Secret Handshakes from Pairing-Based Key Agreements

Dirk Balfanz[1], Glenn Durfee[1], Narendar Shankar*[2],
Diana Smetters[1], Jessica Staddon[1], Hao-Chi Wong[1]

[1] Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
{balfanz, gdurfee, smetters, staddon, hcwong}@parc.com

[2] University of Maryland
A. V. Williams Building
College Park, MD 20742
narendar@cs.umd.edu

Abstract

Consider a CIA agent who wants to authenticate herself to a server, but does not want to reveal her CIA credentials unless the server is a genuine CIA outlet. Consider also that the CIA server does not want to reveal its *CIA credentials to anyone but CIA agents – not even to other CIA servers.*

In this paper we first show how pairing-based cryptography can be used to implement such secret handshakes. *We then propose a formal definition for secure secret handshakes, and prove that our pairing-based schemes are secure under the Bilinear Diffie-Hellman assumption. Our protocols support role-based group membership authentication, traceability, indistinguishability to eavesdroppers, unbounded collusion resistance, and forward repudiability.*

Our secret-handshake scheme can be implemented as a TLS cipher suite. We report on the performance of our preliminary Java implementation.

1. Introduction

The folklore of exclusive societies or groups includes the notion of a *secret handshake* whose purpose is to allow members of the group to identify each other. Secret handshakes guarantee the following: 1) non-members cannot recognize the handshake, and therefore are not able to recognize group members; and 2) non-members can't perform the handshake and therefore are unable to fool group members into thinking they are also members.

In this paper, we propose a scheme that can be used by members of a group to authenticate each other with the same guarantees as a secret handshake. Moreover, group members can play different roles within a group, and can authenticate themselves in these roles. For example, if the group is a secret society with different membership levels (novice, grand master, *etc*), then not only is group membership authenticated, but the membership level (role) of the other party is as well. Thus, if party A is a member of group G_1 and has the role r_A, and B is a member of group G_2 with role r_B, our scheme is such that, after a handshake between A and B,

- Neither A nor B learns anything about the other party if G_1 does not equal G_2;
- Both A and B learn their respective group memberships only if G_1 equals G_2 (*i.e.,* if they are, in fact, members of the same group);
- A can choose to only authenticate to members with certain roles. For example, A can decide not to reveal anything about itself unless B is a member of the same group as A, and has role r_B. The same is true for B.
- A third party observing the exchange between A and B does not learn anything (including whether A and B belong to the same group, the specific identities of the groups, or the roles of either A or B).

Our scheme can also provide traceability (if an adversary breaches the scheme by corrupting a true member, or a set of members, then that member will be traceable), forward repudiability (a successful handshake interaction between two members U_1 and U_2 does not give either of them the ability to prove the membership of the other to a third party), and collusion resistance (the system remains secure even if collections of users pool their secrets in an attempt to undermine the system). However, unlike a physical handshake, our scheme is asymmetric, and fairness cannot be guaranteed.

Our scheme is a simple adaptation of the non-interactive key agreement scheme of Sakai, Ohgishi and Kasahara [27],

*This work was done while visiting the Palo Alto Research Center.

Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)
1081-6011/03 $17.00 © 2003 IEEE

and its security rests on the hardness of the bilinear Diffie-Hellman problem (see, for example, [6]). In addition to presenting the scheme, we offer a formal notion of secure secret handshakes, and prove that our construction satisfies the formal definition of security. Both the definition of secure secret handshakes and the analysis that our scheme satisfies it are novel, and constitute the core of our contributions.

Secret handshakes can be used to securely discover services that are restricted to authorized users. For example, if an "air marshal service" is deployed at an airport, secret handshakes can be used to ensure that only air marshals can discover and use that service. Likewise, other service providers (like weather forecasters or commercial service providers) can be prevented from learning of the presence of air marshals through the use of secret handshakes.

Secret handshakes can also be used for privacy-preserving authentication. Unlike other solutions (*e.g.,* [8]), the use of secret handshakes does not require users to blind, or withhold, part of their credentials in order to achieve privacy. Instead, users can present all their credentials, and rest assured that the receiving party will not learn anything about credentials that were issued by a different group.

The ability of our scheme to handle different roles within a group has practical applications. For example, while a pro-democracy movement may have a flat organization (every member is a peer), and members authenticate each other simply as "member-of-the-movement", in some groups members play different roles. In a group that implements the traffic-regulating arm of the government, two roles are clearly needed: "traffic cop" and "vehicle operator". A vehicle operator should be able to authenticate to a legitimate traffic cop if and only if the operator is certified to operate a vehicle. If, however, an impostor is posing as a traffic cop then he will be unable to verify the driver's license of the other party, (even if the impostor is a certified vehicle operator).

This paper is organized as follows. We start off discussing, and contrasting secret handshakes with, related work. We then run through an extensive example that explains how our scheme works. In Section 4 we give a more detailed treatment of our scheme – we introduce the notion of a secret handshake, explain what it means for a secret handshake to be secure, and show that indeed our scheme implements a secure secret handshake. Our implementation experience is discussed in Section 6. We conclude with a discussion of practical issues in Section 7.

2. Related Work

Secret handshakes require a mechanism for group-based authentication. That is, users must only be able to authenticate themselves as members of a group to other members of the group. In addition, handshakes can be performed in a privacy-preserving manner, meaning that as a result of the handshake each user only learns that the other party is a member of the group, not the party's identity. The group membership detection problem has been studied in a variety of settings that overlap partially, but not completely, with ours. In the following we describe a number of these works and explain why they don't solve the secret-handshake problem.

KEY AGREEMENT/EXCHANGE. We use the novel key agreement protocol of Sakai, Ohgishi and Kasahara [27] to build a new tool: the secret handshake. This key agreement protocol is used in [24] to accomplish authenticated identity-based encryption in a simpler setting than the one we consider (*i.e.,* there is no notion of roles or groups). In our scheme, completing the secret handshake is essentially equivalent to computing a key that is particular to the two interacting group members. Hence, the secret handshake changes according to the group members involved. This gives us the opportunity to ensure collusion resistance: a coalition of corrupted groups members should not be able to perform the handshakes of group members outside the coalition. Requiring collusion resistance means that some care must be used if a key agreement protocol forms the foundation of a handshaking scheme. For example, such collusion resistance is not even an option with a shared group key. Further, a scheme based on shared group keys has the additional disadvantages of untraceable key leaks, rekeying upon every member revocation and no role capability. The Two-party Diffie-Hellman [17] key agreement scheme provides pair-specific handshakes but no collusion resistance (for details on this point see Section 5.3). In fact, the goals of key agreement protocols do not include group-based authentication, so as a class such protocols don't fit our needs. We use the Sakai *et al.* scheme because it is secure against colluding sets of users of unbounded size provided the discrete log problem is hard.

The standardized key exchange scheme IKE [20] in its identity protection mode uses unauthenticated Diffie-Hellman key agreements to hide the identities of the participants from eavesdroppers. It does not, however, provide identity protection from active attackers. Similarly, [1] provides protection against eavesdroppers but doesn't preserve the privacy of the communicating parties.

ANONYMOUS CREDENTIALS. When group membership is proven via credentials obtained from a central authority, anonymity can still be achieved. For example, if a restrictive blind signature issuing protocol [8] or self-blinding certificates [29] are used, the amount of identifying information that is revealed can be limited by blinding parts of the certificate. In addition, Chaum's [13] pseudonym systems (see also [25, 10]) allow users to prove membership via cer-

tificates issued under unlinkable pseudonyms. In either of these approaches there is an untraceable key that can be used by *anyone* to verify membership. Although public verifiability is a useful attribute that's quite difficult to achieve, it is not appropriate for secret handshakes. Even though our system makes use of pseudonyms, verification is only possible by group members because it relies on unique, secret information (*i.e.,* the secret handshake).

SIGNATURES. There are a variety of techniques that allow users to generate signatures with anonymity. For example, signatures generated with a group signature scheme [14, 11, 18, 9] only allow the verifier to determine the signature was generated by someone in the group. Identity escrow schemes [23] are essentially equivalent. Ring signatures [26] provide group signatures for ad hoc groups. Both techniques are inappropriate for secret handshakes for the same reason as anonymous credentials, namely, anyone can verify the signatures. This is somewhat remedied by designated verifier signatures [22]. With a designated verifier scheme the signer can generate signatures that are only verifiable by a set of users of the signer's choosing. However, when designating the set it is necessary to know the public keys of the members of the set, hence such schemes aren't immediately applicable to the secret-handshake setting in which users are relying on the handshakes themselves to discover group members. Finally, although our handshake schemes build on previous work in identity-based cryptography, identity-based signatures (see, for example [28, 12]) aren't appropriate here because they aren't intended to restrict verification to group members.

MATCHMAKING. The setting of private matchmaking [30] is similar to ours in that the goal is to allow members of the same group to authenticate each other. However, it's quite possible that non-members will be able to identify members. For example, in the schemes in [30] any user may search for air marshals by generating the key corresponding to the term "air marshals".

ACCUMULATORS. A group of users who wish to recognize each other without the use of membership lists or a central authority may choose to form an object called an accumulator and witnesses for each user in the accumulator [5, 2, 19, 10]. To authenticate each other, two users exchange witnesses and perform operations that result in the original accumulator if indeed they are both members. Although accumulators can be used to achieve anonymous authentication they are ill-suited to the secret-handshake problem for the following reasons. First, the accumulator is an untraceable object that can be used to verify (but not prove) membership. We require that membership can only be proven to other group members. Second, adjustments to both the accumulator and the witnesses are necessary when members leave the group. We present a scheme in which both proofs and verifications of membership require traceable keys and no modification of existing keys is needed when membership changes.

Finally, we note that we provide traitor tracing in the sense of [15]. That is, if an adversary uses a compromised user's keys to engage in handshakes the adversary is not authorized to perform, it is possible to trace the identity of the compromised user by examining the handshake transcripts.

3. An Example

3.1. Preliminaries

Pairing-based cryptography is finding an ever-expanding field of applications, ranging from identity-based encryption [6], to signature schemes [21, 7], to key agreements [27]. In this paper, we use pairing-based cryptography to perform secret handshakes.

Before we give an example of our protocol, we remind the reader that pairing-based cryptography is based on bilinear maps over groups of large prime order. For example, if $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of some large prime order q, then $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ is called a bilinear map if for all $a, b \in \mathbb{Z}_q$, $P, Q \in \mathbb{G}_1$ we have $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.

Modified Weil and Tate pairings on supersingular elliptic curves are examples of such bilinear maps that are efficiently computable, non-degenerate,[1] and for which the *Bilinear Diffie-Hellman Problem* is assumed to be hard, *i.e.,* it is assumed that, given P, aP, bP, cP for random $a, b, c \in \mathbb{Z}_q$ and $P \in \mathbb{G}_1$, it is hard to compute $\hat{e}(P, P)^{abc}$. Armed with such a particular map $\hat{e}$ and a hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ that maps from arbitrary strings to points in $\mathbb{G}_1$, we can now describe our secret handshake protocols by way of an example.

3.2. Protocol Sketch

Let's consider a user Alice who lives in a country with a questionable human-rights record. The ministry of transportation in that country possesses a master secret $t \in \mathbb{Z}_q$, and issues driver's licenses to all drivers who have passed the driving test. For Alice, this license comes in the way of a pseudonym and a secret point T_A in $\mathbb{G}_1$. Let's say Alice's driver's license looks like this:

$$(\text{“p65748392a”}, T_A)$$

where $T_A = tH_1(\text{“p65748392a-driver”})$. Alice can show her pseudonym to anyone, but keeps her secret point secret.

The ministry for transportation also issues credentials for traffic cops. Bob is such a traffic cop, and this is his traffic

[1] *i.e.,* $\hat{e}(P, Q)$ does not map to the identity for all P and Q

cop credential:

$$(\text{“xy6542678d”}, T_B)$$

where $T_B = tH_1(\text{“xy6542678d-cop”})$.

Alice is on her way to a secret meeting of a pro-democracy movement of which she is a member. Since she is late, she is speeding on the highway and gets pulled over by Bob. Bob demands to see Alice's driver's license. Alice wants to make sure that Bob is a real cop, and not an impostor. She therefore asks him for his pseudonym, which he sends to her:

$$\text{Bob} \xrightarrow{\text{“xy6542678d”}} \text{Alice}$$

Alice, in return, sends her pseudonym to Bob:

$$\text{Alice} \xrightarrow{\text{“p65748392a”}} \text{Bob}$$

Now, Alice generates a session key K_A by calculating

$$K_A = \hat{e}(H_1(\text{“xy6542678d-cop”}), T_A)$$

By calculating the session key this way, Alice makes sure that she will only end up communicating with Bob if he is a real cop. Bob also calculates a session key K_B by calculating

$$K_B = \hat{e}(T_B, H_1(\text{“p65748392a-driver”}))$$

A simple calculation, using the bilinear properties of $\hat{e}$, shows that these two session keys are, in fact, the same. Once Alice notices that she can communicate with Bob using her session key, she will be convinced that Bob is indeed a cop. Bob, on the other hand, will have learned that Alice is a legitimate vehicle operator. Note that an impostor Igor in Bob's stead might have sent his own pseudonym, but he would not have been in possession of a secret point T_I that corresponds to that pseudonym, and would therefore not have been able to calculate the correct session key.

Alice gets away with a warning and drives off to her meeting. The pro-democracy movement also has a master secret m, and has issued all its members credentials. Alice's credential looks like this:

$$(\text{“y23987447y”}, M_A)$$

where $M_A = mH_1(\text{“y23987447y-member”})$. At the meeting she runs into Claire. Alice has never met Claire before and is worried that she might be with the secret police, rather than the pro-democracy movement. Naturally, Claire (who in fact is a legitimate member of the movement) has the same worries about Alice. Neither of them wants to authenticate herself as a member of the movement unless the other one is a legitimate member herself. So Alice sends her pseudonym to Claire, and receives Claire's pseudonym in return. Let's say Claire's pseudonym is k61932843u. Alice calculates a session key as follows:

$$K_A = \hat{e}(H_1(\text{“k61932843u-member”}), M_A)$$

Claire, on the other side, does the corresponding calculation with Alice's pseudonym and her own secret point M_C. Alice and Claire then verify that they can communicate with each other using their respective session keys, and are thus each convinced that the other is a member of the secret movement.

Alice now meets Dolores, and follows the same protocol with her. Once she generates the session key with Dolores, she encrypts a number N under that session key, and asks Dolores to send her back $N+1$. The reply she receives, however, does not decrypt to $N+1$. Alice has now reason to believe that Dolores is not, in fact, a legitimate member of the movement. Alice can nonetheless rest assured that Dolores learned nothing about Alice's membership in any secret organization. In fact, from Dolores' point of view, the secret handshake was indistinguishable from one in which Alice had used, say, her driver's license instead of her membership credentials for the secret movement.

We end this section with a few observations about the above example. First, we point out that the protocol as presented above is a simplification of the actual protocol we're proposing in this paper. For a more detailed treatment of our protocol, see Section 4. Second, our protocol can handle mutual authentication of different roles (*e.g.,* “cop” vs. “driver”). In the driver's license example, Bob presumably has little incentive to hide his role from Alice, so a more traditional authentication protocol would have done the trick. Imagine, however, a scenario in which both roles want to hide their identities. Let's say Radio Liberty operates streaming audio servers inside Alice's country, and Alice is a subscriber to that service. The servers don't want to authenticate themselves as Radio Liberty outlets unless it's to a legitimate subscriber, and the subscribers don't want to authenticate themselves as Radio Liberty listeners unless it's to a real Radio Liberty server. It's easy to see how our protocol can handle this case. Lastly, in our example we did not address such issues as revocation or linkability, which are instead addressed in Section 7.

4. Secret-Handshake Schemes

4.1. Definitions

In this section we introduce the basic definition of a *secret-handshake scheme*. A secret-handshake scheme operates in a universe consisting of a set $\mathcal{U}$ of possible users, a set $\mathcal{G}$ of groups in which users may be enrolled, and a set $\mathcal{A}$ of administrators who create groups and enroll users in

$$1.\quad A \xrightarrow{\mathsf{id}_A, n_A} B$$
$$2.\quad A \xleftarrow{\mathsf{id}_B, n_B, V_0} B$$
$$3.\quad A \xrightarrow{V_1} B$$

The various symbols denote:

$\mathsf{id}_A, \mathsf{id}_B$:	A's and B's chosen pseudonyms
n_A, n_B :	random nonces, generated by A and B
V_0 :	$H_2(\hat{e}(H_1(\mathsf{id}_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 0)$
V_1 :	$H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 1)$
H_1 :	Collision-resistant hash function from strings to $\mathbb{G}_1$
H_2 :	Collision-resistant hash function from strings to strings with fixed-length output, e.g. SHA-1
$\mathsf{priv}_A, \mathsf{priv}_B$:	A's and B's secret points

Figure 1. PBH.Handshake

groups. (We note that the term "group" refers to a *grouping* of users, rather than the mathematical definition of a group.) These sets can be infinite, and do not need to be specified in advance. A *secret-handshake scheme* SHS consists of the following algorithms:

- $\mathsf{SHS.CreateGroup} : \mathcal{G} \to \{0,1\}^*$
  When $\mathsf{SHS.CreateGroup}(G)$ is executed by an administrator $A \in \mathcal{A}$, a group secret $\mathsf{GroupSecret}_G \in \{0,1\}^*$ is output for the group G;

- $\mathsf{SHS.AddUser} : \mathcal{U} \times \mathcal{G} \times \{0,1\}^* \to \{0,1\}^*$
  When run by an administrator on input $(U, G, \mathsf{GroupSecret}_G)$, enrolls U in the group G (denoted $U \in G$) by creating a user secret $\mathsf{UserSecret}_{U,G} \in \{0,1\}^*$ to be given to the user U;

- $\mathsf{SHS.Handshake}(A, B)$
  Specifies a protocol to be executed between users A and B, which, upon completion[2] ensures that B discovers $A \in G$ if and only if A also discovers $B \in G$;

- $\mathsf{SHS.TraceUser} : \{0,1\}^* \to \mathcal{U}$
  An algorithm run by the system administrator, which, given a transcript T of interaction of a user U with one or more users, outputs the identity of the user whose keys were used by U during the interaction.

- $\mathsf{SHS.RemoveUser} : \{0,1\}^* \times \mathcal{U} \to \{0,1\}^*$
  On input $(\mathsf{RevokedUserList}, U)$, inserts U into RevokedUserList.

[2]Note that this definition does not guarantee fairness: If A aborts the protocol before sending his final message, he may learn whether or not $B \in G$ without revealing corresponding information about himself. This will not a serious issue, however, as the security definitions in Section 5.1 will guarantee that A must be a member of G to learn anything about B in this fashion.

To keep the presentation clear, we assume here that each user is a member of exactly one group. All results generalize to the case where users are allowed to join multiple groups.

Ideally, if the execution of $\mathsf{SHS.Handshake}(A, B)$ establishes that A and B are members of the same group, it should also have set up a temporary session key for securing further communication between A and B. Although not required for the security definitions below, this additional requirement is satisfied by our schemes.

4.2. A Concrete Secret-Handshake Scheme

We now present a concrete secret-handshake scheme based on bilinear pairings. We call this scheme *Pairing-Based Handshake* (PBH).

Our system uses a computable, non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ for which the Bilinear Diffie-Hellman Problem is assumed to be hard (see Section 3 for definitions of these terms). Modified Weil or Tate pairings on supersingular elliptic curves are examples for such maps. We also assume there are two hash functions H_1, H_2 available: $H_1 : \{0,1\}^* \to \mathbb{G}_1$ maps arbitrary strings to points in $\mathbb{G}_1$, and H_2 is a collision-resistant hash function taking arbitrary strings as input (such as SHA-1).

PBH.CreateGroup. The administrator A sets the group secret $\mathsf{GroupSecret}_G$ to be a random number $s_G \in \mathbb{Z}_q$ (where q is the order of both $\mathbb{G}_1$ and $\mathbb{G}_2$).

PBH.AddUser. To add a user U to the group G, the administrator A does the following: First, it generates a list of random "pseudonyms" $\mathsf{id}_{U1}, \ldots, \mathsf{id}_{Ut} \in \{0,1\}^*$ for U, where t is chosen to be larger than the number of handshakes U will execute before receiving new user secrets. Since only the administrator and the user itself know the identity U,

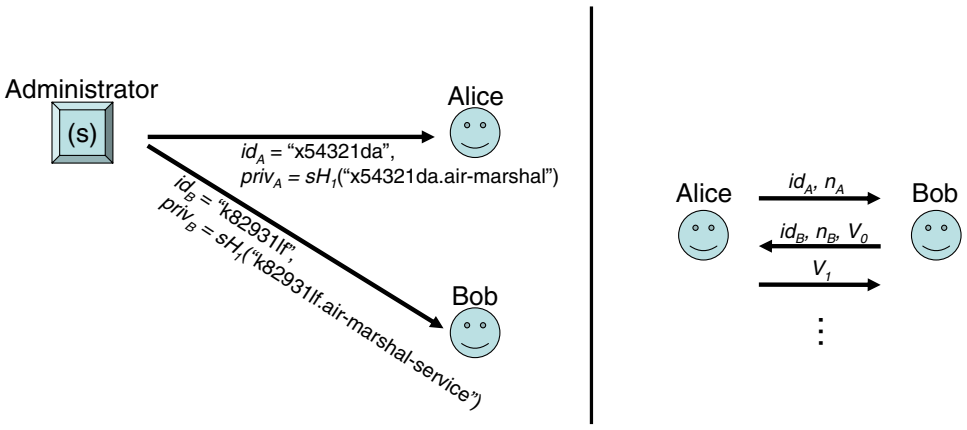

Figure 2. Pairing-Based Handshake with Roles. The administrator issues credentials to users (left), who then perform secret handshakes (right).

only the administrator and the user can link any id_{Ui} back to U. The administrator then calculates a corresponding list of *secret points* priv_{U1}, ..., priv_{Ut} as

$$\mathsf{priv}_{Ui} = s_G H_1(\mathsf{id}_{Ui})$$

where $s_G = \mathsf{GroupSecret}_G$. This list of pseudonyms together with the list of secret points is given as $\mathsf{UserSecret}_{U,G}$ to U.

PBH.Handshake. Let A and B be two users who wish to conduct a secret handshake. A pulls from his user secret an unused pseudonym $\mathsf{id}_A \in \{\mathsf{id}_{Ai}, \ldots, \mathsf{id}_{At}\}$, together with the corresponding secret point priv_A. B likewise pulls id_B and priv_B. First, A sends his pseudonym, along with a random nonce n_A, to B (see Figure 1). B replies with her pseudonym, a nonce n_B of her choosing, and a value V_0. A verifies that

$$V_0 = H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 0)$$

and replies with V_1 (message 3 in Figure 1) . B verifies

$$V_1 \stackrel{?}{=} H_2(\hat{e}(H_1(\mathsf{id}_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 1)$$

If both verifications succeed[3], then A and B can create a shared secret S for future communication. A calculates the shared secret like this:

$$S = H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 2)$$

B calculates the same shared secret S as:

$$S = H_2(\hat{e}(H_1(\mathsf{id}_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 2)$$

PBH.TraceUser. Given a transcript of a handshake between user A and B, the administrator can easily recover the pseudonyms id_A and id_B and look up which users these pseudonyms had been issued to.

PBH.RemoveUser. To remove a user U from the group G, the administrator looks up the user secret $(\mathsf{id}_{U1}, \ldots, id_{Ut}, \mathsf{priv}_{U1}, \ldots, \mathsf{priv}_{Ut})$ it has issued to U and alerts every other user to abort any handshake should they find themselves performing the handshake with a user using any pseudonym $\mathsf{id}_U \in \{\mathsf{id}_{U1}, \ldots, \mathsf{id}_{Ut}\}$.

4.3. A Concrete Secret-Handshake Scheme with Roles

While we defined secret-handshake schemes in terms of group membership, we can easily extend the PBH scheme to handle roles in the sense described in Section 1 and Section 3. Here is a pairing-based secret-handshake scheme with roles (PBH-R):

PBH-R.CreateGroup. This step is identical to PBH.CreateGroup.

PBH-R.AddUser. Let $R \in \{0,1\}^*$ be an arbitrary string describing a role within group G. To add a user U to the group G in role R, the administrator A does the following: First, it generates random pseudonyms $\mathsf{id}_{Ui} \in \{0,1\}^*$ for $i = 1, \ldots, t$ (this is identical to PBH.AddUser). The administrator then calculates the secret points priv_{Ui} as

$$\mathsf{priv}_{Ui} = s_G H_1(\mathsf{id}_{Ui} \| R)$$

where $s_G = \mathsf{GroupSecret}_G$. The list of pseudonyms together with the secret points is given as $\mathsf{UserSecret}_{U,G}$ to U.

PBH-R.Handshake. Let A and B be using respective pseudonyms id_A and id_B, their respective secret points priv_A

[3] They will either both succeed or both fail.

and priv_B, and their respective roles R_A and R_B. First, A sends his pseudonym, along with a random nonce n_A, to B:

$$A \xrightarrow{\mathsf{id}_A, n_A} B$$

B decides that she only wants to perform a secret handshake with someone in role R'_A. She replies with her pseudonym, a nonce n_B of her choosing, and a value V_0:

$$A \xleftarrow{\mathsf{id}_B, n_B, V_0} B$$

where

$$V_0 = H_2(\hat{e}(H_1(\mathsf{id}_A \| R'_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 0)$$

A decides that he only wants to perform the handshake with someone in role R'_B. He verifies that

$$V_0 = H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B \| R'_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 0)$$

and replies with V_1:

$$A \xrightarrow{V_1} B$$

where

$$V_1 = H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B \| R'_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 1)$$

B verifies

$$V_1 \stackrel{?}{=} H_2(\hat{e}(H_1(\mathsf{id}_A \| R'_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 1)$$

The two verifications either both succeed or both fail. If they both succeed, then B has authenticated A as a member of group G in role $R'_A = R_A$, and A has authenticated B as a member of group G in role $R'_B = R_B$. A and B can now create a shared secret for future communication. A calculates the shared secret like this:

$$S = H_2(\hat{e}(\mathsf{priv}_A, H_1(\mathsf{id}_B \| R'_B)) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 2)$$

B calculates the shared secret like this:

$$S = H_2(\hat{e}(H_1(\mathsf{id}_A \| R'_A), \mathsf{priv}_B) \| \mathsf{id}_A \| \mathsf{id}_B \| n_A \| n_B \| 2)$$

See Figure 2 for examples of PBH-R.AddUser and PBH-R.Handshake.

PBH-R.TraceUser and PBH-R.RemoveUser. These steps are identical to PBH.TraceUser and PBH.RemoveUser.

In the following formal treatment of secret handshakes, we will only consider secret schemes without roles. All our definitions, arguments, and security proofs, however, easily extend to handshake schemes with roles.

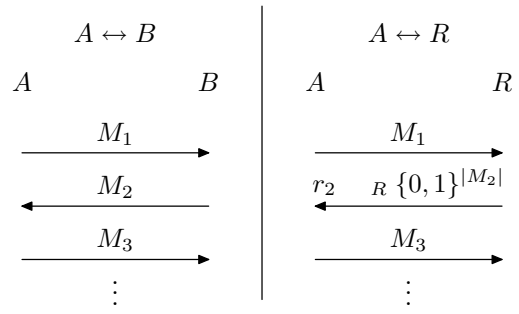

Figure 3. Interaction between A and B compared to interaction between A and a random simulation R.

5. Security for Secret-Handshake Schemes

5.1. Definitions

Before defining security for a secret-handshake scheme, we first introduce some auxiliary definitions.

Security Parameter: All primitives discussed in this paper take an implicit *security parameter*. Typically, this is the length of the prime modulus used in cryptographic operations (in our case, the length of q).

Negligible: Informally, a function $\varepsilon(t)$ is negligible when $\varepsilon(t) \approx 0$ for big enough t. Formally, a function $\varepsilon(t)$ is *negligible in t* if for all polynomials $p(\cdot)$, $\varepsilon(t) \leq 1/p(t)$ for sufficiently large t. When t is the security parameter, we simply say ε is *negligible*.

Random Simulation: A random simulation R of a participant in a protocol replaces all outgoing messages with uniformly-random bit strings of the same length. (See Figure 3.)

Interaction: We denote by $\mathrm{A.Handshake}(A,B)$ an alteration of $\mathsf{SHS.Handshake}(A,B)$ by an adversarial player A. The adversary may choose to respond differently than is specified in the original protocol, and may choose to terminate the protocol early. What each party learns may be different than in the original secret-handshake protocol.

We say that *A interacts with B* when $\mathrm{A.Handshake}(A,B)$ is executed. When A executes a handshake with a random simulation, we write this as $\mathrm{A.Handshake}(A,R)$, and say that *$A$ interacts with a random simulation*.

Group Member Impersonation

To motivate the following definitions, consider an adversary A that has as its goal to learn how to impersonate members of a certain group G^*. A interacts with players of the

system, corrupts some users, communicates with legitimate members of G^*, and eventually picks a target user U^* and attempts to convince U^* that A is a member of G^*. Intuitively, if A does not obtain secrets for any other $U \in G^*$, then it should remain unable to convince U^* of its membership in G^*.

An additional property we would like our scheme to have is the ability to trace the user secrets a successful adversary might be using. We wish to argue that if A *is* able to convince U^* that $A \in G^*$, then A must be using secrets obtained by some user $U \in G^*$, and the transcript of A's interaction with U^* will allow an administrator to identify U. Consequently, U may be placed on a revocation list to prevent the adversary from further use of U's stolen secrets. We model this by saying there is an efficient algorithm which, given the transcript of A's interaction with U^* (but not necessarily access to A's internal state), extracts the identity of some user $U \in G^*$ whose secrets A has been using. This motivates the definition of impostor tracing below.

We define the *Member Impersonation Game* for a randomized, polynomial-time adversary A:

Step 1: The adversary A interacts with users of its choice, and obtains secrets for some users $\mathcal{U}' \subseteq \mathcal{U}$.

Step 2: A selects a target user $U^* \notin \mathcal{U}'$ satisfying $U^* \in G^*$.

Step 3: A attempts to convince U^* that $A \in G^*$; that is, A attempts to construct the correct responses in the protocol $\mathsf{SHS.Handshake}(A, U^*)$.

We say that A *wins the Member Impersonation Game* if it engages correctly in $\mathsf{SHS.Handshake}(A, U^*)$ when $U^* \in G^*$. We define A's *impersonation advantage* AdvMIG_A as the following quantity:

$$\mathsf{AdvMIG}_A := \Pr[\, A \text{ wins Member Impersonation Game} \,].$$

We will also consider A's conditional advantage restricted to the occurrence of event E:

$$\mathsf{AdvMIG}_A^E := \Pr[\, A \text{ wins Member Impersonation Game} \,\big|\, E \,].$$

These probabilities are taken over the randomness in the algorithms $\mathsf{SHS}.*$, the coin flips of A, and the coin flips of all participating users.

We are ready to define two notions of security using the Member Impersonation Game.

Impersonation Resistance: Suppose A never corrupts a member of the target group G^*. Then $\mathcal{U}' \cap G^* = \emptyset$. The secret-handshake scheme SHS is said to ensure *impersonation resistance* if $\mathsf{AdvMIG}_A^{\mathcal{U}' \cap G^* = \emptyset}$ is negligible for all A.

In other words, if an adversary never corrupts a member of its target group, it has only a negligible chance of impersonating as a member of the target group.

Impersonator Tracing: Let T be a transcript of the interaction of A and U^*. The secret-handshake scheme SHS is said to permit *impostor tracing* when

$$\left|\Pr[\mathsf{SHS.TraceUser}(T) \in \mathcal{U}' \cap G^*] - \mathsf{AdvMIG}_A\right|$$

is negligible for all A.

In other words, with success probability very close to that of the adversary, an administrator can determine which user's secrets A has obtained to perform its impersonation.

Group Member Detection

To motivate the following definitions, consider an adversary A that has as its goal to learn how to identify members of a certain group G^*. A interacts with players of the system, corrupts some users, picks a target user U^*, and attempts to learn if $U^* \in G^*$.

Intuitively, if A does not obtain secrets for any other $U \in G^*$, then it should remain clueless when detecting whether $U^* \in G^*$. In other words, the final interaction with U^* should yield *no new information* to the adversary unless it has already obtained secrets from another member of G^*.

To model this formally, we consider the behavior of an adversary in an environment where it is either allowed to interact with its target user U^* or it is instead presented with a random simulation, and asking it to tell the difference. An adversary unable to distinguish between U^* and R quantitatively learns nothing new about U^* (let alone whether $U^* \in G^*$). This motivates the definition of detection resistance given below.

An additional property we would like our scheme to have is the ability to trace which user's secrets a successful adversary is using. We wish to argue that if A *is* able to distinguish between R and some user $U^* \in G^*$, then A must have already corrupted some other user $U \in G^*$, and this U is revealed by A. We model this by saying there is an efficient algorithm which, given transcripts of A's interaction with the system (but not necessarily access to A's internal state), extracts the identity of some user $U \in G^*$ whose secrets A has been using. This motivates the definition of detector tracing below.

We define the *Member Detection Game* for a randomized, polynomial-time adversary A:

Step 1: The adversary A interacts with users of its choice, and obtains secrets for some users $\mathcal{U}' \subseteq \mathcal{U}$.

Step 2: A selects a target user $U^* \notin \mathcal{U}'$.

Step 3: A random bit $b \leftarrow \{0,1\}$ is flipped.

Step 4: If $b = 0$, A interacts with U^*. If $b = 1$, A interacts with a random simulation R.

Step 5: The adversary outputs a guess b^* for b.

We say that A *wins the Member Detection Game* when $b^* = b$. We define A's *advantage* AdvMDG_A as the following quantity:

$$\mathsf{AdvMDG}_A := |\Pr[A \text{ wins Member Detection Game}] - 1/2|.$$

We will also consider A's conditional advantage restricted to the occurrence of event E:

$$\mathsf{AdvMDG}_A^E := \left|\Pr[\, A \text{ wins MDG} \;\middle|\; E\,] - 1/2\right|.$$

These probabilities are taken over the randomness in the algorithms $\mathsf{SHS}.*$, the randomness of R, the coin flips of A, and the coin flips of all participating users.

We are ready to define two notions of security using the member detection game.

Detection resistance: Let G_{U^*} be the group to which U^* belongs, and suppose A never corrupts a member G_{U^*}. Then $\mathcal{U}' \cap G_{U^*} = \emptyset$. The secret-handshake scheme SHS is said to ensure *detection resistance* if $\mathsf{AdvMDG}_A^{\mathcal{U}' \cap G_{U^*} = \emptyset}$ is negligible for all A.

In other words, if an adversary never corrupts a member of its target user's group, it has only a negligible chance of distinguishing the target user's messages from random strings.

Detector tracing: Let T be a transcript of the interaction of A and U^*, and let G_{U^*} be the group to which U^* belongs. The secret handshake scheme SHS is said to permit *detector tracing* when

$$\left|\Pr[\mathsf{SHS.TraceUser}(T) \in \mathcal{U}' \cap G_{U^*}] - \mathsf{AdvMDG}_A\right|$$

is negligible for all A.

In other words, with success probability very close to that of the adversary, an administrator can determine which user's secrets A has obtained to perform its unauthorized detection.

5.2. Security of the Pairing-Based Handshake

We claim that if the Bilinear Diffie-Hellman problem is hard, the simple Pairing-Based Handshake Scheme outlined in Section 4.2 provably satisfies the security properties outlined in the previous section. We provide the statements of security here; the security analysis is outlined in the appendix.

With straightforward modifications to the security analysis, analogous security properties can be defined and shown to hold for the secret-handshake scheme with roles outlined in Section 4.3.

Hardness of BDH Problem: We say that *the Bilinear Diffie-Hellman Problem (BDH) is hard* if, for all probabilistic, polynomial-time algorithms B,

$$\mathsf{AdvBDH}_B := \Pr[B(P, aP, bP, cP) = \hat{e}(P,\, P)^{abc}]$$

is negligible in the security parameter. This probability is taken over random choice of $P \in \mathbb{G}_1$ and $a, b, c \in \{1, \ldots, q\}$ where q is the order of $\mathbb{G}_1$.

We now state the security claims for the Pairing-Based Handshake. We outline proofs of Theorems 1 and 4 in the appendix. For this analysis we model the hash functions H_2 and H_1 as random oracles [4].

Let A be a probabilistic, polynomial time adversary. We denote by Q_{H_2} the number of distinct queries A makes to H_2, and we denote by Q_{H_1} the number of distinct queries A makes to H_1. We write $e \approx 2.78$ as the base of the natural logarithm.

Theorem 1 *Suppose A is a probabilistic, polynomial time (PPT) adversary. There is an PPT algorithm B such that*

$$\begin{aligned}\mathsf{AdvMIG}_A \;\leq\; & \Pr[\,\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G^*\,] \\ & + e\, Q_{H_1} Q_{H_2} \cdot \mathsf{AdvBDH}_B + \varepsilon,\end{aligned}$$

where ε is negligible in the security parameter.

Intuitively, Theorem 1 says that the probability that an adversary succeeds in the Member Impersonation Game is less than the probability that he is traceable plus the probability that he can be used to solve the Bilinear Diffie-Hellman problem.

The hardness of the BDH problem then implies the following:

Corollary 2 (PBH Impersonator Tracing)
Suppose A is a probabilistic, polynomial time adversary. If the BDH problem is hard, then

$$\left|\Pr[\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G^*] - \mathsf{AdvMIG}_A\right|$$

is negligible.

In other words, the Pairing-Based Handshake satisfies the definition of Impersonator Tracing outlined in the previous section.

Note that if $\mathcal{U}' \cap G^* = \emptyset$, then $\Pr[\,\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G^*] = 0$. This immediately yields the following:

Corollary 3 (PBH Impersonation Resistance)
Suppose A is a probabilistic, polynomial time adversary. If the BDH problem is hard, then $\mathsf{AdvMIG}_A^{\mathcal{U}' \cap G^* = \emptyset}$ is negligible.

In other words, the Pairing-Based Handshake satisfies the definition of Impersonation Resistance outlined in the previous section.

We now turn our attention to the Member Detection Game. Using the notation from Section 4, we claim the following.

Theorem 4 *Suppose A is a probabilistic, polynomial time adversary. There is an PPT algorithm B such that*

$$\mathsf{AdvMDG}_A \leq \Pr[\,\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G(U^*)\,] + e\,Q_{H_1}Q_{H_2} \cdot \mathsf{AdvBDH}_B + \varepsilon,$$

where ε *is negligible in the security parameter.*

Intuitively, Theorem 4 says that the probability that an adversary succeeds in the Member Detection Game is less than the probability that he is traceable plus the probability that he can be used to solve the Bilinear Diffie-Hellman problem.

The hardness of the BDH problem then implies the following:

Corollary 5 (PBH Detector Tracing)
Suppose A is a probabilistic, polynomial time adversary. If the BDH problem is hard, then

$$\left|\Pr[\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G(U^*)] - \mathsf{AdvMDG}_A\right|$$

is negligible.

In other words, the Pairing-Based Handshake satisfies the definition of Detector Tracing outlined in the previous section.

Note that if $\mathcal{U}' \cap G(U^*) = \emptyset$, then $\Pr[\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G(U^*)] = 0$. This immediately yields the following:

Corollary 6 (PBH Detection Resistance)
Suppose A is a probabilistic, polynomial time adversary. If the BDH problem is hard, then $\mathsf{AdvMDG}_A^{\mathcal{U}' \cap G(U^*) = \emptyset}$ *is negligible.*

In other words, the Pairing-Based Handshake satisfies the definition of Detection Resistance outlined in the previous section.

5.3. Additional Security Notions

In this section we consider several additional security notions that may be desirable in a secret handshake scheme: *forward repudiability*, *indistinguishability to eavesdroppers*, *collusion resistance*, and *unlinkability*. Forward repudiability is a desirable property that may optionally be satisfied by a secret handshake scheme, and is in fact satisfied by our scheme. Indistinguishability to eavesdroppers, collusion resistance, and unlinkability follow the security definitions given in Section 5.1, and are discussed only for completeness.

Forward Repudiability: Suppose honest users U_1 and U_2 interact, and they both learn they are members of the same group G. It should not be possible for U_2 to prove to a third party that U_1 is a member of the group G – even if U_2 reveals its own secrets. For example, if U_2's secrets are later compromised, the transcript of U_1 and U_2's interaction together with U_2's secrets should not constitute a proof of U_1's membership in G.

Note that U_2 and U_1 may not be able to conceal the fact that they communicated (this might require the use of steganographic techniques, and is outside the scope of this paper). However, any evidence (transcripts, U_2's secrets, etc) should not provide a non-repudiable proof that U_1 is a member of G. We refer to this as *forward repudiability*.

Forward repudiability is not achieved by schemes that rely on credentials that support non-repudiation for the underlying authentication. For example, consider a scheme that gave every member of a group G a public-key certificate attesting to their group membership, and a shared group secret key under which they could encrypt those certificates to limit their exchange to only other group members. Such a certificate, together with a transcript showing a demonstration of group membership using that public key would be sufficient to publicly implicate the sender of the certificate as a group member.

Forward repudiability follows immediately in our scheme: notice that U_2 always has enough information to generate the entire transcript between U_1 and U_2. So this transcript could have been completely faked by U_2, and cannot be used to convince a third party of U_1's membership in G.

Indistinguishability to Eavesdroppers: Consider an adversary A who corrupts some set $\mathcal{U}'$ of users, interacts with others, and observes a transcript of $\mathsf{SHS.Handshake}$ between users U_1^*, $U_2^* \notin \mathcal{U}'$ (possibly of A's choice). The adversary A should be unable to learn anything from this handshake that it did not already know, including whether U_1^* and U_2^* belong to the same group or to different groups.

We model this by giving A either a transcript of the real handshake between U_1^* and U_2^*, or giving it a transcript of a handshake between random simulations, and asking it to tell the difference.

We define A's *distinguishing advantage* as follows. Let T_{Real} be the transcript of $\mathsf{SHS.Handshake}(U_1^*, U_2^*)$, and let T_{Rand} be a transcript of $\mathsf{SHS.Handshake}(R, R)$. Define

$$\mathsf{AdvDST}_A := \left|\Pr[A(T_{\mathsf{Real}}) = 1] - \Pr[A(T_{\mathsf{Rand}}) = 1]\right|.$$

The secret-handshake scheme SHS is said to provide *indistinguishability to eavesdroppers* when AdvDST_A is negligible for all adversaries A.

Our secret handshake scheme satisfies this additional security property; in fact, this follows from detection resistance. A proof is outlined in the appendix.

Note that communication outside the protocol (*e.g.,* the presence of continued communication after the handshake)

may reveal the success or failure of the protocol. Protecting against such traffic analysis is outside the scope of our paper; approaches such as steganographic techniques may be appropriate in this context.

Collusion Resistance and Traitor Tracing: The system must remain secure even if collections of users pool their secrets in an attempt to undermine the system (collusion resistance); if a coalition of users manages to detect or impersonate group members, it should be possible to detect at least one member of the coalition (traitor tracing). Collusion resistance and traitor tracing follow immediately from the definitions given in Section 5.1: a pool of colluding users can be modeled as a virtual adversary that "corrupts" the set of colluding users and uses their secrets. Nevertheless, this security notion is worth restating because is the main reason why variations of the traditional Diffie-Hellman based key exchange protocol fail to produce a secret handshake, helping to explain the motivation for using pairing-based cryptography.

To see how collusion resistance breaks down in a Diffie-Hellman-based analogue of this scheme, consider a scenario in which Diffie-Hellman key agreement is used in a group $\mathbb{Z}/N\mathbb{Z}$ where $N = pq$ is a product of two large primes. Alice, a member of group G_A, would have a private key x_A and a public key $(g_{G_A})^{x_A}$ derived from her private key and the secret group base g_{G_A}. By the hardness assumption of the RSA problem, Alice is unable to compute the group secret g_{G_A} as it would require computing the x_Ath root of her public key.

Bob would have analogous quantities; an attempted secret handshake would then be a standard Diffie-Hellman key exchange between Alice and Bob and a verification of its success. They would obtain a shared secret key $g_{G_A}^{x_A x_B}$ if and only if their secret group bases were equal, i.e. $g_{G_A} = g_{G_B}$.

While it is tempting to use Diffie-Hellman based key agreement to implement secret handshakes, this scheme is trivially not collusion-resistant. If a set of members of G_A collude whose secrets x_i satisfy $\gcd(x_1, \ldots, x_t) = 1$, they may compute $\alpha_1, \ldots, \alpha_t \in \mathbb{Z}$ such that $\sum \alpha_i x_i = 1$. They may then compute $\prod_i (g_{G_A}^{x_i})^{\alpha_i} = g_{G_A}$, giving them the untraceable group secret for G_A. This gives them the ability to detect and impersonate arbitrary group members untraceably.

Unlinkability: The schemes presented in Section 4 specify that a user obtains a list of pseudonyms for one-time use. This allows handshakes to be *unlinkable*: If an eavesdropper sees two different handshakes performed by Alice, the content of the handshakes alone are unlinkable

It may be desirable instead to have a system in which a user reuses a single pseudonym together with a single secret point in all handshakes. This prevents a user from running out of pseudonyms, and dramatically decreases the size of revocation lists that may be required for users.

This modification does not undermine the security of our system. Indeed, all the security proofs go through with one minor change. We must introduce the notion of a *modified random simulation* $R(B)$ of a user B. The modified random simulation of B randomizes all messages outgoing from B *except* id_B*, the sole pseudonym used by* B. Under this slightly weaker notion of random simulation and resulting security definitions, the Pairing-Based Handshake with pseudonym reuse is provably secure with slightly weaker security bounds.

In our implementations discussed below, we are using the more efficient, but linkable, version of the protocols.

6. Implementation

6.1. Secret Handshakes in TLS

Section 4.2 presented our basic PBH scheme. In practice, it would be preferable to incorporate such a secret-handshake authentication scheme into widely used secure communication protocols. We present here a method to securely use a PBH protocol to authenticate the standard SSL/TLS handshake [16], requiring only small modifications of two of the TLS handshake messages. To maintain our proofs of security, we require only that the verification values (H_0 and H_1 in the notation of section 4.2) each participating party sends to the other to prove their ability to compute the shared secret, are a keyed pseudorandom function (PRF) of: the shared secret, both parties' identities, independent randomness contributed by each party, and some factor that makes each party's verification value different from the other's (thus forcing each to independently prove possession of the shared secret).

Following the notation in section 4.2, we modify the standard TLS handshake as follows: TLS begins with the client (in our case, the initiating party) sending the server (the responding party) a *ClientHello* message. This message contains both a random nonce and a timestamp, which together will correspond to n_A in our PBH protocol. The server responds with a *ServerHello* message, which contains an independent nonce and timestamp generated by the server, corresponding to n_B. These messages provide the exchanges of randomness needed by the PBH protocol.

The server then sends a *ServerKeyExchange* message, which is typically used to exchange additional keying information necessary for anonymous or ephemeral key exchange methods. It contains an indication of the algorithm being used (*e.g.,* Diffie-Hellman), and a set of parameters necessary for that algorithm (*e.g.,* a Diffie-Hellman public value and parameters). We modify that message for our PBH scheme; the new message contains an indication that

PBH is the algorithm being used and the server's identity, id_B. The server then completes its portion of the exchange by sending *ServerHelloDone*. The client then continues the exchange by sending a *ClientKeyExchange* message, again modified to contain an indication that a PBH scheme is being used and the client's identity, id_A. Each participant now has sufficient information to calculate the shared secret, $\hat{e}_0 = \hat{e}(H(\mathsf{id}_A), \mathsf{priv}_B) = \hat{e}(\mathsf{priv}_A, H(\mathsf{id}_B))$. We take $\hat{e}_0$ to be the TLS *pre-master secret*, used to generate all further encryption and authentication keys.

The remainder of the exchange is unchanged from the standard TLS handshake. The parties exchange *ChangeCipherSpec* messages, which indicate that they should begin to use the keys and algorithms they have just negotiated. They then exchange *Finished* messages, which contain the verification values necessary to allow each of them to confirm that the other has correctly computed the pre-master secret, $\hat{e}_0$, and hence in our case, that the other is a member of the desired group.

These verification values, which correspond to our H_0 and H_1, are computed as follows: first the pre-master secret (*pms*) is used to compute a master secret (*ms*), using:

$$ms = PRF_{TLS}(pms,\ \text{“master secret”},\ random_c \| random_s)$$

where PRF_{TLS} is the TLS keyed pseudo-random function $PRF_{TLS}(secret,\ label,\ seed)$,[4] $\|$ is concatenation, and $random_c$ and $random_s$ are the random nonce and timestamp structures exchanged in the *ClientHello* and *ServerHello* messages, respectively. Each party then uses this master secret to compute their own verification value (vv_i), as:

$$vv_i = PRF_{TLS}(ms,\ label_i,\ MD5(hm) \| SHA1(hm))$$

where $label_{client}$ is “client finished” and $label_{server}$ is “server finished”, and *hm*, or *handshake_messages* is the concatenation of all the previous messages sent by both parties during the handshake.

The resulting protocol meets the security requirements outlined above. The verification values are a keyed pseudorandom function of the shared secret, $\hat{e}_0$, the independent randomness contributed by both parties, and the identities of both parties. Each party's verification value is different, because of the the requirement that each use a different $label_i$.

This combined protocol can be implemented very simply, as it makes no changes to the TLS message flow or key derivation algorithm, and only requires small modifications to two existing TLS messages, which already come in algorithm-specific variants.

[4] TLS's *PRF* combines its arguments using SHA-1, MD5 and both $HMAC_{SHA-1}$ and $HMAC_{MD5}$; details can be found in [16].

6.2. Implementation Choices

We implemented the pairing-based handshake protocol described in Figure 1. We also implemented a secure transport layer protocol following the TLS specification (*i.e.,* we generate MAC and cipher keys from the master secret the same way TLS does, *etc.*).

The security parameters we use are the lengths of two primes, p and q. Typical parameters would be 1024 bits for the length of p and 160 bits for the length of q. We generate the primes such that $p = 12qr - 1$ (for some r large enough to make p be the correct size)[5].

The curve E we use is $y^2 = x^3 + 1$. See [6] for a discussion of the properties of this curve. To implement the hash function H that maps random strings to points in $E(\mathbb{F}_p)[q]$,[6] we simply seed a pseudorandom number generator with the string we want to map, and then generate a pseudorandom point in $E(\mathbb{F}_p)[q]$.

The bilinear map $\hat{e}$ we used is the Tate pairing, with some of the modifications and performance improvements described in [3, 6].

6.3. Measurements

We ran our 100% Java implementation on a 1.8GHz Pentium 4. The table below shows the handshake times for various parameter values, alongside RSA key sizes that are believed to deliver comparable security.

size of q	size of p	handshake time	comparable RSA key size
120 bits	512 bits	0.8sec	512 bits
160 bits	1024 bits	2.2sec	1024 bits
200 bits	2048 bits	11.8sec	2048 bits

These times can be cut in half by a slight alteration to the secret handshake protocol. The protocol as presented in Section 4.2 is designed to minimize the number of rounds. As a result, the parties must compute the results V_i of their respective Tate pairings in series. This may be optimized by rearranging the protocol so that these computations can instead be performed in parallel. This results in one additional message, but approximately halves the running time of the protocol.

We believe that these running times can be substantially improved. The running time of the protocol is dominated by the computation of the Tate pairing, and we have not yet implemented some of the performance enhancements suggested in [3]. Furthermore, our current implementation

[5] Choosing q to be a Solinas prime (*i.e.,* $q = 2^\alpha \pm 2^\beta \pm 1$ for some $\alpha > \beta > 1$) could further improve the performance of our scheme.

[6] $E(\mathbb{F}_p)[q]$ is the set of solutions of E over $\mathbb{F}_p$ of order dividing q.

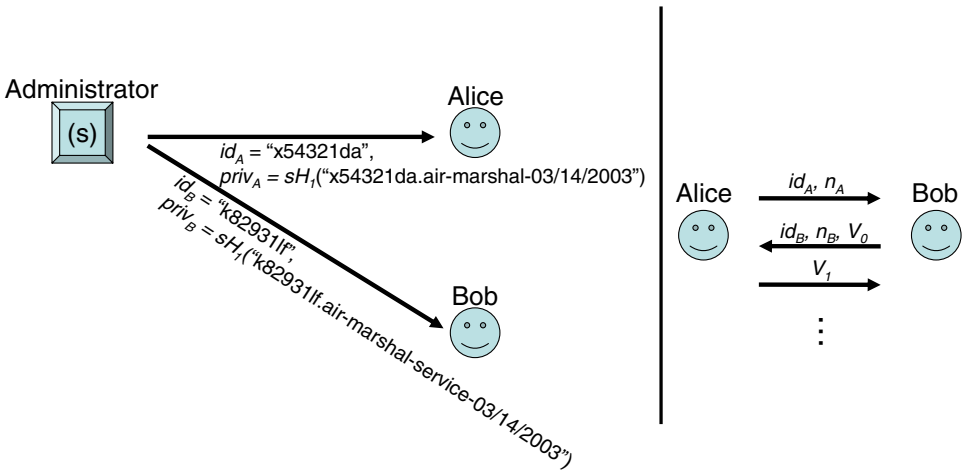

Figure 4. Short-lived credentials: The role manager only issues credentials that are good for a certain, short, period of time.

is purely written in Java. We expect future optimized implementations to be comparable to RSA-based TLS handshakes.

7. Practical Issues

7.1. User and Role Authorization

When a new user wants to assume a certain role in a group, she gets a pseudonym and a secret point for that role from the administrator. The new user may have to be authorized to assume the role, in which case the administrator has to perform user authorization. How this is done is orthogonal to the schemes presented in, and outside the scope of, this paper.

7.2. Revocation

If a user of a system gets compromised and his secret point stolen, then the thief of the secret point can impersonate the compromised user, as well as authenticate other users in the system (and learn their roles). To address this problem, we need a revocation system.

PUBLIC-KEY REVOCATION LISTS. In Section 4.2 we explain how the administrator can publish public-key revocation lists that show which public keys should no longer be trusted. This scales relatively well with the number of users (it only requires work in the order of number of revoked users) but introduces the well-known consistency problem for Certificate Revocation Lists – we need to make sure that all users have an up-to-date and correct view of the current public-key revocation lists.

SHORT-LIVED CREDENTIALS. To address this consistency problem, we could address revocation using short-lived credentials (borrowing an idea from [6]). In addition to folding the user's role into her secret point (see Section 4.3), the administrator could also fold in the date at which the secret point is valid, as shown in Figure 4 (compare with Figure 2). With a slight modification in the secret-handshake protocol (again, following the suggestions in [6]) users can then make sure that their peers in the handshake protocol use fresh keys.

Although users obtain new secret points in regular intervals from the administrator, it turns out that they do not have to re-authenticate themselves to the administrator. Using identity-based encryption, or some other suitable scheme, the administrator could just publish fresh credentials for unrevoked users, encrypted under their current pseudonym.

7.3. Protocol Deployment

While we have proven that an observer of a secret handshake between users U_1 and U_2 cannot learn whether they belong to the same group at the end of the protocol, the observer can certainly learn one thing – that U_1 and U_2 executed our protocol. In fact, if our scheme (as proposed) is implemented as a TLS cipher suite, then the two parties will exchange a cipher suite designator that clearly shows that they wish to engage in a secret handshake. If a government makes it illegal to perform our protocol, with penalties similar to those of belonging to certain illegal groups, then using our secret handshake protocol may actually bring more problems than not using it.

Also, even though the observer would not be able to tell whether U_1 and U_2 belong to the same group at the end of

the protocol, he or she may actually learn more information by monitoring U_1's and U_2's communications after the execution of the protocol. If they continue talking with each other, then they were probably able to authenticate each other as members of the same group. Furthermore, if it is known that there exists only one group G that uses our secret handshake scheme, then both U_1 and U_2 must belong to that group.

These and other practical deployment issues can be mitigated by using some form of anonymous communication, which makes it hard to find out exactly who is engaging in a secret handshake. However, anonymizers can be subject to the same caveats (everybody has to use them, they must not be illegal, *etc.*) as secret handshake protocols.

In summary, our secret handshake schemes provide the best protection if the number of groups that are using it is large. For example, if it were to become a TLS cipher suite that was routinely used for secure discovery, then the above concerns would be alleviated.

8. Conclusion

A secret-handshake mechanism is a mechanism that would allow members of a group to authenticate each other secretly. Because members of a group often play different roles, a handshake scheme that allows members of a group to authenticate not only the fact that they belong to the same group, but also each other's roles would be very desirable.

In this paper, we proposed a secret handshake scheme that can be used by members of a group to authenticate each other, as well as the roles they play in the group. Our protocol uses Weil or Tate pairings on elliptic curves, and takes advantage of their bilinearity to compute unique shared secret keys when two members perform a handshake.

We also proposed a formal definition of secure secret handshakes, and outlined a proof that our scheme is secure under this definition.

We are implementing our protocol as a new cipher suite for TLS. Preliminary measurements show promising performance, with security parameters comparable to 1024-bit RSA yielding quite practical handshake timings.

References

[1] M. Abadi. Private authentication. In *Proceedings of the Workshop on Privacy Enhancing Technologies (PET 2002)*, San Francisco, CA, April 2002.

[2] N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Proc. International Advances in Cryptology Conference – EUROCRYPT '97*, pages 480–494, 1997.

[3] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*. Springer Verlag, August 2002.

[4] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.

[5] J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT ' 93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1994. Extended abstract.

[6] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. CRYPTO 01*, pages 213–229. Springer-Verlag, 2001. LNCS 2139.

[7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT01: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 2001.

[8] S. Brands. Restrictive binding of secret-key certificates. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology—EUROCRYPT 95*, volume 921 of *Lecture Notes in Computer Science*, pages 231–247. Springer-Verlag, 21–25 May 1995.

[9] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In *Proc. EUROCRYPT 00*, pages 93–118. Springer-Verlag, 2001. LNCS 2045.

[10] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology Crypto 2002*, pages 61–76, 2002.

[11] J. Camenisch and M. Stadler. Efficient group signatures for large groups. In *Advances in Cryptology Crypto '97*, pages 410–424, 1997.

[12] J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Proceedings of the International Workshop on Practice and Theory in Public Key Cryptography (PKC 2003)*, Miami, FL, 2003.

[13] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct 1985.

[14] D. Chaum and E. van Heijst. Group signatures. In *Proc. EUROCRYPT 91*, pages 257–265. Springer-Verlag, 1991. LNCS 547.

[15] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In Y. G. Desmedt, editor, *Proc. CRYPTO 94*, pages 257–270. Springer, 1994. Lecture Notes in Computer Science No. 839.

[16] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. IETF - Network Working Group, The Internet Society, January 1999. RFC 2246.

[17] W. Diffie and M. E. Hellman. Multiuser cryptographic techniques. In *Proc. AFIPS 1976 National Computer Conference*, pages 109–112, Montvale, N.J., 1976. AFIPS.

[18] M. J. G. Ateniese, J. Camenisch and G. Tsudik. A practical and provable secure coalition-resistant group signature scheme. In *Advances in Cryptology Crypto 2000*, pages 255–270, 2000.

[19] M. Goodrich, A. Schwerin, and R. Tamassia. An efficient dynamic and distributed cryptographic accumulator, 2000.
[20] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*. IETF - Network Working Group, The Internet Society, November 1998. RFC 2409.
[21] F. Hess. Exponent group signature schemes and efficient identity based signature schemes based on pairings. Cryptology ePrint Archive, Report 2002/012, 2002. `http://eprint.iacr.org/`.
[22] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. Maurer, editor, *Advances in Cryptology—EUROCRYPT 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 12–16 May 1996.
[23] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology Crypto '98*, pages 169–185, 1998.
[24] B. Lynn. Authenticated identity-based encryption. `http://eprint.iacr.org/2002/072`.
[25] A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas of Cryptography 1999*, pages 184–199, 1999.
[26] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Advances in Cryptology Asiacrypt 2001*, pages 552–565, 2001.
[27] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Proceedings of the Symposium on Cryptography and Information Security (SCIS 2000)*, Okinawa, Japan, January 2000.
[28] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. C. Chaum, editors, *Proc. CRYPTO 84*, pages 47–53. Springer, 1985. Lecture Notes in Computer Science No. 196.
[29] E. Verheul. Self-blindable credential certificates from the weil pairing. In *Advances in Cryptology Asiacrypt 2001*, pages 533–551, 2001.
[30] K. Zhang and R. Needham. A private matchmaking protocol. `http://citeseer.nj.nec.com/71955.html`.

A. Security Analysis

In this section we outline proof sketches for the security claims made in Section 5.2. We make use of only standard cryptographic assumptions: we work in the random oracle model [4], and assume that the Bilinear Diffie-Hellman Problem is difficult for the elliptic curves we are using. We recall the definition of the latter.

Hardness of BDH Problem: We say that *the Bilinear Diffie-Hellman Problem BDH is hard* if, for all probabilistic, polynomial-time algorithms B,

$$\mathsf{AdvBDH}_B := \Pr[B(P, aP, bP, cP) = \hat{e}(P,\ P)^{abc}]$$

is negligible in the security parameter. This probability is taken over random choice of $P \in \mathbb{G}_1$ and $a, b, c \in \{1, \ldots, q\}$ where q is the order of $\mathbb{G}_1$.

The Game "GetPair":

For a user U we denote by $G(U)$ the group such that $U \in G(U)$. The following game is played similar to the games outlined in Section 5.1 against the system outlined on Section 4.2. In particular, there is a universe of users $\mathcal{U}$ where each user U has a list of pseudonyms id_{U1}, ..., id_{Ut} and corresponding secret points $\mathsf{priv}_{U1}, \ldots, \mathsf{priv}_{Ut}$ satisfying

$$\forall U \forall i \ \ \mathsf{priv}_{Ui} = s_{G(U)} H_1(\mathsf{id}_{Ui})$$

where $s_{G(U)}$ is the group secret of $G(U)$. For simplicity, we assume t is large enough so that users do not exhaust their supply of pseudonyms (as discussed previously, pseudonym exhaustion can be dealt with by having a user contact the group authority to obtain a new list of pseudonyms.)

An adversary is placed in an environment where it is allowed to interact with users of its choice; it may corrupt a set of users $\mathcal{U}' \subset \mathcal{U}$, obtaining from each user $U \in \mathcal{U}'$ every pseudonym id_{Ui} and corresponding secret point priv_{Ui}, $i = 1, \ldots, t$.

We define the game GetPair as the following:

Step 1: The adversary A obtains interacts with arbitrary users and corrupts a set $\mathcal{U}'$ of users; for each $U \in \mathcal{U}'$, the adversary obtains all pseudonyms id_{U1}, ..., id_{Ut} and all secrets priv_{U1}, ..., priv_{Ut}.

Step 2: The adversary chooses a target user $U^* \notin \mathcal{U}'$.

Step 3: The adversary, given id_{U^*}, outputs a pair (id_A, e_0) for some $\mathsf{id}_A \neq \mathsf{id}_{Ui}$ for all $U \in \mathcal{U}' \cap G(U^*)$ and all i.

We say that A wins the game GetPair if the following equation holds:

$$e_0 = \hat{e}(H_1(\mathsf{id}_A),\ s_{G(U^*)} \cdot H_1(\mathsf{id}_{U^*})). \qquad (1)$$

We define A's advantage $\mathsf{AdvGetPair}_A$ as

$$\mathsf{AdvGetPair}_A := \Pr[A \text{ wins the game GetPair}].$$

This probability is taken over the random choices for x_i and the random coin flips of A, U^*, and all other players in the system.

We denote by Q_{H_1} the number of distinct queries A makes to the random oracle H_1. We write $e \approx 2.78$ as the base of the natural logarithm.

Lemma 7 *Suppose A is a probabilistic, polynomial time (PPT) adversary. There exists a probabilistic, polynomial time algorithm B such that*

$$\mathsf{AdvGetPair}_A \ \leq \ e\, Q_{H_1} \cdot \mathsf{AdvBDH}_B + \varepsilon,$$

where ε is negligible as a function of the security parameter.

Proof Sketch: We define B as follows. B is given an instance (P, aP, bP, cP) of the BDH problem, and wishes to use A to compute $\hat{e}(P, P)^{abc}$. The algorithm B simulates an environment in which A operates, using A's advantage in the game GetPair to help compute a solution $\hat{e}(P, P)^{abc}$ to the BDH problem.

Here is a high-level description of how the algorithm B will work. B will "cook" the responses to all queries to the random oracle H_1 so that the resulting distribution remains random, but any advantage A has in the game GetPair will be used to compute a solution to the BDH problem. B does this by using the point bP to create secret points for all pseudonyms used in the system; except the pseudonym id_A used by the adversary, whose corresponding secret will be derived from aP; and, the pseudonym id_{U^*} used in the last step, whose corresponding secret will be derived from cP.

To set up, B picks random auxiliary group secrets s'_G for all groups G.

We must specify how B will answer queries given by A. On a query $H_1(x)$, if a result has already been assigned to $H_1(x)$ it is returned again. Otherwise, B flips a random biased coin $\mathsf{guess}(x) \in \{0,1\}$ biased by some parameter δ to be determined later:

$$\mathsf{guess}(x) = \begin{cases} 0 & \text{with probability } \delta, \\ 1 & \text{with probability } 1-\delta. \end{cases}$$

The algorithm B then responds as follows:

- $\mathsf{guess}(x) = 0$:
  B picks a uniformly random $r_x \in \{1, \ldots, q\}$ and returns $H_1(x) := r_x \cdot (aP)$.
- $\mathsf{guess}(x) = 1$:
  B picks a uniformly random $r_x \in \{1, \ldots, q\}$. If x is a pseudonym of an existing user U, then B sets $H_1(x) := r_x \cdot s'_{G(U)} \cdot (bP)$, otherwise it assigns x to an uncorrupted user U and return the same.

In the first step, A obtains pseudonym/secret lists for users of its choice. For a corrupted user U and for all $i = 1, \ldots, t$, B picks a random values for the id_{Ui}, random values $r_{\mathsf{id}_{Ui}} \in \{1, \ldots, q\}$, and sets

$$H_1(\mathsf{id}_{Ui}) = r_{\mathsf{id}_{Ui}} \cdot s'_{G(U)} \cdot P, \qquad \mathsf{priv}_{Ui} = r_{\mathsf{id}_{Ui}} \cdot s'_{G(U)} \cdot (bP).$$

In stage 2, A picks a target user U^*.

In stage 3, B responds with a random value for id_{U^*} and sets $H_1(\mathsf{id}_{U^*}) := cP$. Then A outputs (id_A, e_0).

If A never queried H_1 on input id_A, then $H_1(\mathsf{id}_A)$ is assigned a random value as described above.

Suppose $\mathsf{guess}(\mathsf{id}_A) = 0$. Then $H_1(\mathsf{id}_A) = r \cdot (aP)$ for some value r known to B. B computes $w := (r s'_{G(U^*)})^{-1} \bmod q$ and returns $(e_0)^w$ as its solution to the BDH instance. It is straightforward to check that if Equation 1 holds then $(e_0)^w = e(P,P)^{abc}$ as desired.

A detailed analysis shows that if $\mathsf{guess}(\mathsf{id}_A) = 0$ and $\mathsf{guess}(x) = 1$ for all queries $x \neq \mathsf{id}_A$ to H_1, then the execution environment which B creates for A is indistinguishable from the actual game GetPair except for an error probability ε that is a negligible function of the security parameter.

It remains to optimize δ to maximize the success probability of B. We see

$$\Pr[\mathsf{guess}(\mathsf{id}_A) = 0 \text{ and } \mathsf{guess}(x) = 1 \text{ for all } x \neq \mathsf{id}_A] = \delta \cdot (1-\delta)^{Q_{H_1}}. \tag{2}$$

Using standard calculus we optimize δ to find $\delta \approx 1/Q_{H_1}$; plugging this back in to equation (2) results in $\Pr[(2)] \geq 1/(e \cdot Q_{H_1})$. The claim follows. □

We now restate the security claims for the Pairing-Based Handshake as Theorems 8 and 9, and Corollary 10. We outline proofs at the end of the appendix.

We denote by Q_{H_2} the number of distinct queries A makes to the random oracle H_2, and we denote by Q_{H_1} the number of distinct queries A makes to the random oracle H_1. We write $e \approx 2.78$ as the base of the natural logarithm.

Theorem 8 *Suppose A is a probabilistic, polynomial time adversary. There is an PPT algorithm B such that*

$$\mathsf{AdvMIG}_A \leq \Pr[\,\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G^*\,] + Q_{H_2} \cdot \mathsf{AdvGetPair}_B + \varepsilon,$$

where Q_{H_2} is the number of queries A makes to H_2, and ε is negligible in the security parameter.

Proof of Theorem 1: This follows from Theorem 8 and Lemma 7.

We now turn our attention to the Member Detection Game. Using the notation from Section 4, we claim the following.

Theorem 9 *Suppose A is a probabilistic, polynomial time adversary. There is an PPT algorithm B such that*

$$\mathsf{AdvMDG}_A \leq \Pr[\,\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G(U^*)\,] + Q_{H_2} \cdot \mathsf{AdvGetPair}_B + \varepsilon,$$

where Q_{H_2} is the number of queries A makes to H_2, and ε is negligible in the security parameter.

Proof of Theorem 4: This follows from Theorem 9 and Lemma 7.

Finally, we present the claim of eavesdropper indistinguishability for our scheme.

Corollary 10 (PBH Eavesdropper Indistinguishability) *Suppose A is a probabilistic, polynomial time adversary. There is a PPT algorithm B such that*

$$\mathsf{AdvDST}_A \leq 2 \cdot Q_{H_2} \cdot \mathsf{AdvGetPair}_B + \varepsilon,$$

where Q_{H_2} is the number of queries A makes to H_2, and ε is negligible in the security parameter.

Proof Sketch of Theorem 8: We construct an algorithm B that plays the game GetPair. It creates an environment in which A plays the Member Impersonation Game, and uses the information from A to gain an advantage in the game GetPair.

Since we model H_2 as a random oracle, B can specify how to answer queries to H_2 as long as the resulting distribution is random. If H_2 has been queried for the first time on input x, we generate a random result y, record (x,y), and return y as the result. If $H_2(x)$ has been invoked already, we find an entry (x,y) in the table and return y.

In step 1 of the Member Impersonation Game, A asks for user secrets. B passes these through to the environment in step 1 of the game GetPair.

In step 2, the adversary A picks a target group G^* and user U^*.

In step 3, A sends a message $(\mathsf{id}_A, n_A, V_1)$ to U^*. Define

$$D := e(H_1(\mathsf{id}_A), s_{G^*} \cdot H_1(\mathsf{id}_{U^*}))\|\mathsf{id}_A\|\mathsf{id}_B\|n_A\|n_B\|0.$$

A wins the Member Impersonation Game exactly when $V_1 = H_2(D)$.

Suppose $\mathsf{id}_A = \mathsf{id}_{Ui}$ for some $U \in \mathcal{U}' \cap G^*$. Let T be the transcript of A's interaction with U^*. Since id_{Ui} was uniquely assigned (with high probability), $\mathsf{PBH.TraceUser}(T)$ uses id_{Ui} to uniquely identify $U \in \mathcal{U}' \cap G^*$, and we are done.

Now suppose $\mathsf{id}_A \neq \mathsf{id}_{Ui}$ for all $U \in \mathcal{U}' \cap G^*$. Since the string D contains random nonces, with high probability $H_2(D)$ has never appeared in any of A's interactions with the system. If A never queried H_2 at any point with prefix D, we may assign a random value to $H_2(D)$, independent of A's view. Then probability that $V_1 = H_2(D)$ is negligible.

So we assume A queried $H_2(D)$ at some point during the execution of the game. B chooses a random pair (x,y) from the list of queries to H_2, pulls the prefix e_0 from x, and returns e_0 as its guess in the game GetPair. Suppose A made Q_{H_2} queries of the random oracle H_2. Then with probability $1/Q_{H_2}$, $x = D$, and B wins the game GetPair. The bound follows. □

Proof Sketch of Theorem 9:

This follows as in the proof of Theorem 8, with the following change: Instead of claiming that A must have queried H_2 at the point D in order to construct the message V_1, we claim that A must have queried H_2 at D in order to distinguish U^*'s messages from the random strings a simulator would send. The rest of the analysis is similar. □

Proof Sketch of Corollary 10:

Suppose A is a PPT adversary with nonzero distinguishing advantage. We build an algorithm A' to play the Member Detection Game.

We begin by starting the adversary A. Any request from A to interact with users or obtain secrets from users is passed through by A', which is in Step 1 of the Member Detection Game. When A picks users U_1^* and U_2^*, A' picks $i \in \{1,2\}$ and requests $U^* := U_i^*$ as the target user. It then acts as a "man-in-the-middle" for the interaction between U_1^* and U_2^*, and sends the resulting transcript to A. By a standard hybrid argument, a distinguishing advantage AdvDST_A for A translates to $\mathsf{AdvMDG}_{A'} = (1/2) \cdot \mathsf{AdvDST}_A$.

Without loss of generality, we will assume $i = 2$. Notice that the message $\mathsf{id}_{A'}$ sent by A' to U_2^* satisfies $\mathsf{id}_{A'} = \mathsf{id}_{U_1^*}$. By the restriction on A, $U_1^* \notin \mathcal{U}'$. So if T is the transcript of the interaction of A' with U_2^*, we know $\mathsf{PBH.TraceUser}(T) = U_1^*$, implying

$$\Pr[\mathsf{PBH.TraceUser}(T) \in \mathcal{U}' \cap G(U^*)] = 0.$$

The bound follows from Theorem 4. □