lag* between revocation updates received by the two protocol participants. The computational costs consist of $\Delta \cdot \log n$ off-line exponentiations and $2 \log n$ on-line exponentiations, if the scheme is instantiated with the least expensive CCA-secure variant of ElGamal encryption, e.g. DHAES [ABR01]. For practical values like $\Delta = 10$ and $n \leq 2^{16}$, the bandwidth comes to $20KB$, assuming computational Diffie-Hellman holds on groups of residues of 1024-bit primes, and the on-line computation involves 32 (short) exponentiations with a fixed base, i.e. less than 100 milliseconds.

(2) We provide very strong definitions of both security and privacy for an SH scheme. Security is modeled as in a standard AKE protocol (e.g. [CK02]), and hence our definition implies security against the man-in-the-middle attack, and it offers independence between keys on every session, thus neutralizing session-interference attacks. (This was unknown for the protocols given in [BDS+03, BHS04, CJT04].) Similar AKE-based definition for secret handshakes was given for 2-party secret handshake protocols in [JKT07b], but here we *strengthen* the privacy property so that it includes *unlinkability* of the protocol instances in addition to *affiliation and policy hiding* modeled in [JKT07b].

(3) The main ingredient in our solution is a construction of a *key-private* public-key group key management [PKGKM], which is a stateful version of the public-key broadcast encryption. We show an efficient key-private PKGKM scheme based on the "Logical Key Hierarchy" GKM scheme of Wallner et al. [WHA97], and we show a CCA-secure version of it. Key-privacy for standard encryption has been recognized as an important tool for achieving anonymity and privacy properties in various protocols. A key-private (stateful) *broadcast* encryption might find such applications as well. As a side contribution, we extend results of Bellare et al. [BBs03] on batched encryption, to ElGamal encryption compiled into its CCA-secure version via the Fujisaki-Okamoto construction [FO99].

Note on Intrinsic Limitations of Affiliation/Policy Hiding AKE's. The security notion for AKEs implies complete independence between protocol sessions in the sense that a key agreed on any chosen session remains secure regardless of what happens to keys agreed on all other sessions, including the most extreme case when all these other keys are simply revealed in the clear. In the works on secure AKE's, *e.g.* [CK02], this is modeled by giving the adversary against an AKE scheme an access to a "key revelation" oracle which can reveal keys computed on any protocol session except the session that the adversary is attacking. In particular, this implies that any AKE session remains secure even if the adversary sees whether or not all other protocol sessions were successful or not. (Adversary can do that by revealing the keys computed in these sessions and testing whether they are non-empty and equal.) In contrast to security, the *privacy* property of affiliation/policy hiding *cannot* be achieved if (1) the propagation of a revocation list is not instantaneous, (2) an adversary can engage with any player in a protocol session, and (3) the adversary can observe whether or

not any of these sessions were successful. This is because an active adversary who corrupts some player can find out the affiliation/policy (assume these are equal) of any player A who has not yet updated his or her revocation list. Namely, an adversary can engage in a protocol with A assuming the corrupted player's identity, and since A's revocation list is not updated, the protocol succeeds, and the adversary observes that and learns that A is affiliated with the same group as the corrupted player. The adversary then arranges a session between A and any other player B, e.g. by engaging both players and acting as a man in the middle, and if the adversary observes whether the session was successful, he/she can conclude if B's affiliation/policy matches that of A's. Note that this attack can be staged even if player B always has the most recent revocation list, either because the protocol is supposed to tolerate the lag in A's and B's revocation lists *or* because A might have updated his revocation list before running the protocol with B.

Therefore, an AKE scheme *cannot* protect the affiliation/policy hiding of even the players who always immediately update their revocation lists if (1) there are players who do not, (2) the adversary can engage any player in the protocol, and (3) the adversary can observe if a session is successful. Given this intrinsic limitation to privacy of authentication protocols, we believe that the most useful relaxation that would enable privacy protection in practical applications is to remove the third item. In other words, we must require that the adversary cannot tell an execution of a successful protocol that is executed over the secure channel established by an instance of the AKE scheme, from an execution of a "simulation" of such protocol, which a player will perform whenever an AKE instance fails. In other words, the AKE scheme can offer affiliation/policy privacy but only for a special class of protocols which utilize the keys agreed-upon by this AKE, namely for protocols that can be simulated in this fashion. This class includes, for example, protocols which have fixed number of rounds and whose message sizes can be fixed without big efficiency losses. The privacy-preserving execution of such protocol involves padding every message to the fixed upper bound, and its simulation consists of sending random messages of the same size. In both cases the protocol messages need to be encrypted with key-private symmetric encryption (which standard symmetric encryption schemes provide), but in the simulation the key is chosen at random. We defer the full specification of such privacy-preserving protocols to the full version [JL07].

Organization. We define key-privacy for PKGKM's in Section 2, and Unlinkable SH's in Section 3. In Section 4 we introduce multi-encryption, which we use to build key-private GKM in Section 5, and from that we build an SH scheme in Section 6. All proofs have been delegated to the full version of this paper [JL07].

2 Key-Private Group Key Management: Definition

We describe the syntax of a (Public-Key) Group Key Management [PKGKM] scheme, and we define two PKGKM properties: (1) Semantic security under the chosen-ciphertext attack, IND-CCA, which is an adaptation of the standard

security notion of IND-CCA for standard encryption [GM84] to GKM schemes; and (2) Key-privacy under CCA, IK-CCA, which is also an adaptation of the IK-CCA notion of key-privacy introduced by Bellare et al. [BBDP01] for standard encryption.

In a Public-Key GKM scheme we consider a group of players administered by a *group manager*, who creates a *public* (encryption) key, issues private (decryption) keys to the group members, and can revoke any member by broadcasting a revocation information, which is used to update both the public and the private keys. A PKGKM scheme is a stateful version of a Public-Key Broadcast Encryption scheme, considered e.g. by Dodis and Fazio [DF02] and by Boneh et al. [BBW06]. In a PK BE scheme, the public and private keys are fixed, and the encryptor can encrypt a message for *any subset* of players. In a PKGKM scheme, messages are always encrypted under the most recent public key, and the revocation information used in computing this key determines the subset of players who can decrypt.

A PKGKM scheme is a tuple of algorithms (Setup, KGen, Revoke($\mathsf{PKUpdate}$, $\mathsf{SKUpdate}$), Enc, Dec):

- $\mathsf{Setup}(1^k)$, on input a security parameter k, generates parameters params.
- $\mathsf{KGen}(\mathsf{params})$, executed by the group manager, generates the initial group public key $PK^{(0)}$, the initial master secret $\mathsf{MSK}^{(0)}$, and members' initial private keys $SK_1^{(0)}, ..., SK_n^{(0)}$.
- $\mathsf{Revoke}(\mathsf{MSK}^{(t)}, i, t)$, executed by the group manager in epoch t (initially $t = 0$) revokes key SK_i of member U_i by generating an update message $U^{(t+1)}$, and updating the master secret to $\mathsf{MSK}^{(t+1)}$. Message $U^{(t+1)}$ is used to update the public key as $PK^{(t+1)} = \mathsf{PKUpdate}(PK^{(t)}, U^{(t+1)})$, and each decryption key as ${SK_j}^{(t+1)} = \mathsf{SKUpdate}({SK_j}^{(t)}, U^{(t+1)})$, for each $j \neq i$.
- $\mathsf{Enc}(PK^{(\mathsf{et})}, m)$ encrypts message m on public key $PK^{(\mathsf{et})}$.
- $\mathsf{Dec}({SK_i}^{(\mathsf{dt})}, C)$ is a decryption algorithm which either returns some message m or rejects.

Initialization of the Static Adversary: To express security properties of the PKGKM scheme in the *static* adversary model, it's convenient to denote the initialization pattern for the static adversary who corrupts subset Rev of players, on public parameters params, as $\mathsf{Init}(\mathsf{params}, \mathsf{Rev})$. This initialization involves an execution of $\mathsf{KGen}(\mathsf{params})$ and t executions of Revoke, for t from 1 to $\tau = |\mathsf{Rev}|$, which generates the initial public/private keys and t update messages $U^{(1)}, \ldots, U^{(\tau)}$, which in turn define a set of public/private keys $PK^{(t)}$ and ${SK_i}^{(t)}$, for all $i \notin \mathsf{Rev}$ and $0 \leq t \leq \tau$. The static adversary receives params, the public keys, and the private keys of the players in Rev. In the key-privacy definitions, when we initialize the keys of two groups G_0, G_1, on the same public parameters params, we'll generate the two sets of keys for these two groups as $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_0, 0)$ and $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_1, 1)$.

Δ-Limited Completeness. We stress that we do *not* assume that the key update messages are propagated immediately to all the participants, and hence there

can be a discrepancy between the epoch of the public key used by the encryptor and the epoch of the private key used by the decryptor. The PKGKM scheme we construct in this paper handles such discrepancy between the epochs up to some value Δ, and otherwise it does not guarantee proper decryption. In practice the group members will have to retrieve update messages in an anonymous way, e.g. using onion-routing, often enough to offset this Δ-limitation on the tolerated epoch difference.

Let $\mathsf{params} = \mathsf{Setup}(1^k)$ and all the private/public keys are generated via $\mathsf{Init}(\mathsf{params},\mathsf{Rev},\epsilon)$. Let $\mathsf{Rev}(t)$ denotes the first t indices in Rev. We call a PKGKM scheme Δ-*Limited* ϵ_{comp}-*Complete* if it holds with probability at least $1-\epsilon_{comp}$ that for any message m, for any $0 \leq \mathsf{et}, \mathsf{dt} \leq \tau$ and $i \notin \mathsf{Rev}(max\{\mathsf{et},\ \mathsf{dt}\})$, if $|\mathsf{et}-\mathsf{dt}| \leq \Delta$ then

$$\mathsf{Dec}(SK_i^{(\mathsf{dt})}, C) = m \quad \text{if} \quad C = \mathsf{Enc}(PK^{(\mathsf{et})}, m)$$

IND-CCA Security and IK-CCA Privacy for PKGKM's. We define the IND/IK-CCA security notions for a PKGKM scheme only for *static adversaries*. Both notions are analogous to the IND-CCA security notion and the IK-CCA key-privacy notion for standard public key schemes. The differences in the IND-CCA game for a PKGKM scheme proceeds are as follows: (1) The game proceeds on some set Rev of corrupted players chosen before the game starts; (2) The adversary is given the initial group public key together with $\tau = |\mathsf{Rev}|$ update messages and the private keys of the players in Rev, as generated by $\mathsf{Init}(\mathsf{params}, \mathsf{Rev})$; (3) The queries to the decryption oracle the adversary can make are of the form (C, i, t), where $i \notin \mathsf{Rev}$ and $t \leq \tau$, and the decryption oracle responds with $\mathsf{Dec}(SK_i^{(t)}, C)$; (4) The encryption challenge, on adversarially chosen pair of messages (m_0, m_1), is computed as $C^* = \mathsf{Enc}(PK^{(\tau)}, m_b)$; and (5) After receiving C^* the adversary cannot make decryption queries of the form (C^*, i, t) for any i, t. As in the standard IND-CCA notion, we say that the PKGKM scheme is IND-CCA if the probability that the adversary guesses b is at most negligibly larger than $1/2$.

The IK-CCA game for a PKGKM scheme is defined in a similar way, except that the adversary is given two sets of private/public keys, for groups G_0 and G_1, the adversary can access decryption oracles for both groups, the encryption challenge is computed as $C^* = \mathsf{Enc}({PK_b}^{(\tau)}, m)$ on adversarially chosen message m, and after receiving C^* the adversary cannot ask for decryption of C^* to either group. Again, the PKGKM scheme is IK-CCA if the probability that the adversary guesses b is at most negligibly larger than $1/2$. We provide both definitions in the full version of this paper on eprint [JL07].

3 Unlinkable Secret Handshake Scheme: Definition

Similarly as in the Group Key Management setting, the model for a Secret Handshake scheme consists of a set of groups $\mathcal{G}$, each managed by its manager, and a set of users $\mathcal{U}$. For notational convenience in describing the security *model* of a Secret Handshakes scheme, we'll assume that every user $U \in \mathcal{U}$ is a member of a unique group $G \in \mathcal{G}$, which we'll denote $G = \mathsf{Membership}(U)$. We say that

$U \in G$ if $G = \mathsf{Membership}(U)$. Each $U \in G$ has its unique index in this group, denoted $i = \mathsf{Index}(U)$. We stress that the restriction that each user is a member of a unique group is taken only for notational convenience in defining the security and privacy properties of a Secret Handshake scheme. It is *not* a restriction on the actual applications of such scheme, where the same user can indeed be a member of many groups.

Syntax of a Secret Handshake Scheme. We define a *Secret Handshake* (SH) scheme as a tuple of algorithms (Setup, KGen, Revoke($\mathsf{PKUpdate}$, $\mathsf{SKUpdate}$), $\mathsf{Handshake}$), where all algorithms except $\mathsf{Handshake}$ have the same syntax as in a GKM scheme. As in the GKM scheme, we'll denote by Init the *static* initialization pattern involving an execution of KGen followed by $|\mathsf{Rev}|$ instances of Revoke, one per each element in the set Rev of revoked players, chosen beforehand.

The new procedure, $\mathsf{Handshake}$, is an interactive protocol executed by any user U, on public inputs params and private inputs (SK, TPK, r), where $SK = SK_{G,i}{}^{(\mathsf{dt})}$ for $G = \mathsf{Membership}(U)$, $i = \mathsf{Index}(U)$, dt is U's current epoch for group G, $TPK = PK_{TG}{}^{(\mathsf{et})}$ is a public key of some "target" group TG, not necessarily equal to G, at some epoch et, and r is the *role* of U in this execution, which is equal either to init for *initializer* or resp for *responder*. Since it's an interactive protocol, it is intended that two users, U and U', execute two *matching* instances of the $\mathsf{Handshake}$ protocol and exchange the messages generated on these protocol instances. Adopting the standard terminology for Key Agreement protocols, we refer to every instance of $\mathsf{Handshake}$ protocol as a *session*, and we'll call the matching instances of this protocol *matching sessions*. We will denote an instance of the protocol executed by θ-th instance of user U as Π_U^θ. Consider the session instance Π_U^θ running on inputs (SK, TPK, r), and instance $\Pi_{U'}^{\theta'}$ running on inputs (SK', TPK', r'). We call these two sessions *matching* if (1) $r \neq r'$, (2) $\mathsf{Membership}(U) = \mathsf{PKGroup}(TPK')$, and (3) $\mathsf{Membership}(U') = \mathsf{PKGroup}(TPK)$, where $\mathsf{PKGroup}(PK)$ identifies the group of the public key. We call two matching instances Π_U^θ and $\Pi_{U'}^{\theta'}$ *partnered* if the protocol transcript on these two sessions are the same, i.e. if the messages sent by Π_U^θ are delivered to $\Pi_{U'}^{\theta'}$ and vice versa. Two partnered sessions should both accept and output the same authenticated and random key K, which can be then used for any subsequent secure communication. If a session instance does not output a key then it rejects.

Δ-Limited Completeness: Informally, we require that if U and U' run matching instances of the $\mathsf{Handshake}$ protocol, i.e. U is a member of group G and wants to authenticate itself to members of group TG, while U' is a member of group $G' = TG$ and wants to authenticate itself to members of group $TG' = G$, then both players accept and output the same key on this session. However, we can only guarantee such completeness property if neither of the two players is revoked from their respective groups and if the epochs of the public and private keys these players use are no farther than Δ apart, for both groups.

Formally, set $\mathsf{params} = \mathsf{Setup}(1^k)$, and initialize two groups $G = 0$ and $G = 1$ on two sets $\mathsf{Rev}_0, \mathsf{Rev}_1 \subset \{1, ..., n\}$ with the static initialization pattern $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_0, 0)$ and $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_1, 1)$. Let $\mathsf{Rev}_G(t)$ denote the first

t indices in Rev_G and let $\tau_G = |\mathsf{Rev}_G(\tau)|$. We call an SH scheme *$\Delta$-Limited ϵ-Complete* if the following holds: For any $0 \leq \mathsf{et}_G, \mathsf{dt}_G \leq \tau_G$ s.t. $|\mathsf{et}_G - \mathsf{dt}_G| \leq \Delta$, and any $i_G \notin \mathsf{Rev}_G(max\{\mathsf{et}_G,\ \mathsf{dt}_G\})$, for both $G = 0$ and $G = 1$, if U and U' run two sessions of the Handshake protocol on inputs

$$U\text{'s inputs: } (SK_{0,\,i_0}^{(\mathsf{dt}_0)}, PK_1^{(\mathsf{et}_1)}, \mathsf{init}) \qquad U'\text{'s inputs: } (SK_{1,\,i_1}^{(\mathsf{dt}_1)}, PK_0^{(\mathsf{et}_0)}, \mathsf{resp})$$

then if all the protocol messages are properly exchanged between these two instances, then with probability $1 - \epsilon$ they both output a common key K.

Security of a Secret Handshake Scheme: Denote a set of groups as $\mathcal{G}$, a set of users $\mathcal{U}$ partitioned between these groups via the Membership function, and for each group $G \in \mathcal{G}$ some set $\mathsf{Rev}_G \subset \{1, ..., n\}$. We define SH security via a game between an adversary $\mathcal{A}$ and the challenger simulating a network of players, on common input $(1^k, \mathcal{U}, \mathcal{G}, \{\mathsf{Rev}_G\}_{G \in \mathcal{G}})$. First, the challenger picks $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$ and initializes each group $G \in \mathcal{G}$ by the static initialization procedure $\mathsf{Init}(\mathsf{params}, Rev_G, G)$, which produces all the public/private keys and gives $\mathcal{A}$ all the public keys, update messages, and the private keys of the revoked members in each group. After that, $\mathcal{A}$ adaptively issues to the challenger any number of commands of the following type, and outputs a single bit when it's done:

- [Handshake, U, dt, PK_{TG}^{et}, r]: The challenger initializes an instance Π_U^θ where θ is the next index that has not yet been used for player U, and then starts the protocol for user U in epoch dt, i.e. runs Handshake on inputs $(SK_{G,i}^{(\mathsf{dt})}, PK_{TG}^{\mathsf{et}}, r)$ where $G = \mathsf{Membership}(U)$, $i = \mathsf{Index}(U)$, $\mathsf{dt} \leq |\mathsf{Rev}_G|$, and $\mathsf{et} \leq |\mathsf{Rev}_{TG}|$. The challenger keeps the state of the instance, and hands to the adversary any message it generates.
- [Message, U, θ, m]: The challenger wakes up the Π_U^θ instance of the Handshake protocol on message m, if such instance exists, and follows the protocol on behalf of that instance. If the instance outputs another message, the challenger hands it to the adversary. If the instance rejects, the challenger abandons it. If the instance outputs a key, the challenger records it by saving the tuple $(\Pi_U^\theta, PK_{TG}^{\mathsf{et}}, K)$.
- [Reveal, U, θ]: If either the Π_U^θ session or some session partnered with Π_U^θ was *tested* (see below), or if session Π_U^θ did not output a key, the challenger returns $\perp$. Otherwise, the challenger returns the key K output on this session to $\mathcal{A}$.
- [Test, U, θ]: The challenger picks a random bit b. If this is the only Test query $\mathcal{A}$ makes, if session Π_U^θ outputted some key K, if the adversary has not revealed the key on either Π_U^θ *or* some session $\Pi_{U'}^{\theta'}$ partnered with Π_U^θ, and if Π_U^θ executed on the target public key PK_{TG}^{et} s.t. $\mathsf{et} = |\mathsf{Rev}_{TG}|$ (i.e. if the target public key PK on that session is updated so that all adversary's keys in group TG are revoked), then the challenger returns K to $\mathcal{A}$ if $b = 0$, or a random string of length $|K|$ if $b = 1$. If any of these conditions are not met, the challenger returns $\perp$.

Let $\mathsf{AdvSH}_{\mathcal{A},k}$ denote the probability that $\mathcal{A}$ outputs the correct bit b chosen by the challenger. (If $\mathcal{A}$ never makes the Test query we pick b at random.) We

say that a SH scheme is $(t, \epsilon, n, n', R, q_S)$-secure if for $|\mathcal{U}| = n$, $|\mathcal{G}| = n'$, for maximum R sessions per user, for all subsets $\{\mathsf{Rev}_G\}_{G\in\mathcal{G}}$, and all algorithms $\mathcal{A}$ running in time t which invoke a total of q_S SH protocol instances, we have $|\mathsf{AdvSH}_{\mathcal{A},k} - \frac{1}{2}| < \epsilon$.

Unlinkability and Affiliation/Policy-Hiding of a Secret Handshake Scheme: We define unlinkability and affiliation /policy-hiding similarly to SH-security, via a game between an adversary $\mathcal{A}$ and a challenger on common input $(1^k, \mathcal{U}, \mathcal{G})$ and the specification of corrupt (and revoked) players $\{\mathsf{Rev}_G\}_{G\in\mathcal{G}}$). As in the SH-security game, the challenger picks $\mathsf{params} \leftarrow \mathsf{Setup}(1^k)$ and initializes each group by $\mathsf{Init}(\mathsf{params}, \mathsf{Rev}_G, G)$. After the initialization, the challenger picks a random bit $b \in \{0, 1\}$. Since our AKE protocols protect privacy only for the privacy-preserving protocols (see the note in the introduction), we model the privacy adversary without access to a key-revealing oracle. In the full version of the paper [JL07] we give a full model where the adversary has an access to a protocol-execution oracle instead, and privacy of an AKE scheme holds only if the protocol is privacy-preserving and compiled using a privacy simulator. In these proceedings we simply restrict the privacy adversary $\mathcal{A}$ to issuing only the Handshake and Message commands to the challenger.

The challenger in this privacy game services $\mathcal{A}$'s commands depending on bit b the challenger chooses: If $b = 0$ then the challenger executes each command by following the corresponding protocol on behalf of the user entities, as in the SH-security game. However, if $b = 1$, the challenger uses a special interactive machine SIM to serve the commands issued to the instances whose target keys are updated so that all adversary's keys are revoked. For other instances, the challenger follows the protocol on behalf of the user as in the SH-security game. The SIM machine is initialized on string params, it can keep state between invocations, but has no access to group keys created by the challenger in all the Init instances. The challenger executes as follows using SIM:

- [Handshake, U, dt, PK^{et}_{TG}, r]: If the target key PK^{et}_{TG} satisfies $\mathsf{et} = |\mathsf{Rev}_{TG}|$ (i.e. if it is updated so that all adversary's keys in group TG are revoked), then the challenger picks a next index θ that has not been used yet by U and an additional globally unique (random) string $\hat{\theta}$, which we will call an *identifier* for the Π^{θ}_U session. The challenger hands $[\mathsf{Handshake}, \hat{\theta}, \hat{\theta}\text{-list}]$ to SIM where $\hat{\theta}$-list contains the identifiers of all sessions which match Π^{θ}_U.

  If $\mathsf{et} < |\mathsf{Rev}_{TG}|$, the challenger initializes an instance Π^{θ}_U for θ the next index not used by U, runs the Handshake protocol on behalf of Π^{θ}_U as in the SH-security game, and hands any message generated by Π^{θ}_U to the adversary.
- [Message, U, θ, m]: If [Handshake, U, dt, PK^{et}_{TG}, r] has been issued and $\mathsf{et} = |\mathsf{Rev}_{TG}|$, the challenger retrieves $\hat{\theta}$ identifier for this session, passes $[\mathsf{Message}, \hat{\theta}, m]$ to SIM, and forwards SIM's answer to $\mathcal{A}$. If $\mathsf{et} < |\mathsf{Rev}_{TG}|$, the challenger follows the real protocol on behalf of Π^{θ}_U as in the SH-security game, and hands any message generated by Π^{θ}_U to the adversary.

Let $\mathsf{AdvSH}_{\mathcal{A},k}$ denote the probability that $\mathcal{A}$ outputs the correct bit b chosen by the challenger in the above game. We say that a SH scheme is $(t, \epsilon, n, n', q_S)$-*unlinkable and affiliation/policy hiding* if there exists a simulator algorithm SIM running in time polynomial in k s.t. for $|\mathcal{U}| = n$, $|\mathcal{G}| = n'$, for R setting the maximum number of sessions per user, for all subsets $\{\mathsf{Rev}_G\}_{G\in\mathcal{G}}$, and all algorithms $\mathcal{A}$ running in time t which invoke a total of q_S SH protocol instances, we have $|\mathsf{AdvSH}_{\mathcal{A},k} - \frac{1}{2}| < \epsilon$.

4 Security and Key-Privacy for Batched Encryption

Our construction of a key-private (Public-Key) Group Key Management scheme (PKGKM), is based on the so-called "multi-encryption" [ME] introduced by Bellare et al. [BBs03]. A multi-encryption is a non-standard method of encrypting a message under many independent public keys, where the encryptor uses the *same randomness* when encrypting the message under each key. The reason we rely on multi-encryption instead of standard encryption in our key-private group key management scheme construction, is that it reduces the cost of decryption procedure from $O(N)$ exponentiations to $O(1)$, where N is the number of component ciphertexts.

Here is why: In our key-private (public-key) group key management scheme [PKGKM], the ciphertext is a vector of ciphertexts. We cannot tag any information that links each component to the public key, as that would leak information about the group. As a consequence, the decrypting party will not know which of these component ciphertexts it should decrypt, and hence it can decrypt the PKGKM ciphertext only in an *oblivious* way, i.e. by attempting to decrypt each of the component (standard) ciphertexts. Now, if the N component ciphertexts are computed in a standard way, such oblivious decryption would take $O(N)$ exponentiations. However, if the component ciphertexts are computed using the same randomness vector, then the decryption procedure requires only $O(1)$ exponentiations and $O(N)$ fast symmetric operations, e.g. xors and tests for equality.

In the rest of this section we (1) define a multi-encryption [ME] version of a public-key encryption scheme, and the IND-CCA and IK-CCA notions for it. Then we (2) define a version of multi-encryption in which decryption is *oblivious*, and we discuss converting an ME to an oblivious multi-encryption [OME]. Finally, (3) we show examples of IND/IK-CCA encryption schemes which yield IND/IK-CCA ME and OME schemes.

Multi-Encryption and its Security and Privacy Properties. We define multi-encryption as a version of a standard public-key encryption, where the same message is encrypted under a set of public keys, and the encryptor uses the same randomness in each encryption. A standard public-key encryption is given by a triple of algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, but for the purpose of multi-encryption we need to split the key-generation procedure into two parts, Setup, which generates some public parameters params, e.g. $\mathsf{params} = (p, q, g)$ where p, q are primes and g is an element of order q in $\mathbb{Z}_p^*$, and $\mathsf{KGen}(\mathsf{params})$ proper, which for example picks private key x at random in $\mathbb{Z}_q$ and sets the public key as $y = g^x \bmod p$.

The multi-encryption Π^{ME} version of the public key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{ME})$, (note that the setup does not change), where:

$\mathsf{KGen}_{ME}(\mathsf{params})$ executes $\mathsf{KGen}(\mathsf{params})$ n times to produce a vector of n public keys $\boldsymbol{pk} = \{pk_i\}_{i=1,\ldots,n}$ and the corresponding private keys $\boldsymbol{sk}$.
$\mathsf{Enc}_{ME}(\boldsymbol{pk}, m)$ picks a (long-enough) random string r and outputs a vector of ciphertexts $\boldsymbol{c} = \{c_i\}_{i=1,\ldots,n}$ where $c_i = \mathsf{Enc}(pk_i, m; r)$ [i.e. the encryption of m under key pk_i using randomness r].
$\mathsf{Dec}_{ME}(sk_i, \boldsymbol{c})$ outputs $\mathsf{Dec}(sk_i, c_i)$ where $\boldsymbol{c} = (c_1, \ldots, c_n)$.

The IND-CCA notion of security and the IK-CCA notion of key-privacy for a multi-encryption scheme Π^{ME} is defined analogously to the IND-CCA security and IK-CCA key-privacy definitions for the underlying standard encryption scheme Π. In the security notion (IND-CCA), the difference between security of multi-encryption and standard encryption is that (1) in the multi-encryption case the adversary receives a vector of public keys $\boldsymbol{pk}$ instead of a single key pk, (2) the challenge ciphertext is a multi-encryption vector $\boldsymbol{c}^* = \mathsf{Enc}_{ME}(\boldsymbol{pk}, m_b)$ for the randomly chosen message m_b instead of a single ciphertext $c^* = \mathsf{Enc}(pk, m_b)$, and (3) the adversary has an adaptive access to a "flexible" decryption oracle, which takes as input the index i of the decryptor and the ciphertext vector $\boldsymbol{c} = \{c_i\}$, and outputs $\mathsf{Dec}(sk_i, c_i)$. Also, after the adversary sees the encryption challenge ciphertext $\boldsymbol{c}^* = (c_1^*, ..., c_n^*)$ the adversary's queries $(i, \boldsymbol{c})$ must satisfy $c_i \neq c_i^*$ where $\boldsymbol{c} = (c_1, ..., c_n)$.

The changes between IK-CCA notion for multi-encryption and the IK-CCA notion for standard encryption are analogous. We note that the above IND-CCA and IK-CCA notions for multi-encryption scheme are for a *static adversary*, which cannot corrupt parties after the protocol starts.

Oblivious Multi-Encryption. An *oblivious* multi-encryption scheme is a multi-encryption scheme as described above, except that the decryptor is not told which ciphertext is directed to him. In other words, an oblivious multi-encryption scheme Π^{OME} is a version of the public key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{OME})$, where Setup is as in the underlying standard encryption Π, algorithms $\mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}$ are as in the multi-encryption scheme Π^{ME} formed from Π as defined above, and the decryption procedure $\mathsf{Dec}_{OME}(sk_i, \boldsymbol{c})$ proceeds differently than $\mathsf{Dec}_{ME}(sk_i, \boldsymbol{c})$: Namely, if $\boldsymbol{c} = (c_1, \ldots, c_n)$, the decryption procedure computes $m_j \leftarrow \mathsf{Dec}(sk_i, c_j)$, for each j ranging from 1 to n. If all m_j's are equal to the rejection symbol $\perp$, then $\mathsf{Dec}_{OME}(sk_i, \boldsymbol{c})$ outputs $\perp$ as well. Otherwise, it outputs the first m_j s.t. $m_j \neq \perp$.

The IND-CCA and IK-CCA notions for oblivious multi-encryption scheme Π^{OME} are very similar to those for the (non-oblivious) multi-encryption Π^{ME} (see above), and the only difference is in the adversary's interaction with the (flexible) decryption oracle: First, the decryption oracle implements the Dec_{OME} procedure instead of Dec_{ME}. Second, after getting the encryption challenge $\boldsymbol{c}^* = (c_1^*, \ldots, c_n^*)$, the decryption query $(i, \boldsymbol{c})$ made by the adversary must satisfy $\boldsymbol{c} \neq \boldsymbol{c}^*$. (Note that $\boldsymbol{c}$ can contain one or more component ciphertexts of c^*, as long

as $\boldsymbol{c} \neq \boldsymbol{c}^*$.) Analogous changes are made for the IK-CCA notion of an oblivious multi-encryption scheme.

Constructing IND+IK-CCA Secure and Complete Oblivious Multi-Encryption Schemes. Consider the following conversion from an OME scheme $\Pi^{OME} = (\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{ME}, \mathsf{Dec}_{OME})$ to another OME scheme $\Pi'^{OME} = (\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}'_{ME}, \mathsf{Dec}'_{OME})$, where $\mathsf{Enc}'_{ME}(\boldsymbol{pk}, m)$ picks random $r \leftarrow \{0,1\}^k$ for the security parameter k, computes $\boldsymbol{c} \leftarrow \mathsf{Enc}_{ME}(\boldsymbol{pk}, (m, r))$ and outputs ciphertext $C = (\boldsymbol{c}, H(\boldsymbol{c}, m, r))$, where $H(\cdot) \rightarrow \{0,1\}^{2k}$ is a hash function (modeled as a random oracle in the security analysis); and $\mathsf{Dec}'_{OME}(sk, C)$ parses ciphertext C as $(\boldsymbol{c}, h)$, and outputs $\hat{m}$ if $(m, r) \leftarrow \mathsf{Dec}_{OME}(sk, \boldsymbol{c})$ s.t. $h = H(\boldsymbol{c}, \hat{m}, \hat{r})$.

Theorem 1. *If Π^{ME} is an IND+IK-CCA ME scheme, then Π'^{OME} is an $(n2^{-k})$-complete IND+IK-CCA OME scheme in ROM.*

Examples of IND+IK-CCA Oblivious Multi-Encryption Schemes. Bellare et al. showed a generic method for converting IND-CCA standard encryption into an IND-CCA ME. It required a technical property of "reproducibility" of the underlying encryption.[1] It's easy to extend their results to IK privacy. I.e., the same reproducibility implies that IK-CCA encryption yields IK-CCA ME. By combining the results of [BBDP01] and [BBs03] with the discussion above, this yields IND/IK-CCA and complete OME from DHAES and Cramer-Shoup. We extend these results in the following sense: Fujisaki and Okamoto showed a way to convert one-way encryption schemes, with additional technical property of γ-uniformity, into IND-CCA encryption schemes, in the ROM model, via a hybrid with symmetric encryption [FO99]. Their main theorem can be easily extended to cover also key privacy. We refer to the full paper [JL07] for details.

5 Key-Private PKGKM from Oblivious Multi-encryption

Our key-private public-key group key management scheme is based on the so-called *Wallner Tree* key distribution scheme proposed by Wallner et al [WHA97] [WGL98], which uses a binary tree to assign subsets of keys to group members. Our key of a node in the tree is a pair of public/private keys $(pk_z{}^{(t)}, sk_z{}^{(t)})$ in epoch t. We could encrypt a message under the top key $pk_\varepsilon{}^{(t)}$, as it is known to all members of the group. However, such scheme would work only if the public key used by the encryptor has the same epoch as the private key used by the decryptor, e.g. if both parties have the most recent key-update message.

Such synchrony assumption is not realistic in practice. On the other hand, we can relax this assumption and construct a practical group key management scheme with Δ-limited completeness, i.e. a scheme which works assuming a limit Δ on the discrepancy between the key epochs assumed by the encryptors and decryptors. Our way to use the set of keys is similar to the extension of the

[1] We note that the notion of multi-encryption introduced in [BBs03] is stronger than here. Namely, the messages encrypted for each public key need not be the same.

Wallner Tree construction in which any subset of Δ players can be revoked in a batch. Let $\mathsf{Rev}^{(t)}$ be a set of Δ indices corresponding to Δ most recently revoked users. Let $\mathsf{co\text{-}path}(u)$ be the co-path of user u and $\widetilde{\mathcal{R}}^{(t)}$ be a $\Delta \times \log n$ table of indices whose i-th column is made of indices in set $\mathsf{co\text{-}path}(r_i)$ for i-th element r_i in $\mathsf{Rev}^{(t)}$. In any column i, the element in row j is the j-bit long element in $\mathsf{co\text{-}path}(r_i)$. Let $\mathcal{R}^{(t)}$ be a transformation of the $\widetilde{\mathcal{R}}^{(t)}$ matrix, where every element of $\mathcal{R}^{(t)}$ which is a prefix of some index i in the revoked set $\mathsf{Rev}^{(t)}$ is replaced by a special symbol $\star$. For example, if $n = 16$, $\Delta = 3$, and $Rev^{(t)} = \{000, 011, 101\}$ then

$$\widetilde{\mathcal{R}}^{(t)} = \begin{pmatrix} 1 & 1 & 0 \\ 01 & 00 & 11 \\ 001 & 010 & 100 \\ 0000 & 0111 & 1010 \end{pmatrix} \quad \longrightarrow \quad \mathcal{R}^{(t)} = \begin{pmatrix} \star & \star & \star \\ \star & \star & 11 \\ 001 & 010 & 100 \\ 0000 & 0111 & 1010 \end{pmatrix} \tag{1}$$

In this way, matrix $\mathcal{R}^{(t)}$ consists of nodes which cover all the leaves except those in set $\mathsf{Rev}^{(t)}$. Therefore, if the encryptor encrypted a message m under public keys ${pk_z}^{(t-\Delta)}$ for all indices $z \in \mathcal{R}^{(t)}$, then every node except those in $\mathsf{Rev}^{(t)}$ would be able to get m using its key from epoch $t - \Delta$. Let $PK^{(t)}$ be a $\log n \times \Delta$ table of public keys from epoch $t - \Delta$ corresponding to the indices in $\mathcal{R}^{(t)}$. If some entry in $\mathcal{R}^{(t)}$ is a symbol "$\star$", then the entry in the same position of $PK^{(t)}$ is also a "$\star$". Continuing the above example we have

$$PK^{(t)} = \begin{pmatrix} \star & \star & \star \\ \star & \star & pk_{11}^{(t-3)} \\ pk_{001}^{(t-3)} & pk_{010}^{(t-3)} & pk_{100}^{(t-3)} \\ pk_{0000}^{(t-3)} & pk_{0111}^{(t-3)} & pk_{1010}^{(t-3)} \end{pmatrix} \tag{2}$$

We can then encrypt a message m under $PK^{(t)}$ using an OME scheme, as $\mathcal{C}^{(t)}$ given below. For the entries in $PK^{(t)}$ that are marked "$\star$", the corresponding ciphertexts are filled with randomness, in the format of real ciphertexts.

$$\mathcal{C}^{(t)} = \begin{pmatrix} \$ & \$ & \$ \\ \$ & \$ & Enc_{pk_{11}^{(t-3)}}(m) \\ Enc_{pk_{001}^{(t-3)}}(m) & Enc_{pk_{010}^{(t-3)}}(m) & Enc_{pk_{100}^{(t-3)}}(m) \\ Enc_{pk_{0000}^{(t-3)}}(m) & Enc_{pk_{0111}^{(t-3)}}(m) & Enc_{pk_{1010}^{(t-3)}}(m) \end{pmatrix} \tag{3}$$

Then each user U_i for $i \notin \mathsf{Rev}^{(t)}$ could use its key set of epoch $t - \Delta$, *i.e.* $\{{sk_z}^{(t-\Delta)} : z \in \mathsf{path}(i)\}$, to decrypt the message m. (Actually, $\{{sk_z}^{(t-\Delta)} : z \in \mathsf{path}(i)\}$ remain the same until epoch t.)

However, instead of requiring the decryptor to be Δ epochs behind the encryptor, we want to tolerate any lag between the encryptor and decryptor epochs, et and dt, as long as $|\mathsf{et} - \mathsf{dt}| \leq \Delta$. To do this, we will make each member of the group store its key set for (Δ+1) consecutive epochs. Each player's secret key $SK_i^{(t)}$ is a $\log n \times (\Delta{+}1)$ key table whose j-th column is a key set of user U_i in epoch $\tau = t - (\Delta{+}1){+}j$. The keys in each column are arranged so that ${sk_z}^{(\tau)}$,

for $\tau \in \{t-\Delta, t\}$, is in row number $|z|$, *i.e.* the bit-length of index z.[2] For the same example above, the node 0111 has the secret key table:

$$SK_{0111}^{(t)} = \begin{pmatrix} sk_0^{(t-3)} \neq sk_0^{(t-2)} \neq sk_0^{(t-1)} = sk_0^{(t)} \\ sk_{01}^{(t-3)} = sk_{01}^{(t-2)} \neq sk_{01}^{(t-1)} = sk_{01}^{(t)} \\ sk_{011}^{(t-3)} = sk_{011}^{(t-2)} \neq sk_{011}^{(t-1)} = sk_{011}^{(t)} \\ sk_{0111}^{(t-3)} = sk_{0111}^{(t-2)} = sk_{0111}^{(t-1)} = sk_{0111}^{(t)} \end{pmatrix} \quad (4)$$

where $\neq$ means the key changed from one epoch to the next one.

Let $(\mathsf{Setup}, \mathsf{KGen}_{ME}, \mathsf{Enc}_{OME}, \mathsf{Dec}_{OME})$ be an IND/IK-CCA secure and complete Oblivious Multi-Encryption scheme. The IND/IK-CCA and Δ-complete PKGKM scheme can be constructed as follows:

- $\mathsf{Setup}(1^k)$: Output $(\mathsf{params}, n, \Delta)$, where $\mathsf{params} \leftarrow \mathsf{Setup}$, n is the maximum number of members in each group (assumed to be a power of 2), and Δ is the maximum difference between the encryptor's epoch and the decryptor's epoch for which we guarantee correctness of decryption.
- $\mathsf{KGen}(\mathsf{params})$: For Wallner Tree of depth $\log n$. Use $\mathsf{KGen}_{ME}(\mathsf{params})$ to generate a set of public/private key pairs $(pk_z{}^{(0)}, sk_z{}^{(0)})$ for all tree nodes z. Store all these key pairs as $\mathsf{MSK}^{(0)}$. To simplify the description of the key generation process, we let the group manager revoke Δ dummy members, so that $PK^{(0)}$ has the format as eq.(2), and user U_i's key $SK_i{}^{(0)}$ is a $\log n \times (\Delta{+}1)$ key-table, whose j-th column is filled with keys $\{sk_z{}^{(j-\Delta-1)} \mid z \in \mathsf{path}(i), z \neq \varepsilon\}$, as in eq.(4).
- $\mathsf{Revoke}(\mathsf{MSK}^{(t-1)}, r^{(t)})$: The update message $U^{(t)}$ consists of the standard Wallner Tree update message which revokes user $r^{(t)}$ (and updates both the standard Wallner Tree keys and the keys $pk_z^{(t-1)}, sk_z^{(t-1)}$ for tree-nodes z on the path from the root to the leaf corresponding to player $r^{(t)}$. Additionally, the update message contains also the $PK^{(t)}$ table as in eq.(2). The table contains a set of keys from epoch $t-\Delta$ which is determined by the tree-leaves assigned to the last Δ revoked members.
- $\mathsf{PKUpdate}(U^{(t)})$: Extract $PK^{(t)}$ from $U^{(t)}$.
- $\mathsf{SKUpdate}(SK_i{}^{(t-1)}, U^{(t)})$: User U_i extracts the key update part from $U^{(t)}$, perform the Wallner Tree user key update to get the new set of keys of epoch t, i.e. $\{sk_z{}^{(t)}\}$, for z in the key path of U_i. Denote $\mathbf{sk}_i{}^{(t)}$ as the resulting Wallner-tree secret key set for user U_i in epoch t. Then U_i discards $\mathbf{sk}_i{}^{(t-\Delta-1)}$ from the first column of his key table $SK_i{}^{(t-1)}$, and append $\mathbf{sk}_i{}^{(t)}$ to the last column, so that $SK_i{}^{(t)}$ in the format of eq.(4).
- $\mathsf{Enc}(PK^{(t)}, m)$: Compute $C \leftarrow \mathsf{Enc}_{OME}(PK^{(t)}, m)$ and format it as an $\log n \times \Delta$ table, as in eq.(3).

[2] In this way all the keys in U_i's key set are present in column τ except of the group secret $sk_\varepsilon{}^{(\tau)}$. The users could use these group secrets too, but this makes the description of the scheme slightly more complicated, and it does not improve the performance by much.

– $\mathsf{Dec}(SK_i^{(t)}, C)$: For each key sk in the left-most column $\mathbf{sk}_i^{(t-\Delta)}$ in U_i's key table $SK_i^{(t)}$, from top level down, compute $m \leftarrow \mathsf{Dec}_{OME}(sk, C)$. If $m \neq \perp$, output m. If the trials fail for all $sk \in \mathbf{sk}_i^{(t-\Delta)}$, then repeat the above procedure for the keys in the right-most column $\mathbf{sk}_i^{(t)}$ in $SK_i^{(t)}$. If all trials fail, output $\perp$.

Note on Efficiency. The GKM encryption cost is a multi-encryption with $\Delta \log n$ keys, i.e. $\Delta \log n$ standard encryptions. The GKM decryption cost is $O(\log n)$ *multi-encryption* decryptions. We can reduce both costs by modifying the OME scheme so that the OME scheme is used as Key Encapsulation, to encrypt a random key k for an IND-CCA *symmetric* encryption scheme, padded with a tag of s zero bits. The key k is then used to encrypt the message using the symmetric encryption. The resulting OME scheme remains IND/IK-CCA and complete, but (1) All the public-key encryption costs can be done off-line, before the encryptor knows the plaintext, which in particular pushes all the encryption cost in the secret handshake scheme of Section 6 off-line; and (2) Heuristically, the oblivious decryptor rejects an attempt to decrypt a ciphertext using a wrong key with $1 - 2^{-s}$ probability. For all the ElGamal-based OME schemes we provide (using DHAES, Cramer-Shoup, or DDH-based ElGamal with the Fujisaki-Okamoto transformation), this means that the decryption cost in the above GKM scheme takes only $2 \log n$ exponentiations and $O(\Delta(\log n)^2)$ xor's.

Theorem 2. *If the underlying oblivious multi-encryption scheme is ϵ-complete and IND+IK-CCA secure then the above PKGKM scheme is ϵ'-complete and IND+IK-CCA, where $\epsilon' = 2\Delta \log n \epsilon$.*

6 Unlinkable Handshakes from Key-Private PKGKM

Let $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Revoke}, \mathsf{Enc}, \mathsf{Dec})$ be a Δ-complete IND/IK-CCA GKM scheme and $H : \{0,1\}^* \rightarrow \{0,1\}^k$ is a hash function modeled as random oracle in the security analysis. The SH scheme uses the same algorithm Setup, KGen, and Revoke, and the $\mathsf{Handshake}$ protocol is shown in Figure 1. Each player's inputs in the protocol is a triple $(SK, TPK, \mathsf{resp/init})$ where SK is that player's GKM key for his/her group, TPK is the public key of this player's *target* group, and $\mathsf{resp/init}$ is the player's role in the protocol. Let final be a special symbol, different from resp and init. Parameter $\hat{k}$ can be set as $\hat{k} = 2k$. In particular, it must be large enough so that the probability that two sessions choose the same nonce is negligible in k. Hash function H has a $\hat{k}$-bit range.

We refer to the full paper [JL07] for proofs of the following claims:

Theorem 3 (Δ-limited completeness). *Suppose the underlying Group Key Management scheme is Δ-Limited ϵ-complete. Then our Secret Handshake scheme is Δ-Limited ϵ'-complete, where $\epsilon' = 2\epsilon$.*

Theorem 4 (Security of the SH Scheme). *If the underlying Group Key Management scheme is (t, ϵ, n, q_D)-IND-CCA secure and Δ-limited ϵ_c-complete,*

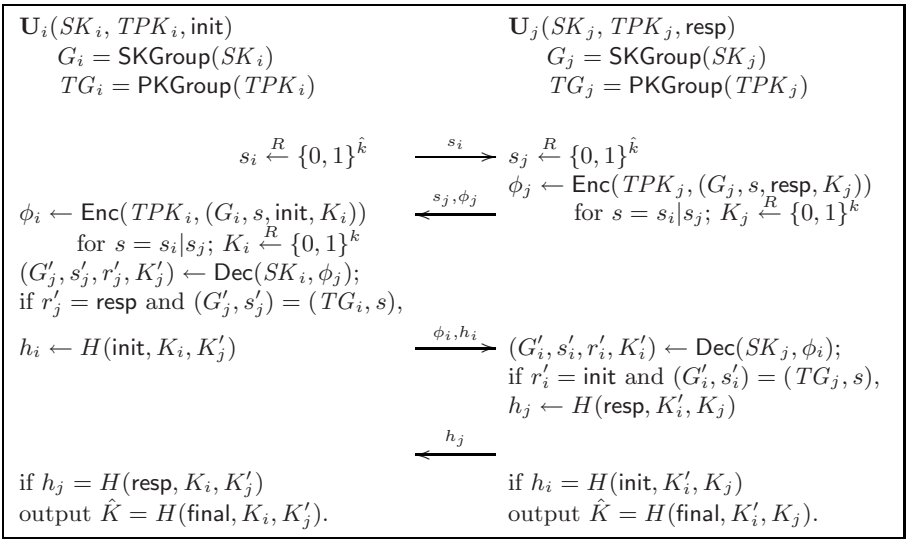

Fig. 1. Privacy-protecting AKE protocol $\mathsf{Handshake}(U_i, U_j)$

then the SH scheme constructed in Section 6 is $(t', \epsilon', n, n', R, q_S, q_H)$*-secure, for at most* q_S *commands,* q_H *hash queries,* n' *groups,* n *members per group, and maximum* R *sessions for each member, for* $R = q_D/n, \quad q_S = q_D, \quad t' = t - (R+2)nn'\Delta \log n \cdot t_{exp}$*, where* t_{exp} *is the cost of a single exponentiation, and* $\epsilon' = nn'R \cdot (\epsilon + (q_H + q_S) \cdot 2^{-\hat{k}} + \epsilon_c)$.

Theorem 5 (Anonymity and Affiliation/Policy Hiding Property of the SH Scheme). *If the underlying group key management scheme is* $(t_1, \epsilon_1, n, q_D)$*-IND-CCA and* $(t_2, \epsilon_2, n, q_D)$*-IK-CCA secure, then the SH scheme constructed in Section 6 is* $(t', \epsilon', n, n', R, q_S, q_H)$*-private (unlinkable and affiliation/policy hiding), for at most* q_S *commands,* q_H *hash queries,* n' *groups,* n *members per group, and maximum* R *sessions for each member, for* $R = q_D/n, \quad q_S = q_D, \quad t = \min\{t_1, t_2\} - (R+2)nn'\Delta \log n \cdot t_{exp}$*, where* t_{exp} *is the cost of a single exponentiation, and* $\epsilon = nn'R(\epsilon_1 + \epsilon_2 + (q_H + q_S) \cdot 2^{-\hat{k}} + \epsilon_c)$.

References

[AB07] G. Ateniese and M. Blanton. Secret handshakes with dynamic and fuzzy matching. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium, NDSS*, 2007.

[ABR01] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHAES. In *CT-RSA*, pages 143–158, 2001.

[BBDP01] M. Bellare, A. Boldyrevay, A. Desaiz, and D. Pointchevalx. Key-privacy in public-key encryption. In *Aisacrypt '01, Proceedings*, 2001.

[BBs03] M. Bellare, A. Boldyreva, and J. staddon. Randomness re-use in multi-recipient encryption schemes. In *Public-Key Cryptography '03, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[BBW06] A Barth, D Boneh, and B. Waters. Private encrypted content distribution using private broadcast encryption. In *Proceedings of Financial Crypto*, 2006.

[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.

[BHS04] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies in hidden credentials. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, 2004.

[CFN88] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proc. Crypto'88*, volume 403 of *LNCS*, pages 319–327. Springer-Verlag, 1988.

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

[CK02] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In *Advances in Cryptology - EUROCRYPT 2002*, 2002.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. Eurocrypt '01*, pages 93–118, 2001.

[DF02] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*, pages 61–80, 2002.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology-CRYPTO'99*, pages 537–554, August 1999.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[JKT07a] S. Jarecki, J. Kim, and G. Tsudik. Authenticated group key agreement protocols with the privacy property of affiliation-hiding. In *RSA Conference – Cryptography Track*, 2007.

[JKT07b] S. Jarecki, J. Kim, and G. Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange protocols with perfect forward privacy. manuscript, 2007.

[JL07] S. Jarecki and X. Liu. Unlinkable secret handshakes and key-privacy in group key management scheme. `http://eprint.iacr.org/2007/`, 2007.

[KP98] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptography - CRYPT0 1998*, Santa Barbara, CA, August 1998.

[TX05] Gene Tsudik and Shouhuai Xu. Brief announcement: a flexible framework for secret handshakes. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 39–39, New York, NY, USA, 2005. ACM Press.

[Ver05] D. Vergnaud. RSA-based secret handshakes. In *In International Workshop on Coding and Cryptography, Bergen, Norway*, 2005.

[WGL98] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *SIGCOMM '98*, 1998.

[WHA97] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. *IETF draft wallner-key*, 1997.

[XY04] Shouhuai Xu and Moti 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.