Authentication for Paranoids: Multi-party Secret Handshakes

Stanisław Jarecki, Jihye Kim, and Gene Tsudik

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

Abstract. In a society increasingly concerned with the steady assault on electronic privacy, the need for privacy-preserving techniques is both natural and justified. This need extends to traditional security tools such as authentication and key distribution protocols. A secret handshake protocol allow members of the same group to authenticate each other *secretly*, meaning that a non-member cannot determine, even by engaging someone in a protocol, whether that party is a member of the group. Whereas, parties who *are* members of the same group recognize each other as members, and can establish authenticated secret keys with each other. Thus, a secret handshake protocol offers *privacy-preserving authentication* and can be used whenever group members need to identify and securely communicate with each other without being observed or detected.

Most prior work in secret handshake protocols considered 2-party scenarios. In this paper we propose formal definitions of *multi-party secret handshakes*, and we develop a practical and provably secure multi-party secret handshake scheme by blending Schnorr-signature based 2-party secret handshake protocol of Castelluccia et al. [5] with a group key agreement protocol of Burmester and Desmedt [4].

The resulting scheme achieves very strong privacy properties, is as efficient as the (non-private) authenticated version of the Burmester-Desmedt protocol [4, 6], but requires a supply of one-time certificates for each group member.

Keywords: privacy-preserving authentication, secret handshakes, group key agreement, anonymity, privacy, authentication protocols.

1 Introduction

Consider the following scenario: two undercover Interpol agents, Alice and Bob, are in a crowded public place, such as an airport or a city square. They are not aware of each others presence or affiliation. However, each wants to discover, and communicate with other Interpol agents. Interpol rules prohibit agents from revealing their affiliation to non-agents. Since the environment is potentially hostile, Alice would thus authenticate to Bob only if he is an agent, and vice versa. No one who is not an Interpol agent as well should be able to determine whether

J. Zhou, M. Yung, and F. Bao (Eds.): ACNS 2006, LNCS 3989, pp. 325–339, 2006.

Alice (or Bob) is an agent, or even if Alice and Bob are members of any single organization. Likewise, if only one of the two (Alice or Bob) is a genuine agent, the other (the impostor) should learn nothing about the counterpart's affiliation. Furthermore, should anyone meet either Alice or Bob again, and engage them in an authentication protocol, they should not be able link the two encounters.

Traditional PKI-based authentication fails in the above scenario. Other intuitive approaches, such as key exchange followed by encrypted authentication, fail as well. Even more exotic cryptographic tools like group signatures and identity escrow are unsuitable since they protect anonymity of members within the same group, but are not designed to hide the *affiliation* of the group members.

1.1 Prior Work on Two-Party Secret Handshakes

To satisfy the security requirements for the secret agent example we need authentication schemes which are anonymous in the sense of hiding an affiliation of the participating parties. Such authentication schemes were named *secret handshakes* by Balfanz et al. in a paper [2] which introduced the notion of privacy (a.k.a. anonymity) for public-key two-party authentication schemes.[1] A (two-party) secret handshake (SH) scheme allows two group members (e.g. two entities certified by the same Certification Authority) to authenticate each other in an anonymous and unobservable manner in the sense that one party's membership is not revealed unless the other party's membership is also ensured. In other words, if party A who is a member of group G_1 engages in a (two-party) secret handshake protocol with party B who is a member of G_2, a secret handshake scheme guarantees the following [2]:

- A and B authenticate each other if and only if $G_1 = G_2$.
- If $G_1 \neq G_2$, both parties learn only the fact that $G_1 \neq G_2$.

A two-party secret handshake scheme can possess further desirable anonymity properties: (1) *Unobservability*: A non-group member cannot tell not only whether A or B belong to some given group but also whether A and B belong to any single group (and hence whether they accept or reject in the handshake protocol); (2) *Unlinkability*: Two occurrences of the same party cannot be linked with each other by anyone except the group manager; and (3) *Privacy against eavesdropping insiders*: Any *passive* observers, even including other group members, cannot learn anything from the protocol as well.

Balfanz, et al. [2] constructed the two-party SH scheme by adapting the key agreement protocol of Sakai, et al. [10]. Its security rests on the hardness of the Bilinear Diffie Hellman (BDH) problem. Subsequently, Castelluccia, et al. [5] developed a more efficient secret 2-party handshake scheme under more standard cryptographic assumption of Computational Diffie Hellman (CDH) problem. Both solutions are secure in the Random Oracle Model (ROM) for hash functions, and both attain properties (1) and (3) above, but attaining property (2), in both solutions, requires a supply of one-time certificates for each group member.

[1] Privacy for symmetric-key authentication schemes was considered before by Abadi [1].

1.2 Group Secret Handshakes: Prior Work and Our Contribution

Both aforementioned techniques are limited to 2-party settings. A natural next step is to explore the space of multi-party settings with similar security requirements. For example, we can re-consider our initial secret agent scenario but, this time, with four undercover Interpol agents. They are, as before, in certain proximity, and would like to discover each other and have a secure "conversation". However, each wants to authenticate to others if and only if all of them are similarly affiliated. The adversarial model is also similar. An adversary may eavesdrop or take part in the protocol in order to impersonate an agent or to detect others' affiliations. All properties of 2-party secret handshakes listed above can be adopted to *group* authentication and authenticated key agreement protocols. We will call authenticated group key agreement scheme which satisfies such privacy properties a *Group Secret Handshake* (GSH).

In a recent paper, Tsudik and Xu [12] presented the first group secret handshake (GSH) solution, which also supports reusable (sometimes called multi-show) certificates, instead of one-time certificates as in [2, 5]. However, their scheme ensures successful authentication between group members only if each member holds the same most recently distributed group key, which requires a lot of real-time communication between group manager and the group members.

In this paper we give a more formal definition of the GSH scheme than that given in [12], and we provide a solution which fits the standard PKI setting, and in particular avoids having the group manager broadcast key-update messages to the group members. Our solution is based on commonly taken assumptions (Computational Diffie Hellman and the Random Oracle Model for hash functions), achieves very strong anonymity properties, and is as efficient as existing (non-private) two-round group key agreement protocol based on the same assumptions, i.e. the Burmester-Desmedt protocol [4, 6]. On the negative side, our scheme requires a supply of one-time certificates for each group member, which implies more storage for group members, more computation for the group manager, and bigger sizes of the certificate revocation lists. However, such solution can still be practical for groups whose members do not engage in this authentication protocol all the time, e.g. no more than 100 times a month on the average.

1.3 Overview of Our GSH Construction

The idea of our scheme is to add affiliation-hiding authentication to the Burmester-Desmedt group key agreement protocol [4] via the signature-based affiliation-hiding encryption method which was given for the discrete-log setting by Castelluccia et al. [5].[2] In the signature-based encryption of [5], the certificate for member of a group G is a Schnorr signature (w, t), where $w = g^r$ and $t = r + x_G H(w, id)$ on a random ID strings id, under the public key $y_G = g^{x_G}$ of this group. The Schnorr signature can be thought of as a private key t and a public key $y = g^t$, which can be computed from the (w, id) pair as $y = w(y_G)^{H(e,id)}$.

[2] See section 1.4 below for a discussion of related works on signature-based encryption.

It was shown in [5] that, under the CDH assumption, if key y is computed from (w, id) as above then only the owner of a signature (t, w) on id under key y_G can decrypt ElGamal ciphertexts encrypted under y. Here we use the above "public key" $y = g^t$ not as an encryption key but as the contribution of a player to the Burmester-Desmedt group key agreement protocol, and we show that only the players who hold valid signatures (w, t) issued on some string id can contribute their values $y = g^t$ by sending (w, id) instead of y in the first round of the BD group key agreement protocol, and then recover the agreed-on key in the second round. We show that in this way only the certified group members can get the key that other certified members output, and moreover, that a non-certified player cannot tell what public key y_G the other players use, and hence that the scheme hides group membership of the authorized participants from the non-authorized.

The reason why this construction requires one-time certificates is that re-using a Schnorr certificate (w, t, id) in the protocol described above, corresponds to re-using the same contribution g^t in more than one instance of the Burmester-Desmedt key agreement protocol, which would yield that protocol insecure.

1.4 Other Related Work

In addition to the prior work described above, the work of Xu and Yung [13] constructed an interesting 2-party secret handshake scheme which achieves unlinkability with reusable credentials. However, this scheme requires each party to be aware of other groups (of which one is not a member) and offers weaker form of anonymity, referred to as *k-anonymity*.

The previously mentioned GSH scheme by Tsudik and Xu [12] achieves additional properties like self-distinction between players participating in the protocol, and traceability of the participating players by the group manager examining the transcript of the protocol. In our GSH protocol we achieve a weaker variant of the self-distinction property, called *counting* (see section 2.2).

Our GSH construction uses the signature-based encryption scheme based on the CDH problem given by [5]. Other signature-based encryption schemes, referred to as "oblivious signature-based envelopes" (OSBE), were developed for other cryptographic settings in [7] and [8]. We note that to satisfy the needs of private (group or two-party) authentication, the signature-based encryption scheme must have additional privacy property of affiliation-hiding[3] and it's an open problem to ensure this property for many OSBE schemes.

2 Definition of a Group Secret Handshake

In this section we describe the components of a GSH scheme and the security properties it should achieve.

A GSH scheme operates in an environment consisting of a set of players and a set of administrators who are responsible for creating groups, admitting chosen

[3] The same property was called "sender and receiver obliviousness" in [5].

players as group members, and possibly also revoking their membership. For simplicity's sake, we assume that each player is a member of exactly one group and that each group manager is responsible for a unique group, but our results can be easily generalized to the case when a player can be a member of many groups, and a manager can manage many groups as well. A GSH scheme is a tuple of algorithms (Setup, CreateGroup, AddPlayer, Handshake, RemovePlayer) described in figure 1.

Communication and Adversarial Model: We assume the existence of *anonymous* broadcast channels between all legitimates parties, where "anonymous" means that an outside attacker cannot determine identities of $\mathcal{GA}$, group members, as well as the dynamics and size of a group. Also, a malicious insider (group member) cannot determine identities of other honest group members as well as the dynamics and size of the group. This assumption is necessary in most privacy-preserving authentication schemes; otherwise, anonymity could be trivially compromised. However, we note that our requirement that SH protocols themselves must rely on anonymous channels does *not* necessarily present a problem. This is because a typical secret handshake application would be in a wireless LAN setting where *broadcast* – a natural source of anonymity – is a built-in feature. Additionally, we assume that participants' clocks are loosely synchronized. They specify when they start the protocol and how long they will wait for other player's messages in each protocol round. We stress that we do not assume any reliability properties of this broadcast medium, i.e. in our adversarial model the adversary can inject any messages into the protocol, delay, erase, and/or modify the messages sent between honest parties, and in particular deliver the broadcasted messages to arbitrarily selected players.

2.1 Basic Security Properties of GSH Scheme

A GSH scheme must be correct, authentic, and affiliation-hiding:

Correctness: For any group G managed by an honest GA, and any set Δ of honest players who are members of G, if the adversary forwards all messages between participants in a protocol GSH.Handshake(Δ), then all players in Δ output identical $(K, IDSet)$ pairs, where $IDSet$ has $|\Delta|$ elements, one per each player in Δ, uniquely identifying this player to the group manager GA.

Authenticity: The essence of this property is that if any honest player outputs a key in an instance of the GSH.Handshake scheme, then an attacker who can be an active participant in this protocol but who does not have a non-revoked certificate for that group, learns nothing about that key. Formally, we say that GSH.Handshake guarantees *authenticity*, if every polynomially-bounded adversary $\mathcal{A}$ has only negligible probability of winning of the following game:

1. GSH.Setup and GSH.CreateGroup algorithms are executed and resulting parameters params and public key $\mathcal{PK}_G$ are given to $\mathcal{A}$.
2. $\mathcal{A}$ triggers the GSH.AddMember algorithm under the public key $\mathcal{PK}_G$ polynomially many times. In each GSH.AddMember instance, $\mathcal{A}$ receives a mem-

GSH.Setup: This algorithm is executed publicly, on input of a sufficient security parameter k, to generate public parameters params common to all subsequently generated groups, e.g., k determines the size of the modulus used in cryptographic operations.

GSH.CreateGroup: This algorithm is executed by a group authority, GA to establish a group denoted G. It takes as input params, and outputs a the group public key $\mathcal{PK}_G$, the GA's private key $\mathcal{SK}_G$, and a certificate revocation list, $\mathcal{CRL}_G$, which is originally empty.

GSH.AddMember: This algorithm is executed between a player U and a GA who administers some group G. The input's are GA's private input $\mathcal{SK}_G$ and shared inputs params and $\mathcal{PK}_G$. The output is a membership cert for the player, which contains in particular a random bitstring id of fixed length, e.g. 160 bits. We say that a player who receives a cert in this protocol is a *member* of group G. We assume that GA admits members according to some admission policies, but specification and enforcement of such policies are outside the scope of this paper. The AddPlayer protocol can be executed between same GA and U many times, in which case U receives a set of certs as a result, each containing a different id string (except for negligible probability).

GSH.Handshake(Δ): This algorithm is executed by a set Δ of n players purporting to be members of a group G, where $\Delta = \{U_1, ..., U_n\}$ and $n \geq 2$. Each player U_i runs the protocol on inputs a public key $\mathcal{PK}_G$, a set of certs received from G's GA, and (U_i's current view of) $\mathcal{CRL}_G$. At the end of the protocol, each player outputs either $(K, IDSet)$, in which case we say that the player accepts, where K is an authenticated key for use in subsequent secure communication, and $IDSet$ is a set of id's, or REJECT, in which case we say that the player rejects.

GSH.RemoveMember:: This algorithm is executed by GA. On input of some player identity U, GA looks up the id's assigned to U in instances of the AddMember between this GA and U, and inserts them into $\mathcal{CRL}_G$. The updated $\mathcal{CRL}_G$ is assumed to be publicly available.

Fig. 1. GSH Scheme Components

bership cert from the $\mathcal{GA}$. Before the protocol starts, all certs $\mathcal{A}$ received are added to $\mathcal{CRL}_G$, which is sent to all honest players in G.

3. $\mathcal{A}$ chooses a set of player $\Delta = (V_1, ..., V_l)$ in G, triggers the execution of GSH.Handshake(Δ), and participates in this execution, i.e. hears all the messages, controls their delivery, and can any messages it wants to the participants.
4. If any honest player in Δ accepts, and outputs $(K, IDSet)$ pair, $\mathcal{A}$ wins if he has non-negligible advantage in distinguishing between the following two games: In game [A], $\mathcal{A}$ is given a key K output by some (randomly chosen) accepting player in Δ. In game [B], $\mathcal{A}$ is given a random bitstring of the same length.

Note: The above definition of is a simplified form of the security requirement of an authenticated group key agreement scheme (AGKA). In particular, it does not model security under concurrent execution of multiple instances of the GSH protocol. However, the emphasis of our contribution is on the *anonymity* properties of a group key agreement, so we examine the *security* of the protocol we propose only under the restricted notion above. The full analysis of the security of the group key agreement protocol involves modeling it as an ideal functionality, as in the Katz-Yung [6], and is out of the scope of this current paper.

Affiliation-hiding:[4] A GSH scheme is affiliation-hiding if all messages from an honest player in the entire protocol do not leak the identity of the GA which certified that player, even if this player is engaged in a group handshake protocol involving malicious participants. Formally, we call a GSH scheme *affiliation hiding* if there exists a probabilistic polynomial-time algorithm SIM, such that no polynomially-bounded adversary $\mathcal{A}$ has a non-negligible advantage in distinguishing between the following two games:

1-2. Steps 1-2 are the same as in the *authenticity* property.
3. $\mathcal{A}$ picks any set of players $\Delta = (V_1, ..., V_l)$, not necessarily belonging to one group, and then:
3.1 In game 1, $\mathcal{A}$ interacts with players in Δ executing protocol $\mathsf{GSH}.\mathsf{Handshake}(\Delta)$.
3.2 In game 2, $\mathcal{A}$ interacts with SIM which runs only on input $l = |\Delta|$ and params.

Note: This definition implies that an adversary $\mathcal{A}$ cannot tell not only if the other participating players are members of some group G (for which $\mathcal{A}$ does not have non-revoked certs), but also if the other players belong to any single group at all. Thus the above definition implies the property of GSH scheme which can be called **unobservability**. This definition also implies the **unlinkability** property, which says that even an active adversary cannot link two instances of the handshake protocol in which the same player participates. These strong anonymity properties are implied by the above definition because the simulator's only input is the *size* of the set Δ, and not the identities of the individual players, nor their group membership(s). We remark that our GSH protocol achieves the unlinkability property in a rather trivial way by using one-time certificates which are discarded after a single use.

2.2 Other Security Properties of a GSH Scheme

We also specify two less central but potentially useful security properties for GSH schemes, *counting* and *affiliation-hiding against eavesdropping insiders*:

Counting: The counting property says that the set of *id*'s, $IDSet$, output by an honest player that accepts in a handshake protocol, has some correspondence to the number of players who are group members among the participants. Namely, as long as no malicious group member participates in the protocol, the size of the $IDSet$ is no larger than the set of group members participating in this protocol. (We cannot require $|IDSet|$ is *equal* to the number of participating group members, because the adversary controls the communication network, and hence can always not deliver some players' messages.) Formally, we say that a GSH scheme accomplishes the *counting* property if every polynomially bounded adversary $\mathcal{A}$ has negligible probability of winning in the following game:

[4] The *affiliation-hiding* property we define here implies what was called *detection-resistance* in previous secret-handshake papers [2, 5, 12].

1. GSH.Setup and GSH.CreateGroup algorithms are executed and resulting parameters params and public key $\mathcal{PK}_G$ are given to $\mathcal{A}$.
2. $\mathcal{A}$ triggers the GSH.AddMember algorithm under the public key $\mathcal{PK}_G$ polynomially many times. In each GSH.AddMember instance, $\mathcal{A}$ receives a membership cert from the GA for this group G. Before the protocol starts, all secrets $\mathcal{A}$ received are added to $\mathcal{CRL}_G$, which is sent to all players in G.
3. $\mathcal{A}$ runs GSH.Handshake with any group Δ of honest members in G.
4. $\mathcal{A}$ wins if any honest player in Δ outputs $(K, IDSet)$, where $IDSet$ includes more id's than the size of set Δ.

Affiliation-hiding against eavesdropping insiders: Note that the affiliation-hiding property implies security against both passive (i.e. only eavesdropping) and active *outsiders*, i.e. adversaries that have no current non-revoked certificates for an attacked group. However, a GSH scheme could also offer affiliation-hiding protection (which, as we pointed out above, implies unobservability and unlinkability) against an adversary who does have non-revoked certificates (i.e. an adversary who is a valid member of the attacked group) but who is only eavesdropping on the handshake protocol. (Note furthermore that this is the best we can ask for, because if such adversary is active, he can learn everything by just participating in the handshake protocol using his non-revoked cert.) We do not formally define this property, since it is very similar to the security against active attackers which we already defined for the properties of authenticity and affiliation-hiding.

3 Construction of a Group Secret Handshake Scheme

We now construct a practical GSH scheme achieving authenticity and affiliation-hiding under the CDH assumption in ROM. As mentioned in section 1.3, it is based on the Burmester-Desmedt (unauthenticated) group key agreement scheme [4] (see figure 4 in the appendix).

We point out from the outset that we modify the Burmester-Desmedt protocol in the process, by adding an extra layer of hashing into the key derivation (see the form of our session key shown in Lemma 1). The reason is that our authentication method is highly non-standard; hence, the security argument for the resulting authenticated group key agreement (AGKA) scheme becomes easier once the components of the session key related to each player are put through a hash function modeled as a random oracle. Our GSH scheme is shown in figure 2.

Lemma 1. *Protocol* AGKA *in figure 2 is a* correct *group key agreement scheme. That is, if all parties adhere to the protocol then each will compute the same key:* $K = F(g^{t_1 t_2})F(g^{t_2 t_3}) \cdots F(g^{t_n t_1}) \pmod p$

Proof. Let
$B_{i-1} \equiv F(z_{i-1}^{t_i}) \equiv F(g^{t_{i-1} t_i}) \pmod p$,
$B_i \equiv F(z_{i-1}^{t_i}) \cdot X_i \equiv F(g^{t_i t_{i+1}}) \pmod p$,
$B_{i+1} \equiv F(z_{i-1}^{t_i}) \cdot X_i \cdot X_{i+1} \equiv F(g^{t_{i+1} t_{i+2}}) \pmod p$,
$\cdots$

Setup: This algorithm outputs the standard discrete logarithm parameters (p, q, g) of security k, i.e., primes p, q of size polynomial in k, s.t. g is a generator of a subgroup in Z_p^* of order q. $\mathcal{GA}$ also defines hash functions $H : \{0,1\}^* \rightarrow Z_q$, $F : \{0,1\}^* \rightarrow Z_p$. The hash functions are modeled as random oracles.

CreateGroup: $\mathcal{GA}$ sets the group secret $\mathcal{SK}_G$ to be a random number $x \in Z_q$ and the group public key $\mathcal{PK}_G$ to be $y = g^x \pmod p$.

AddMember: To add a player U to the group G, $\mathcal{GA}$ does the following: First, it generates a list of random "pseudonyms" $id_1, ..., id_f \in \{0,1\}^{160}$, where f is chosen to be larger than the number of handshakes U will execute before receiving new player secrets. Then, $\mathcal{GA}$ computes a corresponding list of Schnorr signatures $(w_1, t_1),...,(w_f, t_f) \in (Z_p^*, Z_q)$ on all ids picked above under the key y as [11], i.e., a pair (w_k, t_k) where $w_k = g^{r_k} \pmod p$, and $t_k = r_k + xH(w_k, id_k) \pmod q$, for random $r_k \leftarrow Z_q$. A signature pair (w_k, t_k) on id_k satisfies that $g^{t_k} = w_k y^{H(w_k, id_k)} \pmod p$. The player's outputs are the list of *certs* $((t_1, id_1, w_1), ..., (t_f, id_f, w_f))$. Sometimes we will refer to a t_i value as a "trapdoor" for the (id_i, w_i) pair.

RemoveMember: To remove a player U from the group G, $\mathcal{GA}$ looks up pseudonyms $(id_1, ..., id_f)$ it has issued to U, adds the pseudonyms to the current $\mathcal{CRL}$ and outputs an updated $\mathcal{CRL}$.

AGKA(Δ): This is a group key agreement algorithm for some set $\Delta = \{U_1, ..., U_n\}$ of the honest players, where each player $U_i \in \Delta$ receives a signal to start the protocol. Each player U_i *removes* a single cert (t_i, id_i, w_i) from its list of certs. (Note that this cert will be removed from the list whether the subsequent protocol succeeds or not.) The protocol consists of two rounds:

[**Round 1**]: Each player U_i broadcasts (id_i, w_i).

- If there are collisions between id's, U_i just abandons the protocol. If U_i receives any id's on $\mathcal{CRL}$, he broadcasts a random value as X_i in Round 2 and outputs REJECT.
- If there are neither id collisions nor revoked id's, U_i determines the order between players based on id's. We assume that the order of players is determined by their pseudonyms, e.g., increasing order of hash images of pseudonyms. For simplicity of description, wlog, we assume that the ordered result is $(U_1, U_2, ..., U_n)$ and the indices are taken in a cycle modulo n, i.e. $U_{n+1} = U_1$.
  - U_i computes $z_{i+1} = w_{i+1} y^{H(w_{i+1}, id_{i+1})} (= g^{t_{i+1}})$ and $z_{i-1} = w_{i-1} y^{H(w_{i-1}, id_{i-1})} (= g^{t_{i-1}})$.
  - U_i computes $X_i = F(z_{i+1}^{t_i}) / F(z_{i-1}^{t_i}) \pmod p$

[**Round 2**]: Each player U_i broadcasts X_i.

- U_i computes $K_i = F(z_{i-1}^{t_i})^n \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \pmod p$.
- U_i outputs $(K_i, IDSet_i)$, where $IDSet_i = \{id_1, ..., id_n\}$.

Fig. 2. GSH: A Group Secret Handshake Scheme

$B_{i-2} \equiv F(z_{i-1}^{t_i}) \cdot X_i \cdot X_{i+1} \cdot X_{i+2} \cdots X_{i-2} \equiv F(g^{t_{i-2} t_{i-1}}) \pmod p$.
Then $K_i \equiv B_{i-1} B_i B_{i+1} \cdots B_{i-2} \equiv F(z_{i-1}^{t_i})^n \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \pmod p$.
□

Note on Performance: We compare the performance of the GSH scheme in figure 2 with the original (non-authenticated) Burmester-Desmedt scheme, shown in figure 4 of the appendix. Communication cost is the same since both schemes require two communication rounds. When we consider the on-the-fly computation, BD requires two modular exponentiations and GSH involves two modular multi-exponentiations, respectively. Thus, the efficiency of the GSH scheme is comparable to the original BD scheme. Therefore, the GSH scheme not only

provides an authentication to the BD protocol almost for free, but also provides an authentication with very strong privacy property of affiliation-hiding. On the other hand, we note that our GSH scheme requires use of one-time certificates, which compared to standard PKI authentication creates additional storage requirements for the group members, increases the computation cost for the group manager, who needs to create a list of certificates for each group member, and increases the size of the CRL list.

Theorem 2. *The GSH scheme in figure 2 is* **affiliation-hiding** *under the CDH assumption in the Random Oracle model.*

Proof: The simulator required to prove the affiliation-hiding property is very simple: It sends random values on behalf of all the honest players $(V_1, ..., V_l)$ participating in the protocol: It picks random id_i's, w_i's chosen at random in the subgroup generated by g, and random values X_i's in Z_p^*. It is easy to see that neither id_i nor w_i values sent by the honest players in the first round of the protocol reveal any information about the $\mathcal{GA}$ in the first round: Since each w is created as $w = g^r$ for random r, it is independent from $\mathcal{GA}$'s public key y, and id's are randomly chosen as well.

The only values which can reveal something about the group membership of the honest players are the X_i values sent in the second round. However, the only way an adversary can distinguish between a conversation with honest players and a conversation with the above simulator sending random X_i's is if the adversary queries the random oracle F on one of the two inputs, $z_{i+1}^{t_i}$ or $z_{i-1}^{t_i}$, used to compute the X_i value used by any honest player V_i. We will argue that if such adversary exists then this adversarial algorithm can be used to break the Computational Diffie-Hellman assumption, i.e. on input a random pair (y, c) in the subgroup generated by g in Z_p^*, the simulator will output c^x s.t. $y = g^x$ with a non-negligible probability. First, in the initialization procedure the adversary is given the y part of this CDH challenge as the public key of the $\mathcal{GA}$ of the group it is attacking. The simulator then uses the c value in its simulation, and extracts the c^x from one of the queries the adversary makes to the F oracle, as follows.

Without loss of generality, we can assume the adversary queries F on one of the $z_{i+1}^{t_i}$ values, since the argument is the same in the other case. Also, if the adversary has a non-negligible probability of querying F on any such value, then there exists an index $i \in \{1, ..., l\}$ s.t. the adversary has a non-negligible probability of querying F on a value with this particular index i. Moreover, since the adversary makes polynomial number of queries to F, there is an index j of his queries to F and a non-negligible probability ϵ s.t. value $z_{i+1}^{t_i}$ appears as j-th query to F.

For that index i, the simulator in round one sends (w_i, id_i) pair chosen in a special way. Namely, it picks random id_i as before, but it picks also a random value e_i in the range of F, computes $w_i = c * y^{-e_i}$ and sets $H(w_i, id_i)$ to e_i. In this way, we will have $z_i = w_i * y^{H(w_i, id_i)} = w_i * y^{e_i} = c$. (The distribution created by the simulator in this way is correct because c is random in the group generated by g.) Now, note that $z_{i+1}^{t_i} = z_i^{t_{i+1}}$, and since $z_i = c$, it follows that one of the queries the adversary makes to F is equal to $c^{t_{i+1}}$. Now, without loss

of generality we can assume that index $i+1$ corresponds to a corrupt player A_{i+1}, and therefore the value t_{i+1} is defined as a value s.t. $g^{t_{i+1}} = w_{i+1}y^{e_{i+1}}$ where $e_{i+1} = H(w_{i+1}, id_{i+i})$. If we can rewind the adversary and witness two of its executions which run on the same random inputs until the adversary queries H on pair (w_{i+1}, id_{i+1}), but feed the adversary different challenges, $e_{i+1}^{(1)}$ and $e_{i+1}^{(2)}$, as F's responses in these two executions, then by the forking lemma of Pointcheval-Stern [9], it follows that with probability $O(q_H/\epsilon)$, where q_H is the number of queries the adversary makes to H, we see two executions, for $r = 1$ and $r = 2$, s.t. the adversary's j-th query to oracle F is equal to value $\alpha^{(r)} = c^{t_{i+1}^{(r)}}$, where $g^{t_{i+1}^{(r)}} = w_{i+1}y^{e_{i+1}^{(r)}}$. Since it follows from the last constraint that $t_{i+1}^{(r)} = k_{i+1} + x * e_{i+1}^{(r)}$ where $g^{k_{i+1}} = w_{i+1}$, the simulator can extract c^x from these two values $\alpha^{(1)}$ and $\alpha^{(2)}$, by outputting $(\alpha^{(1)}/\alpha^{(2)})^{1/\delta e}$ where $\delta e = e_{i+1}^{(1)} - e_{i+1}^{(2)}$. □

Theorem 3. *The GSH scheme in figure 2 is* **authentic** *under the CDH assumption in the Random Oracle Model.*

Proof: The proof is almost identical to the one above. The only way the adversary can distinguish key K_i output by any honest player V_i is if the adversary queries oracle F at point $z_{i-1}^{t_i}$. The proof above shows that the adversary who can compute *either* $z_{i+1}^{t_i}$ or $z_{i-1}^{t_i}$ for any index i of an honest player, can be reduced to breaking CDH. Therefore the authenticity of our AGKA holds under the same assumption.

4 Group Secret Handshake Scheme with Counting

In this section we add explicit mutual authentication to the GSH scheme from the previous section, which allows us to support the counting property.

Bresson et al. [3] show how to accomplish explicit authentication for any group key agreement protocol with minimal extra computation. We adopt their method, which consists of MAC-ing the transcript using the agreed-upon key, and we show that this simple mechanism enables the counting property, and that the resulting protocol still maintains the properties of authenticity or affiliation-hiding. Note that the extra cost due to generation and verification of hash-based MACs is negligible.

Given a hash function $H_3 : \{0,1\}^* \rightarrow \{0,1\}^k$ modeled as a random oracle, we modify the Handshake protocol in our GSH scheme, as shown in figure 3. We denote the GSH scheme resulting from this modification of the Handshake protocol GSH+MAC.

Theorem 4. *The* GSH+MAC *construction in figure 3 is an authentic and affiliation hiding GSH scheme, which additionally provides the* counting *property.*

Proof of Authenticity (sketch). The authenticity property is very clear since the GSH scheme provides authenticity by theorem 3 and the message in Round 3 does not reveal any information of the agreed key the Random Oracle Model.

GSH+MAC.Handshake(Δ): The protocol proceeds as the GSH.Handshake(Δ) protocol (see figure 2 on page 9), with the following modification:

[**Run** GSH.Handshake(Δ)]

- If U_i computes $(K_i, IDSet_i)$ in round 2 of the GSH.Handshake protocol, it does not output it, but computes $M_i = H_3(K_i, id_i)$. If U_i was to reject in the GSH.Handshake protocol, it picks M_i as a random bitstring of appropriate length.

[**Round 3**]: Each player U_i broadcasts M_i.

- U_i computes $M'_j = H(K_i, id_j)$ and checks if $M_j = M'_j$ for $1 \leq j \leq n$. If U_i verifies all M_j's, then U_i outputs $(K_i, IDSet_i = \{id_1, ..., id_n\})$, in which case we say that U_i accepts. Otherwise it rejects and outputs REJECT.

Fig. 3. GSH+MAC: A GSH Scheme with MAC-based Authentication

Proof of Affiliation-hiding (sketch). We will show a simulator SIM s.t. if $\mathcal{A}$ distinguishes between interactions with SIM and interactions with a group member, we can break the authenticity property. Since the underlying AGKA achieves affiliation-hiding property there exist simulators $SIM_{(AGKA)}$ which satisfy the affiliation-hiding criteria. We define a simulator SIM, running on inputs (params), as follows: (1) To simulate U_i's messages in AGKA, we use $SIM_{(AGKA)}$. (2) To simulate U_i's message in the third round, SIM sends random $M_i \leftarrow \{0,1\}^k$. If $\mathcal{A}$ can distinguish a conversation with such SIM from a conversation with a true group member U_i, since the $SIM_{(AGKA)}$ simulator produces messages which are indistinguishable from the message of an honest U_i, it must be that $\mathcal{A}$ distinguishes random values M_i chosen by SIM from values $M_i = H(K_i, id_i)$, In ROM, it can happen only if $\mathcal{A}$ makes an oracle query on the input (K_i, id_i). In this case, since $\mathcal{A}$ can make only polynomially-many queries to H, we pick one such query at random. And we will have a non-negligible chance of outputting K_i. This contradicts to authenticity property in AGKA. Therefore $\mathcal{A}$ can distinguish a conversation with SIM from a conversation with a group group member with only negligible probability.

Proof of Counting (sketch). The counting property follows immediately from the authenticity property: Since by the latter property, the adversary cannot distinguish a key K_i, for any player U_i in Δ, from a random string. Therefore the adversary also cannot forge a proper MAC M_i on any string, and hence the size of the set $IDSet_i$ output by any honest accepting player U_i in Δ, is at most equal to the size of set Δ.

5 Privacy Issues Involved in Revocation

Every $\mathcal{GA}$ that issues certificates will also need to revoke them. There can be many reasons for this. One reason is that the private keys corresponding to the certificate have been lost or compromised. Then the certificate holder contacts the $\mathcal{GA}$ and asks that the certificate be revoked. A $\mathcal{GA}$ may also decide to

revoke a certificate. For example, the certificate holder may violate the issuing agreement, or there can be promotions or retirement. Whatever the reason, the revoked group member's pseudonyms appears on the CRL of the issuing $\mathcal{GA}$, and anyone who receives the CRL knows which pseudonyms are revoked from the particular $\mathcal{GA}$. Since the CRL is generally public, we should examine whether there is any loss of privacy in the context of secret handshakes. Especially, we recognized that forward secrecy can be subverted if we depend on the normal revocation method, the CRL.

The CRL destroys the forward secrecy property against affiliation hiding and unlinkability. When non-group members receives the CRL, they may detect some group members by comparing pseudonyms on the CRL and pseudonyms they have seen in other protocol executions. In the case that the same group members get the CRL, they may link the same party from the previous protocol runs by looking at the difference in the update CRL. This is because all pseudonyms assigned to one group member are treated atomically in the revocation process.

One solution to mitigate the CRL problem is to issue time-based certificates, which are used only at a specified time and automatically expires after the time. When a group member needs to be revoked, the $\mathcal{GA}$ places only un-expired pseudonyms to the CRL. Since the used pseudonyms expire implicitly, this method is free from leaking any information regarding to the earlier protocol runs. The main disadvantage of this approach is that each group member needs to have lots of pseudonyms more than they use. For example, if a player participates a protocol at least once a week and each certificate expires every day, the player will be given seven certificates only for the one protocol execution. If the certificate expires every minute, the problem will be even worse. This approach may be practical in a very limited setting where players know when and how many times they will execute the protocol.

Another solution is to distribute the CRL only to the non-revoked group members. This can be done, for example, by keeping a group key among the current group members and publish the encrypted CRL using the group key. In this case, the issue will be how to update the group key efficiently. We may need a cryptographic tool such as broadcast encryption. However, security properties should be considered again, while we integrate other cryptographic tools. For example, we should check if updating messages in broadcast encryption reveal affiliation information of the group.

Instead of using the CRL, the $\mathcal{GA}$ can invalidate all the issued certificates by changing its public key. Whenever the public key is updated, non-revoked members synchronize their new pseudonyms lists with their $\mathcal{GA}$. This approach easily solves the revocation problem without revealing any further information. However, each player's burden will not be negligible if the revocation happens frequently.

We briefly mentioned three possible approaches for the private-preserving revocation technique. It is our future work to efficiently implement the proposed methods.

References

1. M. Abadi. Private authentication. In *Workshop on Privacy-Enhancing Technologies (PET)*, 2002.
2. D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *24th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.
3. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *ACM CCS*, 2001.
4. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. D. Santis, editor, *Proc. EUROCRYPT 94*, pages 275–286. Springer, 1994. Lecture Notes in Computer Science No. 950.
5. C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *Advances in Cryptology - ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 293–307. Springer, 2004.
6. J. Katz and M. Yung, Scalable Protocols for Authenticated Group Key Exchange. In Proceedings of CRYPTO 2003, LNCS 2729, pp. 110–125. Springer-Verlag, 2002.
7. N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. *In Proceedings of 22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, Boston, Massachusetts, July 13-16 2003.
8. S. Nasserian and G. Tsudik, Revisiting Oblivious Signature-Based Envelopes. In Proceedings of Financial Cryptography 2006 (FC'06), February 2006.
9. D. Pointcheval and J. Stern. Security proofs for signatures. *Advances in Cryptology - EUROCRYPT 1996*, pages 387–398, Springer, 1996.
10. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the Symposium on Cryptography and Information Security (SCIS)*, 2002.
11. C. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology - CRYPTO 1989*, Santa Barbara, CA, August 1989.
12. G. Tsudik and S. Xu. A Flexible Framework for Secret Handshakes. In *ACM Conference on Principles of Distributed Computing (PODC'05)*, August 2005.
13. S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pages 158–167. ACM Press, 2004.

Appendix A: Burmester-Desmedt Group Key Agreement

Figure 4 shows the Burmester-Desmedt group key agreement protocol. Note that this protocol is not an *authenticated* group key agreement.

$\mathsf{GKA}(\Delta)$: This is a group key agreement algorithm for $\Delta = \{U_1, ..., U_n\}$, where U_i's are members of a group G that want to generate a group key. g is a generator in Z_p^*.

[**Round 1**]: Each player U_i picks a random $t_i \in Z_q$ and broadcasts $z_i = g^{t_i}$.
U_i computes $X_i = (z_{i+1}/z_{i-1})^{t_i} \pmod p$, where the indices are taken in a cycle.

[**Round 2**]: Each player U_i broadcasts X_i
U_i computes the key: $K_i = (z_{i-1})^{nt_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdots X_{i-2} \pmod p$

(It may be easily verified that all players compute that same key $K = g^{t_1t_2+t_2t_3+\ldots+t_nt_1}$.)

Fig. 4. Burmester-Desmedt's Group Key Agreement Protocol