CERIAS Tech Report 2007-24

SECRET HANDSHAKES WITH DYNAMIC AND FUZZY MATCHING

by Giuseppe Ateniese, Marina Blanton, and Jonathan Kirsch

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Secret Handshakes with Dynamic and Fuzzy Matching

Giuseppe Ateniese*
Department of Computer Science
The Johns Hopkins University
ateniese@cs.jhu.edu

Marina Blanton†
Department of Computer Science
Purdue University
mbykova@cs.purdue.edu

Jonathan Kirsch
Department of Computer Science
The Johns Hopkins University
jak@cs.jhu.edu

Abstract

The need for communication privacy over public networks is of growing concern in today's society. As a result, privacy-preserving authentication and key exchange protocols have become critical primitives in building secure distributed systems. Secret handshakes provide such a service by allowing two members of the same group to secretly and privately authenticate to each other and agree on a shared key for further communication.

This paper presents the first efficient secret handshake schemes with unlinkable, reusable credentials that do not rely on random oracles for their security (solving open problems from prior literature). In previous work, secret handshakes were extended with roles, so that a group member A can specify the role another group member B must have in order to successfully complete the protocol with A. We generalize the traditional and role-based secret handshake in two ways. First, we present a secret handshake with dynamic matching*, in which each party can specify* both *the group and the role the other must have in order to complete the handshake. Second, we provide a novel extension of secret handshakes to include attributes, allowing the handshake to be based on approximate (or* fuzzy*) matching.*

We demonstrate the practicality and efficiency of our protocols by evaluating a prototype implementation. We integrate our dynamic matching protocol into IPsec, and we detail the performance tradeoffs associated with our fuzzy matching scheme. Our experiments indicate that our solutions offer attractive performance.

1 Introduction

A secret handshake scheme, introduced by Balfanz et al. [5], allows two members of the same group to secretly authenticate to each other and agree on a shared key for further communication. Such authentication is privacy preserving, meaning that if the participants belong to the same group, they only learn that they are members of that group (without learning each other's identities), and learn nothing about each other otherwise. The most commonly used example of such interaction is the mutual authentication of CIA agents. That is, consider a CIA agent who wants to authenticate to another agent but does not want to reveal his credentials to anyone other than CIA agents. Obviously, two CIA agents should be able to successfully complete the handshake, and other parties should not be able to perform or recognize the handshake. Such schemes can also be used by members of secret societies to identify other members, by the military to discover and use a secret service, etc.

Another important application of secret handshakes that has not been considered in prior literature is handshakes for High-bandwidth Digital Content Protection (HDCP) systems, designed to protect video data from unauthorized copying. In the HDCP protocol, two devices engage in an identity-based authentication protocol to agree on a key that is subsequently used to encrypt the data transmitted on the DVI bus. The protocol used in HDCP was a custom solution, designed to meet efficiency requirements; however, it was shown in [15] to be insecure. An efficient, provably secure handshake protocol could help to fill this gap, as only devices with legitimate credentials would be able to authenticate to each other and agree on a key.

Another domain where secret handshakes proved to be useful is anonymous routing in ad-hoc networks. A recent publication by Li and Ephremides [29] shows that direct

usage of secret handshakes for anonymous routing outperforms all existing solutions. Earlier work in that direction [47, 46] is also reminiscent of (and was inspired by) secret handshake schemes.

An important extension of secret handshakes is to include roles: users can require that group members' affiliations are revealed only to members who hold specific roles in the group. For example, a vehicle operator Alice might want to authenticate herself to Bob only if Bob can authenticate as a policeman. In this case, Alice specifies what role Bob must have in order for the handshake to succeed, and Bob specifies what role Alice must have. In this work, we take the flexibility of secret handshakes with roles to a new level: in addition to being flexible in specifying user roles, members can now specify the group of the person with whom they would like to perform a secret handshake. This *dynamic matching* (rather than a static group setting) is what distinguishes our protocols from prior work. This new model will allow for successful handshakes between, for instance, members of sister societies, instead of just members of the same society.

Furthermore, we extend the framework of secret handshakes to support attributes and provide approximate (or *fuzzy*) attribute-based matching. That is, now each member has a number of attributes (call it n) associated with her membership, such as the group, role, seniority, possibly an alternative group, etc. At the time of a handshake Alice specifies what attributes Bob must have, and the handshake succeeds if Bob's credentials match d or more of the attributes specified by Alice for some threshold $d \leq n$. The same applies to Bob who specifies attributes for Alice. Such an extension adds a lot of flexibility and power to the authentication rules of secret handshakes. For instance, now Alice can require Bob to be a CIA agent and have either top secret or secret clearance level. We refer to our new notion of secret handshakes (in both of the above settings) as *unrestricted* secret handshakes.

Perhaps the most appealing application of secret handshakes with the new extended capabilities, which already can be used today, is social networks such as online dating. That is, consider Alice who has a set preferences that her potential partner must satisfy. The preferences she has are private, and she does not want to reveal them to others. Similarly, Bob has a set of requirements that his partner must match. Attribute-based secret handshakes then naturally allow them to check whether each of them meets the expectations of the other without revealing any additional information and, if so, exchange their contact information using the shared key.

Many existing authentication and signature schemes fall short of solving this seemingly simple authentication problem. That is why secret handshakes received a fair amount of attention in the literature (see, e.g., [5, 44, 13, 40, 39, 42, 48]). Despite this, an efficient secret handshake scheme with unlinkable reusable credentials secure in the standard model remained to be an open problem. In this paper, we show that solutions to secret handshakes exist that combine efficiency and unlinkability and do not rely on random oracles in their security, even in our new flexible models, thus closing this gap. Our protocols are built using an Identity-Based Encryption (IBE) and are the first of their kind. Our solution to secret handshakes with fuzzy matching also uses the ideas underlying the construction of fuzzy IBE [36].

To demonstrate the practicality and efficiency of our protocols, we provide a prototype implementation. It consists of (i) integrating secret handshakes with dynamic matching into the key management functionality of IPsec, which resulted in only a small overhead, and (ii) evaluating an implementation of our fuzzy handshake scheme, which resulted in very reasonable performance.

To summarize, our contributions consist of the following: We extend secret handshakes to permit dynamic matching and present an IBE-based solution to the problem. We also introduce attributes into secret handshakes and extend the model to permit fuzzy matching, which significantly enriches the set of policies that secret handshakes can support. We provide solutions to both types of unrestricted secret handshakes, which are the first schemes that are simultaneously (a) efficient, (b) use unlinkable reusable credentials, and (c) secure in the standard model. Our experimental results indicate that both of our solutions perform well in practice.

The rest of this paper is organized as follows: in Section 2 we give an overview of secret handshake literature. Section 3 defines a secret handshake scheme and its security, and Section 4 provides background information. Section 5 gives our construction for secret handshakes with dynamic matching, and in Section 6 we show how to build secret handshakes with approximate matching. In Section 7 we comment on deployment issues, and Section 8 reports on implementation results. Lastly, Section 9 concludes the paper.

2   Related Work

In this section we review only existing literature on secret handshakes and other closely related constructions. For other anonymity tools that cannot adequately address the problem of secret handshakes see, for instance, [5, 44].

The first solution to the problem of secret handshakes is due to Balfanz et al. [5]. Their scheme is simple and efficient but requires single-use pseudonyms to achieve unlinkability, which means that each user must store a large number of credentials. The solution supports authentication of members with roles, and the scheme is proven to be secure in the random oracle model.

Castelluccia, Jarecki, and Tsudik [13] addressed the problem of secret handshakes through the use of so-called CA-oblivious encryption. This solution is slightly more efficient, but it still does not support reusable credentials. It is secure in the random oracle model.

The work by Xu and Yung [44] permits the use of multi-show credentials, and the security of their construction does not rely on random oracles, but they achieve only a limited notion of anonymity, namely, k-anonymity. Unlinkability is achieved by allowing each user to authenticate as one of k members by selecting $k-1$ public keys of group members, resulting in $O(k)$ computation (the expensive computation is, however, only $O(1)$). Members can reuse their credentials because they always authenticate as someone out of k users.

Tsudik and Xu [40, 39] extend the notion of secret handshakes to a multi-party setting. Their work combines three building blocks (a group signature scheme, a centralized group key distribution scheme, and distributed group key agreement) to create a framework for multi-user handshakes. Unlinkability is achieved by using group signatures, but this solution is not very efficient. Jarecki, Kim, and Tsudik [23] also provide a solution to multi-party handshakes by integrating previous work on secret handshakes [13] with a group key agreement protocol. As in [13], one-time certificates are used.

Finally, Vergnaud [42] constructs secret handshakes using RSA; and Zhou, Susilo, and Mu [48] do so by using ElGamal and DSA. Both papers rely on random oracles in their security.

The notions of Oblivious Signature-Based Envelope (OSBE) [28, 33] and Hidden Credentials [21, 12] are also related to secret handshakes. The authors of [33] show how to construct basic secret handshakes by using OSBE in both directions. Also, work of [20, 22] explores the relationship between CA-oblivious encryption (introduced in [13]), hidden credentials, OSBE, and secret handshakes.

3 Model and Definitions

A secret-handshake scheme SHS consists of the following algorithms:

- Setup is a trusted algorithm that, given a security parameter 1^κ, outputs public parameters params common to all subsequently generated groups.

- CreateGroup is a key generation algorithm run by a group administrator GA which, given params, outputs the group's public information G and group's secret s_G.

- AddMember is a protocol between the GA and a user, which takes GA's secret s_G and public parameters G

and params as input and results in the user becoming a member of the group G with credentials cred.

- Handshake is the authentication protocol executed between players A and B on public input params and private input of A cred_A and private input of B cred_B. At the end of the protocol, if A's requirements for B are matched by B's credentials and B's requirements for A are matched by A's credentials, A and B authenticate by sharing a common key. Such authentication fails otherwise.

In the original setting, the handshake protocol results in acceptance if A and B are members of the same group, but it can be extended with roles and other attributes.

Also, the definition of a secret handshake scheme in certain prior publications includes another algorithm, TraceUser, run by the system administrator. That is, if the scheme supports tracing, then given a transcript T of interaction of user U with one or more users, this algorithm outputs the identity of a user whose keys were used by U during the interaction. We, however, do not consider traceability in this work.

Our secret handshake schemes must provide the following core security properties:

Correctness: honest members satisfying the handshake rules (e.g., belonging to the requested group) will always successfully complete the handshake.

Impersonator resistance: an adversary not satisfying the rules of the handshake protocol is unable to successfully authenticate to an honest member.

Detector resistance: an adversary not satisfying the rules of the handshake protocol cannot decide whether some honest party satisfies the rules or not.

Unlinkability: it is not feasible to tell whether two executions of the handshake protocol were performed by the same party or not, even if both of them were successful.

4 Background and Building Blocks

4.1 Preliminaries

In this section, we describe notation used in the rest of this paper and list number-theoretic preliminaries and cryptographic assumptions. A function $\epsilon(\kappa)$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ, $\epsilon(\kappa) < \frac{1}{p(\kappa)}$. The notation $G = \langle g \rangle$ means that g generates the group G. Our solutions use groups with pairings, and we review concepts underlying such groups next.

Definition 1 (Bilinear map) *A one-way function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map if the following conditions hold:*

- *(Efficient) $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are groups of the same prime order p, and there exists an efficient algorithm for computing e.*

- *(Bilinear) For all $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g^a, \tilde{g}^b) = e(g, \tilde{g})^{ab}$.*

- *(Non-degenerate) If g generates $\mathbb{G}_1$ and $\tilde{g}$ generates $\mathbb{G}_2$, then $e(g, \tilde{g})$ generates $\mathbb{G}_T$.*

Throughout this work, we assume that there is a trusted setup algorithm Set that, on input a security parameter 1^κ, outputs the setup for groups $\mathbb{G}_1 = \langle g \rangle$ and $\mathbb{G}_2 = \langle \tilde{g} \rangle$ of prime order p that have a bilinear map e, and $e(g, \tilde{g})$ generates $\mathbb{G}_T$ (which also has order p). That is, $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e) \leftarrow Set(1^\kappa)$.

Our schemes are built using subgroups of elliptic curves with pairings where the decisional Diffie-Hellman (DDH) problem is hard. This setting has been recently used in many publications ([38, 9, 6, 3, 4] and others), and, according to [18], is the most efficient and versatile type of pairings. The use of DDH-hard pairing groups requires the Symmetric External Diffie-Hellman (SXDH) assumption [6], and we also rely on the standard Bilinear Diffie-Hellman (BDH) assumption in asymmetric bilinear groups (first introduced in [10]), both of which are given next.

Definition 2 (SXDH assumption) *We say that the SXDH assumption holds if, given values y, y_1, y_2, $y_3 \in \mathbb{G}_1$, it is not computationally feasible to decide if there is an integer $a \in \mathbb{Z}_p$ such that $y_1 = y^a$ and $y_3 = y_2^a$, i.e., $\mathbb{G}_1$ is a DDH-hard group. The same requirement must hold for $\mathbb{G}_2$, i.e., it is also a DDH-hard group.*

Definition 3 (BDH assumption) *Let g be a generator of $\mathbb{G}_1$ and h be a generator of $\mathbb{G}_2$. We say that the BDH assumption holds if, given $g, g^a, g^b, g^c \in \mathbb{G}_1$ and $\tilde{g}, \tilde{g}^a, \tilde{g}^b \in \mathbb{G}_2$ for random $a, b, c \in \mathbb{Z}_p$, it is not possible to compute $e(g, \tilde{g})^{abc}$ with a non-negligible probability.*

4.2 Identity-based encryption

Our secret handshake scheme with dynamic matching is built upon the Identity-Based Encryption (IBE) scheme. The first practical IBE scheme is due to Boneh and Franklin [10], but the scheme we use in this work was introduced by Waters [43] and is an improved version of the Boneh-Boyen scheme [8]. An overview of the scheme is given in Appendix A.

Fuzzy IBE [36] is a new type of identity-based encryption where instead of using strings to represent identities,

identities are viewed as a set of descriptive attributes. A user with the secret key for the identity w is able to decrypt a ciphertext encrypted with the public key w' iff w and w' are within a certain distance of each other (according to some metrics). This introduces interesting applications such as usage of biometric identities and attribute-based encryption (which can supersede hierarchical IBE); see [36] for more information. As the approach that we use to achieve secret handshakes with approximate matching borrows techniques from this work, we review the construction of [36] in Appendix A.

4.3 Privacy-preserving set operations

Our fuzzy handshake scheme requires computing set intersection over private datasets, and here we first briefly review prior literature on private set intersection and then give an overview of the protocol that we chose to use.

Early solutions to computing secure set operations involved secure multiparty computation introduced by Yao [45], but have a high communication complexity. Consequently, the problem of set intersection and its variants have been considered in many recent publications [14, 2, 17, 26, 41, 27, 30]. Agrawal, Evfimievski, and Srikant [2] propose a more efficient solution to the two-party set intersection problem in the semi-honest adversarial setting, using a commutative encryption function as a building block. Freedman, Nissim, and Pinkas [17] address problems related to two-party set intersection, or private matching, in both the semi-honest and malicious settings, where the datasets are represented as the roots of a polynomial. More recently, Kissner and Song [27] provide a generic multiparty framework in which to securely and privately compute operations over multisets, including union, intersection, and element reduction. Kiayias and Mitrofanova [26] address the problem of two-party set disjointness on private datasets, i.e., computing whether their intersection is empty or not.

In this paper we are interested in two problems: computing the set intersection of two private datasets and computing whether the cardinality of the intersection of two sets is above a certain threshold. Among the solutions available in the literature, we chose the set intersection protocol of Freedman et al. [17], since it does not require the use of random oracles and is relatively simple. Note that this protocol is secure in the *semi-honest model* (i.e., the players follow the protocol as prescribed but might try to learn additional information about the other party's data by using intermediate results of the computation). In case of secret handshakes, we assume that it is in the best interest of the players to authenticate (and deviating from the prescribed behavior might prevent this), and the players follow the set intersection protocol correctly. If security beyond the semi-

honest behavior is needed, other solutions from the existing literature can be used with our scheme instead.

Let player Alice with dataset $X = \{x_1, x_2, \ldots, x_k\}$ and player Bob with dataset $Y = \{y_1, y_2, \ldots, y_k\}$ participate in the set intersection protocol of [17]. Alice sets up a semantically-secure homomorphic encryption scheme and publishes the public parameters. She constructs a polynomial of degree k with roots $\{x_1, \ldots, x_k\}$ and sends Bob encryptions of the coefficients, $\{Enc(\alpha_0), Enc(\alpha_1), \ldots, Enc(\alpha_k)\}$. Bob uses the homomorphic properties of the encryption scheme to evaluate Alice's polynomial at each point y in his dataset, since $Enc(P(y)) = Enc(\sum_{u=0}^{k} \alpha_u y^u)$. He can also compute $Enc(rP(y) + y)$, with r being chosen at random for each y. If y is a root of the polynomial (i.e., it matches one of Alice's elements), then the ciphertext will equal $Enc(y)$. When Alice decrypts the ciphertexts, she therefore computes the intersection as any $x_i \in X$ for which there is a corresponding decrypted value. Note that the protocol is asymmetric, in that only Alice obtains the result. This is exactly what is needed in our handshake scheme with fuzzy matching.

When Alice and Bob engage in our handshake protocol, they will also need to compute the function $(|X_1 \cap Y_1| \geq d) \wedge (|X_2 \cap Y_2| \geq d)$ for some fixed threshold d, where X_1 and X_2 are known to Alice, Y_1 and Y_2 are known to Bob, and $|X|$ denotes the size of set X. None of the previous solutions (other than circuit evaluation) allow us to compute this function directly, and the general circuit evaluation results are too inefficient for this problem. We therefore use the cardinality threshold matching protocol of [17], which combines a modified set intersection protocol with boolean circuit evaluation. In this modified version, Bob encodes random values r_y instead of his true y to obtain $Enc(rP(y) + r_y)$, Alice enters $rP(y) + r_y$, and Bob enters r_y into the circuit. The circuit then computes whether these values matched, counts the number of matches, compares that number to the threshold, and outputs a bit. Alice and Bob run the modified protocol on inputs X_1 and Y_1 and inputs X_2 and Y_2 as described above, the circuit then computes the AND of the bits, and the result of the computation is sent to both of them. Boolean circuits of small size are rather efficient and they are one of the best approaches available for performing comparison.

5   Secret Handshakes with Dynamic Matching

Dynamic matching means that Alice can specify what group and role Bob must have in order for the handshake to succeed, and similarly Bob can specify the group and role that Alice must have. As mentioned earlier, this extension allows for more flexible and user-chosen authentication rules.

5.1   The scheme

We modify the IBE scheme (described in Appendix A.1) to achieve key privacy through the use of asymmetric DDH-hard groups. By key privacy we mean the inability of an adversary to determine the identities of the protocol participants. The use of DDH-hard groups to achieve key privacy for Waters IBE scheme was mentioned in [6] and [11]. Some of the ideas used in [6] in the context of storage systems are used in our scheme as well. The main difference between our solution and Waters scheme is that, in order to make it key-private, we force the messages transmitted during the protocol to be in the same group (using Waters scheme in the asymmetric setting leads to having values in both $\mathbb{G}_1$ and $\mathbb{G}_2$, disallowing the scheme to be key-private).

In what follows, $||$ denotes concatenation of two strings. We also assume that all identities are n-bit strings (i.e., identities shorter than n bits are padded to the right length).

Setup. Given a security parameter 1^κ, run $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e) \leftarrow Set(1^\kappa)$. Choose $h \overset{R}{\leftarrow} \mathbb{G}_2$ and $\alpha \overset{R}{\leftarrow} \mathbb{Z}_p$ and set $g_\alpha = g^\alpha$. Then choose $n + 1$ random values $u', u_1, \ldots, u_n \overset{R}{\leftarrow} \mathbb{Z}_p$ and set $G' = g^{u'}$, $G_1 = g^{u_1}$, …, $G_n = g^{u_n}$, $H' = \tilde{g}^{u'}$, $H_1 = \tilde{g}^{u_1}$, …, $H_n = \tilde{g}^{u_n}$. The public parameters are params $= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, G', G_1, \ldots, G_n)$.

CreateGroup. There is no computation associated with creating a new group other than selecting a name for the group to which we refer to as groupID. The GA must know $h_\alpha = h^\alpha$, and the values $\tilde{g}$, H', $H_1, \ldots, H_n$ to be able to issue group member credentials.

AddMember. Adding a new member with role r to group groupID consists of issuing to that member a private key corresponding to the identity groupID$||r$. Let $\text{rep}_1(v) \in \mathbb{G}_1$ denote representation of an n-bit string v in $\mathbb{G}_1$ such that $\text{rep}_1(v) = G' \prod_{i \in V} G_i$, where $V \subseteq \{1, \ldots, n\}$ is the set of indices i for which the ith bit of v is equal to 1. Similarly, let $\text{rep}_2(v) = H' \prod_{i \in V} H_i$ denote the representation of string v in $\mathbb{G}_2$.[1] Then the private key of the member of groupID with role r is created as cred $= (d_1, d_2) = (\tilde{g}^s, h_\alpha(\text{rep}_2(\text{groupID}||r))^s)$, where $s \overset{R}{\leftarrow} \mathbb{Z}_p$, and $d_1, d_2 \in \mathbb{G}_2$.

Handshake. Suppose Alice with a secret $\text{cred}_A = (d_1^A, d_2^A)$, which is a private key on the identity groupID$_A||r_A$, and Bob with a secret $\text{cred}_B = (d_1^B, d_2^B)$, which is a private key on the identity groupID$_B||r_B$, engage in a handshake

---

[1] Note that $e(g, \text{rep}_2(v)) = e(\text{rep}_1(v), \tilde{g})$.

protocol. They should successfully complete the protocol if the group and the role specified by Alice for Bob matches Bob's credentials and the group and the role specified by Bob for Alice matches Alice's credentials. The protocol proceeds as follows:

1. A chooses $x \xleftarrow{R} \mathbb{Z}_p$ and sends g^x and $(\mathsf{rep}_1(\mathsf{groupID}'_B || r'_B))^x$ to B, where r'_B ($\mathsf{groupID}'_B$) is the role (resp., group) that B must have in order to complete the handshake.

2. Similarly, B chooses $y \xleftarrow{R} \mathbb{Z}_p$ and sends g^y and $(\mathsf{rep}_1(\mathsf{groupID}'_A || r'_A))^y$ to A, where r'_A ($\mathsf{groupID}'_A$) is the role (resp., group) that A must have in order to complete the handshake.

3. Using her knowledge of x and what she just received from B, A computes the following keys:

$$k_1 = e(g_\alpha, h)^x \quad \text{and}$$
$$k_2 = \frac{e(g^y, d_2^A)}{e((\mathsf{rep}_1(\mathsf{groupID}'_A || r'_A))^y, d_1^A)}.$$

4. Using y and what he just received from A, B computes the keys:

$$k_1 = \frac{e(g^x, d_2^B)}{e((\mathsf{rep}_1(\mathsf{groupID}'_B || r'_B))^x, d_1^B)}$$
$$\text{and} \quad k_2 = e(g_\alpha, h)^y.$$

If $\mathsf{groupID}_A = \mathsf{groupID}'_A$, $\mathsf{groupID}'_B = \mathsf{groupID}_B$, $r_A = r'_A$, and $r_B = r'_B$, then at the end of the handshake both A and B share the key $k = (k_1, k_2)$, where $k_1 = e(g_\alpha, h)^x$ and $k_2 = e(g_\alpha, h)^y$. That is, for Alice we have:

$$
\begin{aligned}
k_2 &= \frac{e(g^y, d_2^A)}{e((\mathsf{rep}_1(\mathsf{groupID}'_A || r'_A))^y, d_1^A)} \\
&= \frac{e(g^y, h^\alpha(\mathsf{rep}_2(\mathsf{groupID}_A || r_A))^s)}{e((\mathsf{rep}_1(\mathsf{groupID}_A || r_A))^y, \tilde{g}^s)} \\
&= \frac{e(g, h)^{y\alpha} e(g, \mathsf{rep}_2(\mathsf{groupID}_A || r_A))^{sy}}{e(\mathsf{rep}_1(\mathsf{groupID}_A || r_A), \tilde{g})^{sy}} \\
&= e(g_\alpha, h)^y.
\end{aligned}
$$

Similarly, for Bob $k_1 = e(g_\alpha, h)^x$ if the groups and roles matched.

We also would like to note that it might be possible to prove our scheme to be a key-private IBE scheme[2]. In addition, other key-private IBE schemes can be used to construct such a handshake scheme. In particular, a recent anonymous IBE scheme of Boyen and Waters [11] is a good alternative to our approach.

---

[2]As in other IBE schemes, a message M can be encrypted by sending $e(g_\alpha, h)^x M$ in addition to g^x and $(\mathsf{rep}_1(\mathsf{recipientID}))^x$.

5.2 Security

To prove the security of the above scheme, we need to show that all of the required security properties listed in Section 3 hold. We first define each of them in more detail. Our definitions largely follow the definitions of the original secret handshakes paper [5].

For an adversary $\mathcal{A}$ we define a *member impersonation game*, during which $\mathcal{A}$ is allowed to corrupt users of her choice, then selects a target group G_t and target role R_t, and tries to impersonate a member of G_t with role R_t during a handshake protocol with an honest user. $\mathcal{A}$ wins if she is successful in impersonating, when she has never corrupted any member of G_t with role R_t. Then we say that a secret handshake scheme is *impersonator resistant* if any polynomial-time adversary $\mathcal{A}$ can win the member impersonation game with at most negligible probability. A more detailed and formal description of this (as well as other) security games and more precise security definitions are provided in Appendix B.

Now consider a *member detection game*, in which $\mathcal{A}$ can corrupt users of her choice, then chooses a target user U_t (having a specific role R_t in group G_t). Intuitively, $\mathcal{A}$ cannot detect members if her interaction with them yields no new information. Thus, $\mathcal{A}$ is asked to engage in a handshake protocol with either U_t or a random simulator and must decide with which entity she is interacting. We say that a secret handshake scheme is *detection resistant* if any $\mathcal{A}$ who never corrupted any member of G_t with role R_t has probability of winning the member impersonation game at most negligibly larger than $1/2$.

Finally, the unlinkability property requires that $\mathcal{A}$ is unable to tell whether two executions of the protocol correspond to the same user or not. Thus, in the *linking game* $\mathcal{A}$ corrupts users, chooses a target user U_t, and then is asked to engage in a secret handshake with either U_t or another user with similar credentials. $\mathcal{A}$ wins if she is able to correctly guess with whom she was interacting during the challenge protocol. Then we say that a secret handshake scheme is *unlinkable* if any $\mathcal{A}$ who never corrupted the users she is asked to interact with wins the linking game with the probability at most negligibly larger than $1/2$. As mentioned above, more precise definitions can be found in Appendix B.

Now we are ready to state the security of our scheme. The proof of this and other theorems in this paper can be found in Appendix B.

Theorem 1 *The above scheme is a secure secret handshake scheme with dynamic matching assuming that the BDH and SXDH assumptions hold.*

6 Secret Handshakes with Fuzzy Matching

In this section we extend the notion of secret handshakes to support approximate attribute-based matching and present a scheme that achieves such handshakes. In what follows, let each member have credentials consisting of n descriptive attributes. At the time of a handshake Alice specifies an n-element set of attributes for Bob, and Bob specifies an n-element set of attributes for Alice. The handshake protocol succeeds if Bob's credentials matched at least d of the attributes specified by Alice, and Alice's credentials matched at least d attributes specified by Bob, for a fixed $d \leq n$.

6.1 The scheme

Our secret handshake scheme with fuzzy matching is built on the fuzzy IBE scheme [36]. As before, we modify the setting to DDH-hard groups and introduce other modifications to achieve secrecy. In particular, we make sure that all messages transmitted during the protocol are in the same group, and this is what allows us to achieve key privacy.

Setup. Given a security parameter 1^κ, run $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \tilde{g}, e) \leftarrow Set(1^\kappa)$. Choose $h \stackrel{R}{\leftarrow} \mathbb{G}_2$ and $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$, and set $g_\alpha = g^\alpha$. Then choose n, d, and $t_1, \ldots, t_{n+1} \stackrel{R}{\leftarrow} \mathbb{Z}_p$, and set $G_1 = g^{t_1}, \ldots, G_{n+1} = g^{t_{n+1}}, H_1 = \tilde{g}^{t_1}, \ldots, H_{n+1} = \tilde{g}^{t_{n+1}}$. The public parameters are params $= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, n, d, G_1, \ldots, G_{n+1})$. Also, the functions $T_1(x) = g^{x^n} \prod_{i=1}^{n+1} G_i^{L_{i,N}(x)}$ and $T_2(x) = \tilde{g}^{x^n} \prod_{i=1}^{n+1} H_i^{L_{i,N}(x)}$, where $L_{i,S} = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$ is the Lagrange coefficient for i and set S and $N = \{1, \ldots, n+1\}$, are public.

CreateGroup. There is no computation explicit to creation of a group or a set of attributes. The GA must know α, $\tilde{g}$, and $H_1, \ldots, H_{n+1}$ to be able to issue member credentials.

AddMember. Membership credentials for a user with attributes $u = (u_1, \ldots, u_n)$, where each $u_i \in \mathbb{Z}_p^*$, are issued similar to the way they are issued in the fuzzy IBE scheme described in Section A.2. The GA chooses at random a $d-1$ degree polynomial q such that $q(0) = \alpha$. The user credentials are cred $= (\{D_{u_i}\}_{u_i \in u}, \{d_{u_i}\}_{u_i \in u})$, where $D_{u_i} = h^{q(u_i)} T_2(u_i)^{r_i}$, $d_{u_i} = \tilde{g}^{r_i}$, and $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ for all $1 \leq i \leq n$.

Handshake. As before, suppose member A with credentials cred$_A$ $= (\{D_{u_i}^A\}_{u_i \in u_A}, \{d_{u_i}^A\}_{u_i \in u_A})$ for attributes u_A and member B with credentials cred$_B$ $= (\{D_{u_i}^B\}_{u_i \in u_B}, \{d_{u_i}^B\}_{u_i \in u_B})$ for attributes u_B engage in a handshake protocol. Their interaction then proceeds as follows:

1. A prepares a set of attributes u'_B that B must match and B prepares a set of attributes u'_A that A must match. They execute a secure protocol for function $f = (|u_A \cap u'_A| \geq d) \wedge (|u_B \cap u'_B| \geq d)$ as described in Section 4.3. At the end of the protocol either both of them learn 0, in which case they abort the handshake protocol, or both of them learn 1, in which case they continue.

2. A and B execute secure protocols for set intersection $u_A \cap u'_A$ and $u_B \cap u'_B$, at the end of which A learns $w_A = u_A \cap u'_A$ and B learns $w_B = u_B \cap u'_B$ (see Section 4.3 for more detail).

3. A chooses $x \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sends to B g^x and $\{T_1(u_i)^x\}_{u_i \in u'_B}$. Similarly, B chooses $y \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and sends to A g^y and $\{T_1(u_i)^y\}_{u_i \in u'_A}$.

4. A computes $k_1 = e(g_\alpha, h)^x$ and chooses an arbitrary d-element subset S of w_A. A then computes $k_2 = \prod_{u_i \in S} \left(\frac{e(g^y, D_{u_i}^A)}{e(T_1(u_i)^y, d_{u_i}^A)} \right)^{L_{u_i,S}(0)}$ and sets the shared key to $k = (k_1, k_2)$.

5. Similarly, B chooses a d-element subset S of w_B and computes $k_1 = \prod_{u_i \in S} \left(\frac{e(g^x, D_{u_i}^B)}{e(T_1(u_i)^x, d_{u_i}^B)} \right)^{L_{u_i,S}(0)}$. Then B computes $k_2 = (g_\alpha, h)^y$ and sets $k = (k_1, k_2)$.

To see that A's k_2 is in fact the same as B's (i.e., $k_2 = e(g_\alpha, h)^y$), we derive the key as follows:

$$
\begin{aligned}
k_2 &= \prod_{u_i \in S} \left(\frac{e(g^y, D_{u_i}^A)}{e(T_1(u_i)^y, d_{u_i}^A)} \right)^{L_{u_i,S}(0)} \\
&= \prod_{u_i \in S} \left(\frac{e(g^y, h^{q_A(u_i)} T_2(u_i)^{r_i^A})}{e(T_1(u_i)^y, \tilde{g}^{r_i^A})} \right)^{L_{u_i,S}(0)} \\
&= \prod_{u_i \in S} \left(\frac{e(g^y, h^{q_A(u_i)}) e(g^y, T_2(u_i)^{r_i^A})}{e(T_1(u_i)^{r_i^A}, \tilde{g}^y)} \right)^{L_{u_i,S}(0)} \\
&= \prod_{u_i \in S} e(g, h)^{q_A(u_i) y L_{u_i,S}(0)} \\
&= e(g, h)^{\alpha y} = e(g_\alpha, h)^y
\end{aligned}
$$

The transition from the third line to the fourth is due to the fact that $e(g, T_2(x)) = e(T_1(x), \tilde{g})$. The same derivation can be performed for B's k_1, which will result in $k_1 = e(g_\alpha, h)^x$.

Note that this protocol prescribes the users to stop after Step 1 if their credentials did not match. This gives outside observers information about whether the handshake protocol succeeded or not, thus violating the indistinguishability to eavesdroppers property. To mitigate this problem, we

advise the users to proceed with the protocol, even if they receive 0 at the end of Step 1. In this case, however, they use randomly generated values to finish the protocol instead of using their true credentials.

6.2 Security

In secret handshakes with attribute-based matching, the notion of the group is replaced with attributes. Thus, now the *member impersonation game* for an adversary $\mathcal{A}$ consists of corrupting users, selecting a target user U_t, and declaring credentials u, an owner of which she would like to impersonate. User U_t must have at least d attributes in common with u and no user controlled by $\mathcal{A}$ can have d or more attributes in common with u. The same modifications apply to the *member detection game* as well.

Because we no longer have groups, for the property of detector resistance we require that an adversary $\mathcal{A}$ who does not have the required d overlapping attributes with the target user does not learn any information about U_t's attributes during a protocol execution. Note, however, that this does not imply perfect hiding of information about attributes a user has, and will be true of any protocol with threshold-based matching. In particular, the threshold-based nature of the protocol allows for probing of user attributes. In our current protocol, a malicious player A can try different values for u_A and u'_B in the attempt to learn as much information as possible about B's credentials from Step 1 of the protocol. One possibility for solving this problem is to ensure that A uses her true credentials u_A during Step 1 of the protocol by, for instance, requiring them to be signed by the GA. This would not completely mitigate the probing attacks, but will significantly constrain the attacker in its capabilities. As the techniques that could be used for such binding of credentials are inefficient, they were not implemented in this work, and we leave a more detailed investigation of this problem as a direction for future work.

The security of our solution holds in the so-called Fuzzy Selective-ID model [36]. Informally, this means that the adversary commits to the identity it would like to impersonate prior to system setup. A more detailed description of this setting is given in Appendix B.

Theorem 2 *The above scheme is a secure fuzzy secret handshake scheme in the Fuzzy Selective-ID model assuming that the BDH and SXDH assumptions hold.*

7 Deployment Issues

As remarked by Castelluccia, Jarecki, and Tsudik [13], real-world deployments of secret handshake protocols require strengthened security notions. This is in order to prevent man-in-the-middle attacks or any other active attacks that affect traditional key agreement protocols. As suggested in [13], a good strategy would consist in adopting well-established techniques devised for Authenticated Key Agreement (AKE) protocols. Signatures, for instance, are usually employed to prevent active attacks. However, within the secret handshake framework, credentials are secret, so it is not possible to generate non-repudiable signatures since there is nothing to verify them against. A possible solution is to create signatures on handshake transactions using public credentials, or "double identities." That is, each user has two types of credentials, one secret (e.g., CIA agent) for secret handshake and one public (e.g., teacher) for digital signatures. We elaborate more on this in Section 8.1.1.

Related to this, it should be noted that secret handshake protocols do not provide certain security guarantees when plainly deployed within a networked environment. For instance, an adversary that monitors network traffic would be able to learn which nodes in the network had a successful secret handshake (even if the traffic is authenticated and encrypted) by measuring the amount of traffic between nodes immediately after the handshake. Even though the adversary will not learn nodes' group affiliations or roles, this information could be valuable. Solutions in this case range from generating fake traffic to employing full-fledged systems that provide anonymous and untraceable communication.

8 Applications and Performance Evaluation

In this section, we demonstrate the practical applicability of our secret handshake protocols in two ways. First, we describe our experience integrating our dynamic matching protocol into the key management functionality of IPsec and evaluate its performance in a prototype implementation. We then evaluate the performance of an implementation of our fuzzy handshake scheme. We also compare the performance of our fuzzy scheme to that of the original fuzzy IBE.

8.1 IPsec integration

The original work on secret handshakes [5] showed how to use a secret handshake to authenticate the SSL/TLS handshake [16]. SSL operates at the transport layer of the network stack and can be used to provide application-level authentication. In this paper, we integrate our dynamic matching protocol at the IP level by extending the key exchange capability of IPsec [25]. Since IPsec operates at the network level, it is the most general and flexible way to achieve security. Our integration thus allows *any* services using IP to benefit from the private authentication guarantees afforded

by the secret handshake paradigm. Our dynamic matching protocol is well-suited for integration with IPsec because of its flexibility and efficiency. Communicating parties can specify arbitrary groups and roles during the handshake, allowing for use across administrative and application domains. As will be shown, our protocol performs well enough to be useful in practice.

8.1.1 Integration description

IPsec implements services for confidentiality, authentication, and key management. Communicating entities negotiate and establish security associations (SA's), which describe and dictate the way in which traffic flowing between them is protected. In establishing an SA, two nodes running IPsec use a key exchange protocol to agree on shared keying material, from which encryption and authentication keys are derived. We integrated our dynamic matching protocol into the Internet Key Exchange protocol (IKE) [19], which IPsec uses to establish these shared keys. Specifically, we replace the authenticated Diffie-Hellman exchange described in the standard with our own protocol. Our prototype extends the functionality of the openswan-2.4.5 implementation of IPsec [1], which supports IKEv1. We note that our extensions can be embedded into the message exchanges of the more recent IKEv2 [24] using similar techniques and are therefore also applicable to the current standard.

IKEv1 is designed within the framework of the Internet Security Association and Key Management Protocol (ISAKMP) [32]. ISAKMP defines the message exchanges and payload formats that can be used to negotiate the parameters of protected channels and establish shared keys. Establishing security associations proceeds in two phases. In the first phase, two nodes running IPsec (the initiator and the responder) set up an ISAKMP security association, which is used to protect further traffic between their keying daemons. In the second phase, the daemons negotiate and set up an IPsec security association, which protects traffic between the ends of the SA. We embed our handshake protocol into the Phase One key exchange. The standard defines two exchange modes for Phase One (Main and Aggressive). We integrated our protocol into Main Mode, which is the identity-protection mode of ISAKMP, in which key exchange and authentication material are transmitted separately.

Main Mode consists of three round-trips. In the first round, the initiator and the responder agree on several security parameters, including encryption and hash algorithms and parameters of the key exchange algorithm to be used. These parameters are contained in a Proposal payload, which contains one or more Transform payloads. The Proposal payload indicates that a handshake-based key agreement is to be run. We place a description of the public parameters used by our protocol in a Transform payload. Specifically, we include a group identifier field, which refers to the particular set of public parameters used by the keying daemons running the handshake. Since these messages are sent in the clear, we assume a sufficient number of groups exists such that an eavesdropper cannot connect the parties participating in the handshake to a group just from the fact that the messages are sent.

In the second round, the keying daemons perform a key exchange. We embed the credentials sent in our dynamic matching protocol in a Key Exchange payload. Upon receiving a Round 2 message, a daemon generates shared keying material by performing the pairing computations detailed in our protocol. It then derives encryption and authentication keys from the shared keying material in the same way as defined by IKE.

When a Diffie-Hellman key exchange is used (as described in the standard), the third round of Main Mode is used to authenticate the exchange. Messages sent in the third round are encrypted using the shared encryption key established after the second round, and they contain Identification and Authentication payloads. We note, however, that this notion of authentication is at odds with the guarantees of the secret handshake paradigm: having each party sign with its secret group credentials would violate the privacy-preserving property. One impact of this is that we cannot, in general, provide integrity against an active attacker who tries to subvert the protocol by flipping bits or introducing noise into the channel. In this case, the handshake will fail, but the attacker cannot be detected, since this is indistinguishable from the case where one of the parties did not meet the other's requirements. While not implemented, one way to overcome this problem would be if each party had a set of public, non-secret credentials in addition to its secret credentials, which is likely in environments where handshakes are run between parties with "double identities." Each party could sign using her non-secret credentials in the last round of the exchange to achieve integrity. Our current implementation uses the third round only to exchange encrypted Identification payloads, containing the IP addresses of each party.

After Main Mode completes, the two nodes run Phase Two, or Quick Mode, to negotiate an IPsec SA, detailing the services to be run on the protected channel (e.g., encryption and/or authentication, tunnel mode, etc.). Our prototype implementation makes no modification to Quick Mode, as the keying material generated in Main Mode can be used to quickly set up IPsec security associations.

8.1.2 Integration environment and results

We integrated our dynamic matching protocol into the openswan-2.4.5 implementation of IPsec. We configured

openswan on two Centaur VIA Nehemiah, 1000 Mhz machines with 256MB memory running Fedora Linux Core 4, kernel 2.6.11. Key management is handled by a userland daemon called pluto, which we extended.

We implemented our dynamic matching protocol as a suite of three C++ programs, one for each portion of the handshake protocol. The Handshake program runs as a cryptographic server, with pluto as its client. The two processes interact by passing messages to each other.

We use the Miracl [37] cryptographic library for performing big number operations. Miracl provides efficient tools for generating elliptic curve parameters and for the pairing operations required by our protocol.

The dynamic matching handshake protocol is run using DDH-hard subgroups of an MNT elliptic curve with pairings. For simplicity, we used a pre-generated curve provided with the Miracl library. The curve has an embedding degree of $k = 6$ and the subgroup $\mathbb{G}_1$ has prime order p, where p is 157 bits long. We use an identity size of 64 bits, with 32 bits each for the group identifier and role.

To evaluate the performance our handshake protocol, we compared its latency with that of the original openswan implementation, which by default uses the Oakley authenticated Diffie-Hellman protocol [34], with 1536-bit RSA signatures for authentication. We measured the time for the initiating daemon to complete both phases (i.e., the time to build an IPsec SA). The average time for our dynamic matching protocol, measured across several runs with various seeds, was 0.78 seconds, and the average time for the Oakley protocol was 0.5 seconds. Thus, while more expensive than a Diffie-Hellman exchange, we believe our protocol is efficient enough to be useful in practice. The efficiency of our protocol stems from the fact that it uses only three pairing operations per party and only requires operations in $\mathbb{G}_1$, which are considerably cheaper than operations in $\mathbb{G}_2$.

8.2 Fuzzy handshake performance

We also implemented the fuzzy handshake scheme of Section 6. We focus on the performance of Steps 2–5 of the protocol; namely, we assume that Alice and Bob have already executed Step 1 (set intersection and circuit evaluation) to jointly learn if the handshake has a chance of succeeding. The overhead of Step 1 due to set intersection protocols is the same as what we report for Step 2, and the the performance of secure two-party circuit evaluation, according to existing results [31], is reasonably fast and practical for small circuits.

In Step 2 of the fuzzy handshaking protocol, Alice and Bob run a secure set intersection protocol to determine the overlapping attributes in their respective sets. We chose to implement the set intersection protocol of [2] for use in

Step 2 due to its simplicity. We note that the protocol is secure only in the random oracle model. One could use a different set intersection protocol, such as the one in [17], for security in the standard model.

The set intersection protocol of [2] uses as a building block a commutative encryption function. Our implementation uses the power function as the encryption function, i.e., $f_e(x) \equiv x^e \bmod p$, where p is a safe prime and Dom $\mathcal{F}$ is the set of all quadratic residues modulo p. We use repeated hashing to map attributes into Q_p.

8.2.1 Evaluation environment and parameters

Our experiments were performed on a 2.8GHz Pentium 4 machine with HT technology and 1GB of RAM, running Fedora Linux Core 4, kernel 2.6.13.

As before, we use the Miracl [37] cryptographic library for performing big number operations, with the same pre-generated curve as the one described in Section 8.1.2. The set intersection protocol run in Step 2 uses the power function modulo a random 512-bit safe prime for commutative encryption.

Each user is associated with a file containing the list of n attributes making up her identity, in addition to a file that describes those attributes the user wants the other party to have when engaging in the handshake protocol. Attributes are chosen from the universe of elements, $\mathcal{U}$, consisting of those contiguous elements from $\mathbb{Z}_p^*$, beginning with 1, for which a parameter is generated in the Setup phase. The overlap parameter, d, specifies the minimum set overlap needed for a successful handshake. We vary these parameters in the experiments below to assess their impact on the performance of the protocol.

8.2.2 Results

Setup: The Setup phase generates the parameters needed in the remaining stages. We assume a suitable elliptic curve has already been chosen and measure the latency for computing the remaining parameters. Note that the Miracl library can be used to generate fresh pairing-friendly MNT curves in several seconds if so desired.

We measured the total latency of the Setup phase as we scale up the universe size from 10 to 1000 attributes. The Setup phase takes 0.89 seconds to generate parameters with a universe size of 10, 7.46 seconds with a universe size of 100, and roughly 73 seconds with a universe size of 1000 attributes. As expected, we observe that the latency scale linearly in the universe size, with a cost of roughly 75 milliseconds per additional attribute.

AddMember: The AddMember phase of the protocol takes as input the group's master secret and the user's identity and generates credentials that the user employs in subsequent
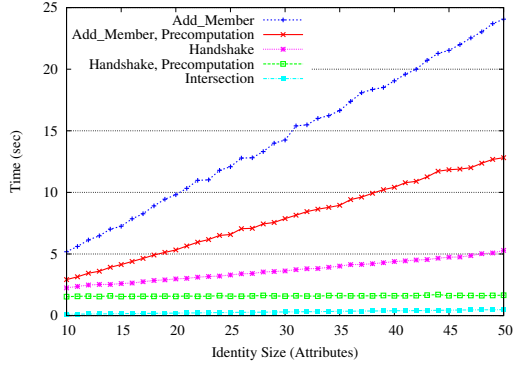
Figure 1. Fuzzy Handshake Performance

Figure 2. Fuzzy Handshake, Overlap Scalability

handshakes. The user receives two credential elements in $\mathbb{G}_2$ per attribute in her identity. Generation of the first element, d_i, involves one exponentiation in $\mathbb{G}_2$, while generation of the second element, D_i, requires evaluation of the public function T_2.

To measure the cost of the AddMember phase, we generated random identities consisting of between 10 and 50 attributes, chosen from a universe size of 100. The top line in Figure 1 shows the total time needed to generate the user's credentials for each identity size. We can see a linear increase in the identity size, from roughly 5 seconds at 10 attributes to just below 25 seconds at 50 attributes.

To achieve better performance at the cost of additional storage space, the group authority can precompute the result of T_2. As seen in Figure 1, the precomputation significantly decreases the amount of time required, from 2.9 seconds at 10 attributes to 12.8 seconds at 50 attributes.

Handshake: In Step 2 of the fuzzy handshake protocol, two users compute a secure set intersection on their appropriate vectors. Figure 1 shows that the intersection protocol adds only a small amount of latency in relation to the overall cost of the handshake, ranging from 0.1 seconds at 10 attributes to 0.5 seconds at 50 attributes. This cost is mostly due to the application of the encryption function to both sets of vectors.

After computing the set intersection, the users engage in the handshake, which uses $2d + 1$ pairings per user to generate the two keys. Each user also evaluates T_1 on n attributes as part of the exchange.

We first investigated the impact of increasing the number of attributes in the users' credentials on the handshake latency. The universe size and overlap variables were kept constant at 100 and 10 attributes, respectively. We generated random pairs of overlapping identities for each identity size. Figure 1 shows a linear increase in latency as the number of attributes increases, from 2.25 seconds when 10 attributes are used, to 5.29 seconds when 50 attributes are
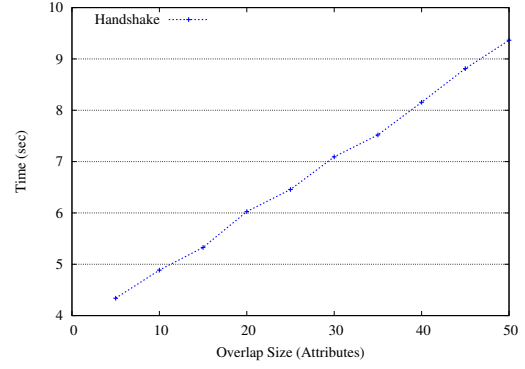
used. This increase is attributed to the cost of additional evaluations of T_1.

For some applications, a user may be likely to use the same vector of desired attributes (e.g., u'_B for Alice) across multiple handshakes. To optimize performance, the T_1 evaluations can be precomputed. The effect of this precomputation is shown in Figure 1. Since the overlap parameter remains constant, the slight increase in latency reflects one additional exponentiation in $\mathbb{G}_1$ per attribute, which costs about 1.6 milliseconds.

Finally, we investigated the impact of increasing the overlap variable, d, which results in more pairing computations and multiplications to compute the key. For this experiment, we maintained a constant universe size of 100 attributes, with 50 attributes in each user's credentials. Figure 2 shows the effects of increasing d from 5 to 50 attributes. The difference in latency, from 4.34 seconds with $d = 5$ to 9.36 seconds with $d = 50$, represents the overhead of the extra pairings and multiplications.

Remarks: Our measurements show that the fuzzy handshake protocol has the ability to perform reasonably well in practice. The Setup phase scales linearly in the universe size, while AddMember scales linearly in the user's identity size. A significant reduction in the latency of AddMember can be achieved via precomputation. As the price of storage continues to decrease, it is reasonable to assume that a group administrator will be willing to trade off the additional storage cost for the performance benefits of precomputation. Similarly, precomputation can reduce the cost of the handshake when a member reuses its vector of desired attributes. This benefit becomes more pronounced as the identity size increases, although memory-scarce devices may prefer to pay a higher cost in latency for reduced storage overhead. While the pairing computation remains computationally expensive, the protocol scales well in both the attribute set size and the overlap size.

8.3 Performance of Fuzzy IBE

Because Attribute-Based Encryption (ABE), and fuzzy IBE as a specific instance of ABE, was introduced only recently, in this section we would like to comment on the performance of our solution, which was designed to achieve privacy, as compared to the original scheme. The only implementation of fuzzy IBE that we are aware of appeared very recently [35] and provides (among other performance results) performance evaluation of KeyGen, Encrypt, and Decrypt algorithms using MNT curves. The goal of this section is then to compare the performance of KeyGen (which corresponds to AddMember in SHS), Encrypt, and Decrypt operations in our implementation of fuzzy IBE using asymmetric groups and the corresponding computations in our modified scheme.

Before presenting the data we collected, we describe how the experiments were conducted.

Fuzzy IBE Scheme using Asymmetric Groups: The best implementation choice of the fuzzy IBE scheme (described in Section A.2) using asymmetric groups $\mathbb{G}_1$ and $\mathbb{G}_2$ is to set $g, g_1 \in \mathbb{G}_2$ and $g_2 \in \mathbb{G}_1$. This gives us that the computation of the function $T(\cdot)$ will always be in $\mathbb{G}_1$, operations in which are significantly more efficient than in $\mathbb{G}_2$. In this case, a private key for identity w consists of values $\{D_i\}_{i \in w} \in \mathbb{G}_1$ and $\{d_i\}_{i \in w} \in \mathbb{G}_2$.

As suggested in [35], decryption optimization is possible by decrypting the message using the order of operations as in

$$M = \frac{E' \prod_{i \in S} e\left(E_i^{L_{i,S}(0)}, d_i\right)}{e\left(\prod_{i \in S} D_i^{L_{i,S}(0)}, E''\right)}$$

instead of

$$M = E' \prod_{i \in S} \left(\frac{e(E_i, d_i)}{e(D_i, E'')}\right)^{L_{i,S}(0)}$$

Our results indicate that in this scheme the above order of computation results in a significant improvement of the decryption time. We also add a slight optimization to the encryption operation by computing the first term of the encryption tuple E' as $e(g_2^s, g_1)$ instead of $e(g_2, g_1)^s$.

Finally, we measure the performance of KeyGen and optimized Encrypt algorithms assuming that the evaluations of function T have been precomputed.

Our SHS-based Scheme: Recall that for privacy reasons the values composing an encryption of a message have to be in the same group (in our case, they are in $\mathbb{G}_1$). Then all parts of the private keys $\{D_i\}_{i \in w}$ and $\{d_i\}_{i \in w}$ are in $\mathbb{G}_2$. Also, computation of the function $T(\cdot)$ in both $\mathbb{G}_1$ and $\mathbb{G}_2$ is needed. Fortunately, computing T in $\mathbb{G}_2$ is needed
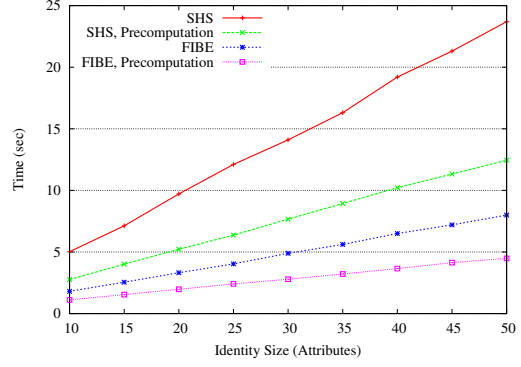

Figure 3. Comparison of Fuzzy IBE KeyGen **and our** AddMember **Performance**

only during the issuance of secret keys, which is performed once per user by the (powerful) server, but not during encryption/decryption operations. The need to evaluate T in $\mathbb{G}_2$ means that our AddMember algorithm is unavoidably slower than KeyGen in the original scheme.

Unfortunately, the decryption optimization described above does not result in faster performance in our SHS-based scheme. In our case, this optimization amounts to computing:

$$M = \frac{E' \prod_{i \in S} e\left(E_i^{L_{i,S}(0)}, d_i\right)}{e\left(E'', \prod_{i \in S} D_i^{L_{i,S}(0)}\right)}.$$

As can be seen from the equation, this order of operations requires us to execute additional operations in $\mathbb{G}_2$ (as opposed to $\mathbb{G}_1$ in fuzzy IBE), where operations are expensive, mitigating the benefits of the reduced number of pairing computations. The other kind of optimization (though very minor) is still possible by computing $e(g_\alpha^x, h)$.

Similar to the previous case, we measure the performance of the regular, optimized, and optimized with precomputation implementations.

The execution environment and implementation parameters we used here were the same as those described in Section 8.2. Figure 3 compares performance of fuzzy IBE KeyGen and our AddMember algorithms for different identity sizes. Figure 4 provides a comparison of the encryption operation in fuzzy IBE and the corresponding computation in our scheme. Note that the optimization of the encryption operation results in negligible advantage, and for clarity of the figure we do not plot such results. Figure 5 shows the results of our experiments for the decryption operation. As was mentioned above, the optimization technique proposed in [35] does not offer computational advantage for our scheme, and thus such results are not included in the figure. Also, since precomputation cannot be done for de-
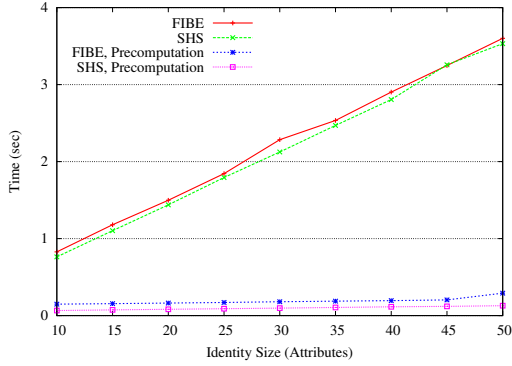
Figure 4. **Comparison of Encryption Computations in Fuzzy IBE and our SHS-based Scheme**
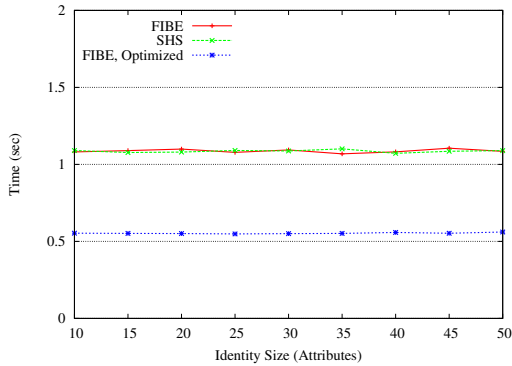

Figure 5. **Comparison of Decryption Computations in Fuzzy IBE and our SHS-based Scheme**

Figure 6. **Comparison of Decryption Computations, Overlap Scalability**

cryption operations, such data is not available in this case. Similar to the experiments of Section 8.2, we varied the identity size in Figures 3, 4, and 5 from 10 to 50 attributes with the universe size of 100 and the overlap size of 10 attributes.

Finally, since performance of the decryption operation is determined by the threshold d (minimum overlap size required) rather than the number of attributes in the identity, Figure 6 illustrates the performance of this operation as a function of the overlap size. We varied the overlap size from 5 to 50 attributes, with the universe size of 100 and the identity size of 50 attributes.

We can see from Figure 4 that our encryption computation achieves very similar (and even slightly faster) performance to the fuzzy IBE scheme, both with and without precomputation. Figure 5 shows that our decryption computation has the same cost as the unoptimized fuzzy IBE scheme, but that the optimization for fuzzy IBE results in a significant speedup. While the SHS-based key generation and decryption computations result in slower performance,
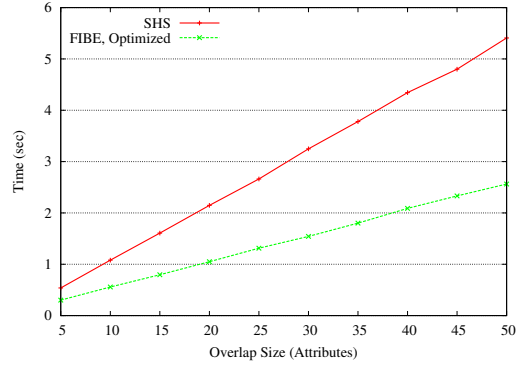
our scheme achieves key privacy, which is not available in the original fuzzy IBE scheme. Thus, the difference in the performance can be viewed as the cost of privacy in this setting.

9 Conclusions

This works extends the original definition of secret handshakes where two users engage in a mutual authentication protocol and jointly agree on a key for further communication only if both of them are members of the same group. In this work, we allow each participant to specify the role and group of the other party and thus add flexibility to the authentication rules. We call such authentication rules dynamic matching. Furthermore, we extend secret handshakes to support arbitrary attributes and define approximate or fuzzy matching authentication rules. Additionally, our result is the first fully unlinkable and efficient secret handshake scheme secure in the standard model.

We provide an implementation showing the feasibility of our approach. Our implementation of secret handshakes with dynamic matching was integrated into IPsec by extending its key exchange capability, and our implementation of secret handshakes with fuzzy matching showed that their performance is efficient enough to be used in practice. Additionally, we provide performance of the original fuzzy IBE scheme and compare it to the performance of our modified approach which achieves user privacy.

Acknowledgments

fuzzy secret handshakes. We also thank the anonymous referees for their insightful comments.

References

[1] The openswan project, http://www.openswan.org/.

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *ACM SIGMOD International Conference on Management of Data*, pages 86–97, 2003.

[3] G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable RFID tags via insubvertible encryption. In *ACM Conference on Computer and Communications Security (CCS'05)*, pages 92–101, Nov. 2005.

[4] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive: Report 2005/385, http://eprint.iacr.org/2005/385, 2005.

[5] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[6] L. Ballard, M. Green, B. de Medeiros, and F. Monrose. Correlation-resistant storage via keyword-searchable encryption. Cryptology ePrint Archive: Report 2005/417, http://eprint.iacr.org/2005/417, Nov. 2005.

[7] D. Boneh and X. Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT'04*, volume 3027 of *LNCS*, 2004.

[8] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology – CRYPTO'04*, volume 3152 of *LNCS*, 2004.

[9] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO'04*, volume 3152 of *LNCS*, pages 41–55, 2004. Full version available at http://crypto.stanford.edu/~dabo/abstracts/groupsigs.pdf.

[10] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[11] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – CRYPTO'06*, 2006.

[12] R. Brawshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security (CCS'04)*, 2004.

[13] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from CA-oblivious encryption. In *ASIACRYPT*, volume 3329 of *LNCS*, pages 293–307, 2004.

[14] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):28–34, Dec. 2002.

[15] S. Crosby, I. Goldberg, R. Johnson, D. Song, and D. Wagner. A cryptanalysis of the high-bandwidth digital content protection system. In *ACM Workshop on Security and Privacy in Digital Rights Management (DRM'01)*, pages 192–200, 2001.

[16] T. Dierks and C. Allen. RFC 2246: The TLS protocol version 1, Jan. 1999.

[17] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – EUROCRYPT'04*, volume 3027 of *LNCS*, pages 1–19, 2004.

[18] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. http://eprint.iacr.org/.

[19] D. Harkins and D. Carrel. RFC 2409: The Internet Key Exchange (IKE), Nov. 1998.

[20] J. Holt. Reconciling CA-oblivious encryption, hidden credentials, OSBE and secret handshakes. Cryptology ePrint Archive Report 2005/215, 2005. http://eprint.iacr.org/2005/215.

[21] J. Holt, R. Bradshaw, K. Seamons, and H. Orman. Hidden credentials. In *ACM Workshop on Privacy in the Electronic Society*, pages 1–8, 2003.

[22] J. Holt and K. Seamons. Reconciling CA-oblivious encryption, hidden credentials, OSBE, and secret handshakes. Technical Report 2006-5 (Internet Security Research Lab), Brigham Young University, June 2006.

[23] S. Jarecki, J. Kim, and G. Tsudik. Authentication for paranoids: Multi-party secret handshakes. In *International Conference on Applied Cryptography and Network Security (ACNS'06)*, June 2006.

[24] C. Kaufman. RFC 4306: Internet Key Exchange (IKEv2) Protocol, Dec. 2005.

[25] S. Kent and R. Atkinson. RFC 2401: Security architecture for the Internet Protocol, Nov. 1998.

[26] A. Kiayias and A. Mitrofanova. Testing disjointness of private datasets. In *Financial Cryptography and Data Security (FC'05)*, volume 3570 of *LNCS*, pages 109–124, 2005.

[27] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology – CRYPTO'05*, volume 3621 of *LNCS*, pages 241–257, 2005.

[28] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *ACM Symposium on Principles of Distibuted Computing (PODC'03)*, pages 182–189, 2003.

[29] S. Li and A. Ephremides. Anonymous routing: A cross-layer coupling between application and network layer. In *Conference on Information Sciences and Systems (CISS'06)*, 2006.

[30] G. Liang and S. Chawathe. Privacy-preserving inter-database operations. In *Symposium on Intelligence and Security Informatics (ISI'04)*, volume 3073 of *LNCS*, pages 66–82, 2004.

[31] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — a secure two-party computation system. In *Usenix Security Symposium*, pages 287–302, 2004.

[32] D. Maughan, M. Schertler, M. Schneider, and J. Turner. RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP), Nov. 1998.

[33] S. Nasserian and G. Tsudik. Revisiting oblivious signature-based envelopes: New constructs and properties. In *Financial Cryptography and Data Security (FC'06)*, 2006.

[34] H. Orman. RFC 2412: The OAKLEY Key Determination Protocol, Nov. 1998.

[35] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security (CCS'06)*, Nov. 2006.

[36] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – EUROCRYPT'05*, volume 3494 of *LNCS*, pages 457–473, 2005.

[37] M. Scott. MIRACL library. Indigo software, `http://indigo.ie/˜mscott`.

[38] M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive: Report 2002/164, `http://eprint.iacr.org/2002/164`, 2002.

[39] G. Tsudik and S. Xu. Brief announcement: A flexible framework for secret handshakes. In *ACM Symposium on Principles of Distributed Computing (PODC'05)*, page 39, 2005.

[40] G. Tsudik and S. Xu. Flexible framework for secret handshakes (multi-party anonymous and un-observable authentication). Cryptology ePrint Archive, Report 2005/034, 2005. `http://eprint.iacr.org/2005/034`.

[41] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4):593–622, 2005.

[42] D. Vergnaud. RSA-based secret handshakes. In *International Workshop on Coding and Cryptography (WCC'05)*, volume 3969 of *LNCS*, pages 252–274, 2006.

[43] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT'05*, volume 3494 of *LNCS*, pages 114–127, 2005.

[44] S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *ACM Conference on Computer and Communications Security (CCS'04)*, pages 158–167, 2004.

[45] A. Yao. Protocols for secure computations. In *Symposium on Foundations of Computer Science (FOCS'82)*, pages 160–164, 1982.

[46] Y. Zhang, W. Liu, and W. Lou. Anonymous communications in mobile ad hoc networks. In *INFOCOM*, 2005.

[47] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Anonymous handshakes in mobile ad hoc networks. In *IEEE Military Communications Conference (MILCOM'04)*, 2004.

[48] L. Zhou, W. Susilo, and Y. Mu. Three-round secret handshakes based on ElGamal and DSA. In *International Conference on Information Security Practice and Experience (ISPEC'06)*, volume 3903 of *LNCS*, pages 332–342, 2006.

A Background

A.1 Identity-Based Encryption

The IBE scheme described in this section is due to Waters [43]. Let $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow Set(1^\kappa)$. As in any other IBE scheme, identities serve the role of public keys and are represented as bitstrings of length n (they can also be represented as bitsrings of arbitrary length and n be the output length of a collision-resistant hash function). The encryption scheme then consists of the following algorithms:

Setup. Choose $\alpha \xleftarrow{R} \mathbb{Z}_p$, and $g_2 \xleftarrow{R} \mathbb{G}$, where g_2 generates $\mathbb{G}$; set $g_1 = g^\alpha$. Also choose $u' \xleftarrow{R} \mathbb{G}$ and an n-length vector $U = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. The public parameters are g, g_1, g_2, u', and U. The master secret is g_2^α.

KeyGen. Let v be an n-bit identity and v_i denote its ith bit. Also let $V \subseteq \{1, \ldots, n\}$ be the set of all indices i for which bit $v_i = 1$. A private key for identity v is generated by first randomly choosing $r \xleftarrow{R} \mathbb{Z}_p$ and then setting the private key to $d_v = (d_1, d_2) = (g_2^\alpha (u' \prod_{i \in V} u_i)^r, g^r)$.

Encrypt. A message $M \in \mathbb{G}_T$ is encrypted for identity v by first choosing $t \xleftarrow{R} \mathbb{Z}_p$ and then constructing the ciphertext as $C = (C_1, C_2, C_3) = (e(g_1, g_2)^t M, g^t, (u' \prod_{i \in V} u_i)^t)$.

Decrypt. Given a valid ciphertext $C = (C_1, C_2, C_3)$ that corresponds to the encryption of M under the identity v, decrypt C using key $d_v = (d_1, d_2)$ as:

$$
\begin{aligned}
C_1 \frac{e(d_2, C_3)}{e(d_1, C_2)} &= (e(g_1, g_2)^t M) \frac{e(g^r, (u' \prod_{i \in V} u_i)^t)}{e(g_2^\alpha (u' \prod_{i \in V} u_i)^r, g^t)} \\
&= \frac{(e(g_1, g_2)^t M) e(g, (u' \prod_{i \in V} u_i)^{rt})}{e(g_1, g_2)^t e((u' \prod_{i \in V} u_i)^{rt}, g)} \\
&= M.
\end{aligned}
$$

A.2 Fuzzy Identity-Based Encryption

Fuzzy IBE [36] is a new type of identity-based encryption, where identities are viewed as a set of descriptive attributes. A user with the secret key for the identity w is able to decrypt a ciphertext encrypted with the public key w' iff w and w' are within a certain distance of each other. We briefly review the construction of [36] next.

Let identities be represented as sets of attributes, and let d represent the minimal number of them that must overlap in order for decryption to succeed (i.e., d represents the error tolerance). Then the authority can create a private key for a user by associating a random $(d-1)$-degree polynomial $q(x)$ with her identity. Let $(p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow Set(1^\kappa)$ and the identities be of length n for some fixed n. The Lagrange coefficient $L_{i,S}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$ is defined as $L_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. Identities will be sets of n elements of $\mathbb{Z}_p^*$.

Setup. Given n and d, choose $g_1 = g^y$, $g_2 \in \mathbb{G}$, and $t_1, \ldots, t_{n+1} \xleftarrow{R} \mathbb{G}$. Let $N = \{1, \ldots, n+1\}$ and let the function $T(x) = g_2^{x^n} \prod_{i=1}^{n+1} t_i^{L_{i,N}(x)}$ (T can be viewed as the function $g_2^{x^n} g^{h(x)}$ for some n-degree polynomial h). The published public key is $g_1, g_2, t_1, \ldots, t_{n+1}$, and the private key is y.

KeyGen. To generate a private key for identity w, a $(d-1)$-degree random polynomial q is chosen such that $q(0) = y$. The private key then consists of a set $\{D_i\}_{i \in w}$, where $D_i =$

$g_2^{q(i)}T(i)^{r_i}$ and r_i is a random element of $\mathbb{Z}_p$ defined for all $i \in w$, and another set $\{d_i\}_{i \in w}$, where $d_i = g^{r_i}$.

Encrypt. To encrypt a message $M \in \mathbb{G}_T$ with the public key w', first choose $s \xleftarrow{R} \mathbb{Z}_p$ and then construct the ciphertext as $E = (w', E' = Me(g_1,g_2)^s, E'' = g^s, \{E_i = T(i)^s\}_{i \in w'})$.

Decrypt. Suppose, given a ciphertext E encrypted with a key for identity w', we would like to decrypt it using a key for identity w such that $|w \cap w'| \geq d$. To do so, choose an arbitrary d-element subset S of $w \cap w'$ and compute:

$$
\begin{aligned}
M &= E' \prod_{i \in S} \left(\frac{e(d_i, E_i)}{e(D_i, E'')} \right)^{L_{i,S}(0)} \\
&= Me(g_1,g_2)^s \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)}T(i)^{r_i}, g^s)} \right)^{L_{i,S}(0)} \\
&= Me(g_1,g_2)^s \prod_{i \in S} \left(\frac{e(g^{r_i}, T(i)^s)}{e(g_2^{q(i)}, g^s)e(T(i)^{r_i}, g^s)} \right)^{L_{i,S}(0)} \\
&= Me(g,g_2)^{ys} \prod_{i \in S} \frac{1}{e(g,g_2)^{q(i)sL_{i,S}(0)}} = M.
\end{aligned}
$$

B Security Definitions and Proofs

In this section we first formally define the properties of impersonation resistance, detection resistance, and unlinkability; and then provide security proofs of both of our constructions with respect to these security properties.

B.1 Security definitions

Consider the following *member impersonation game* for a (polynomial-time) adversary $\mathcal{A}$: $\mathcal{A}$ interacts with users of her choice and obtains secrets for some of them; let $\mathcal{U}_\mathcal{A}$ denote the users that $\mathcal{A}$ controls. Then $\mathcal{A}$ selects a target group G_t a member of which she wants to impersonate, a target role R_t under which she wants to impersonate a user, and a target user U_t such that $U_t \notin \mathcal{U}_\mathcal{A}$ with whom she would like to communicate. In other words, the adversary will interact with U_t trying to impersonate a member of group G_t with role R_t. We also require that $\mathcal{A}$ did not corrupt the GA. $\mathcal{A}$ engages in a handshake interaction with U_t and wins the game if U_t cannot distinguish between $\mathcal{A}$'s messages and the real execution of the handshake protocol and at the end of the interaction $\mathcal{A}$ can compute the same key that U_t obtains. Following the member impersonation games employed in prior literature, we assume that, at the end of the game, $\mathcal{A}$ outputs the key she computed.

Let $\mathsf{Adv}_\mathcal{A}^{\mathrm{imp}}$ denote the probability that adversary $\mathcal{A}$ wins the member impersonation game. Also, let $\mathcal{U}_{G_t,R_t}$ denote

the set of users who are members of group G_t under role R_t. Then the security property of a secret handshake scheme regarding member impersonation can be stated as follows:

Definition 4 (Impersonator resistance) *We say that the secret handshake scheme* SHS *is impersonator resistant if an adversary $\mathcal{A}$ who never corrupts any member of the target group G_t with role R_t and never corrupts the* GA *has at most negligible probability in winning the member impersonation game for G_t and R_t. That is, if $\mathcal{U}_\mathcal{A} \cap \mathcal{U}_{G_t,R_t} = \emptyset$, then $\mathsf{Adv}_\mathcal{A}^{\mathrm{imp}}$ is negligible for all $\mathcal{A}$.*

Note that this bound must hold even if $\mathcal{A}$ corrupts members of G_t or users with roles R_t, as long as a corrupted member does not have group G_t and role R_t simultaneously.

For an adversary $\mathcal{A}$ we also define a *member detection game*. Intuitively, $\mathcal{A}$ cannot detect members of a certain group if her interaction with a group member during a handshake yields no new information to the adversary. More formally, $\mathcal{A}$ should not be able to distinguish between interaction with a group member (having a specific role) and a random simulator. Thus, the member detection game is defined as follows: $\mathcal{A}$ interacts with users of her choice and obtains secrets for some users $\mathcal{U}_\mathcal{A}$. Then $\mathcal{A}$ selects a target user $U_t \notin \mathcal{U}_\mathcal{A}$. A random bit $b \leftarrow \{0,1\}$ is flipped. If $b = 0$, $\mathcal{A}$ interacts with U_t. If $b = 1$, $\mathcal{A}$ interacts with a random simulator. Finally, $\mathcal{A}$ outputs a guess b' for b and wins if $b = b'$. The member detection advantage $\mathsf{Adv}_\mathcal{A}^{\mathrm{det}}$ of $\mathcal{A}$ is defined as the probability of $\mathcal{A}$ winning the member detection game minus 1/2. Then the corresponding security definition is:

Definition 5 (Detector resistance) *Let G_t be the group to which U_t belongs and R_t be the role that she has in G_t. We say that a scheme* SHS *is detector resistant if an adversary $\mathcal{A}$ who did not corrupt any member of G_t with role R_t and did not corrupt the* GA *has at most negligible member detection advantage. That is, if $\mathcal{U}_\mathcal{A} \cap \mathcal{U}_{G_t,R_t} = \emptyset$, then $\mathsf{Adv}_\mathcal{A}^{\mathrm{det}}$ is negligible for all $\mathcal{A}$.*

The last security property that we need to define here is unlinkability of two executions of the secret handshake protocol. Intuitively, in a scheme that support unlinkability an adversary who participates in one handshake protocol and then engages in another should not be able to tell whether she is communicating with the same or a different user. Since we do not want the adversary to make this distinction based on the outcome of the protocol (i.e., if the handshake succeeds in the first execution and fails in the second, she knows that they correspond to members of different groups and the users must be distinct), we will assume that if a different user is chosen for the second, challenge, handshake protocol, the protocol will result in the same outcome as the first execution.

As before, we consider a polynomial-time adversary $\mathcal{A}$ who this time participates in a *linking game*: $\mathcal{A}$ interacts with users of her choice and obtains secrets for some users $\mathcal{U}_\mathcal{A}$. Then $\mathcal{A}$ selects a target user $U_t \notin \mathcal{U}_\mathcal{A}$ such that $\mathcal{A}$ did not corrupt the GA of U_t's group, and engages in a handshake protocols with U_t. A random bit $b \leftarrow \{0,1\}$ is flipped. If $b = 0$, $\mathcal{A}$ engages in a handshake protocol with the same member. If $b = 1$, $\mathcal{A}$ engages in a handshake protocol with a different member whom she did not corrupt. Finally, $\mathcal{A}$ outputs her guess b' for b and wins if $b = b'$. The linking advantage of $\mathcal{A}$, denoted by $\mathsf{Adv}_\mathcal{A}^{\mathrm{link}}$, is the probability that $\mathcal{A}$ wins the linking game minus $1/2$.

Definition 6 (Unlinkability) *We say that a scheme* SHS *is unlinkable if an adversary $\mathcal{A}$ who did not corrupt the members with whom it interacts and the* GA*, has at most negligible linking advantage.*

In secret handshakes with attribute-based matching, the above member impersonation and member detection games and the corresponding definitions must be modified to take into account user attributes. Thus, in the member impersonation game for secret handshakes with fuzzy matching, $\mathcal{A}$ selects the target user U_t and the attributes u, an owner of which $\mathcal{A}$ would like to impersonate. We require that U_t has at most d attributes in common with u and the users corrupted by $\mathcal{A}$ do not have d or more attributes in common in u. Similar modifications apply to the member detection game.

As mentioned earlier, the member impersonation property of our secret handshakes with fuzzy matching holds in the Fuzzy Selective-ID model [36], which we describe next.

Fuzzy Selective-ID: The adversary $\mathcal{A}$ declares the challenge identity u. The challenger runs the setup algorithm, provides $\mathcal{A}$ with public parameters, and creates users. Next, $\mathcal{A}$ is allowed to corrupt users with identities w_j by requesting their private keys from the challenger, as long as the number of attributes that u and w_j have in common is less than d, i.e., $|u \cap w_j| < d$. Finally, $\mathcal{A}$ engages in a handshake protocol with the challenger impersonating the identity u. At the end of the protocol $\mathcal{A}$ outputs the key and wins if she was able to correctly compute it.

B.2 Security of the secret handshake scheme with dynamic matching

Before we proceed with showing the properties required of the handshake scheme, we show that the scheme provides privacy of identities. This result will be used to show other security properties of the scheme. In more detail, the privacy property requires that given two messages $(g^x, (\mathsf{rep}_1(v_1))^x)$ and $(g^y, (\mathsf{rep}_1(v_2))^y)$ for identities v_1 and v_2, respectively (which correspond to the data exchanged

in the secret handshake protocol), it is not possible to tell whether $v_1 = v_2$.

Lemma 1 *Under the SXDH assumption, the secret handshake scheme with dynamic matching provides privacy of identities.*

Proof Let $\mathcal{A}$ be an adversary who can violate the privacy property of the scheme with a certain probability. We then construct an adversary $\mathcal{A}'$ who uses $\mathcal{A}$ as a black box to solve an instance of the decisional Diffie-Hellman problem. $\mathcal{A}'$ receives values $g, g_1, g_2, g_3 \in \mathbb{G}_1$ and must determine whether there is an integer a such that $g_1 = g^a$ and $g_3 = g_2^a$. $\mathcal{A}'$ first sets up the environment for $\mathcal{A}$ by choosing $h \in \mathbb{G}_2$ and initializing the rest of the parameters as in the secret handshake scheme to obtain params $= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, G', G_1, \ldots, G_n)$. $\mathcal{A}'$ adds users as usual, and $\mathcal{A}$ is allowed to communicate with them and corrupt some users who are denoted by $\mathcal{U}_A$.

Then $\mathcal{A}$ receives two messages of the form $(g^x, (\mathsf{rep}_1(v))^x)$ and is asked to decide whether they correspond to the same identity or not. $\mathcal{A}'$ constructs these messages as follows: she chooses $x, y, s, t \overset{R}{\leftarrow} \mathbb{Z}_p$, then forms the first message as (g^{sx}, g_1^{tx}) and the second message as (g_2^{sy}, g_3^{ty}). If $\mathcal{A}$ replies indicating that they correspond to the same identity, $\mathcal{A}'$ answers to its challenge saying that such an integer a exists; and if $\mathcal{A}$ says the messages correspond to different strings, $\mathcal{A}'$ replies saying that no such integer a exists.

It is clear that if $\mathcal{A}$ wins with a non-negligible probability, so does $\mathcal{A}'$, leading to a contradiction to our assumption of $\mathbb{G}_1$ being DDH-hard. Therefore, the scheme must provide privacy of identities. $\qquad\square$

Note that if it is difficult to decide whether such messages correspond to the same identity or not, it implies the difficulty of determining whether they (or one of them) correspond to a specific identity.

Lemma 2 *Under the BDH and SXDH assumptions, the secret handshake scheme with dynamic matching provides impersonator resistance.*

Proof Sketch Let $\mathcal{A}$ be an adversary who attacks impersonation resistance of the secret handshake scheme and can impersonate a member with a certain probability. Then we construct $\mathcal{A}'$ who uses $\mathcal{A}$ to solve an instance of the BDH problem. $\mathcal{A}'$ is given $g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ and the challenge is to compute $e(g_1, g_2)^{abc}$.

The proof strategy we employ is similar to that used in the security proof of Waters IBE scheme [43], which we adapt to the case of asymmetric groups. Therefore, we do not provide the full details of it here.

The simulator $\mathcal{A}'$ sets an integer $m = 4q$, where q is the upper bound on the number of private key queries that an adversary can make, and chooses $k \stackrel{R}{\leftarrow} \{0, n\}$. She then chooses a random vector $X = \{x_i\}_{i=1}^n$, where each $x_i \stackrel{R}{\leftarrow} \{0, m - 1\}$, and also $x' \stackrel{R}{\leftarrow} \{0, m - 1\}$. Additionally, $\mathcal{A}'$ chooses a random vector $Y = \{y_i\}_{i=1}^n$, where each $y_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$, and $y' \stackrel{R}{\leftarrow} \mathbb{Z}_p$.

Recall that, for an identity v, $V \subseteq \{1, \ldots, n\}$ is the set of indices i for which the ith bit of v equals to 1. Following Boneh-Boyen [8] and Waters [43], we define the functions $F(v) = (p - mk) + x' + \sum_{i \in V} x_i$, $J(v) = y' + \sum_{i \in V} y_i$, and $K(v)$ as:

$$K(v) = \begin{cases} 0, & \text{if } x' + \sum_{i \in V} \equiv 0 \ (\text{mod } m) \\ 1, & \text{otherwise} \end{cases}$$

The simulator sets $g = g_1$, $g_\alpha = g_1^a$, $\tilde{g} = g_2$, $h = g_2^b$, $G' = (g_1^b)^{p - km + x'} g_1^{y'}$, $G_i = (g_1^b)^{x_i} g_1^{y_i}$, and outputs public parameters $\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, G', G_1, \ldots, G_n)$. Notice that now $\mathsf{rep}_1(v) = G' \prod_{i \in V} G_i = g_1^{bF(v) + J(v)}$.

When $\mathcal{A}$ corrupts users by querying their private keys, $\mathcal{A}'$ can answer those queries as follows. Suppose that the private key for identity v was requested. If $K(v) = 0$, $\mathcal{A}'$ aborts and submits a random guess. Otherwise, $\mathcal{A}'$ chooses $r \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and computes:

$$\begin{aligned} d &= (d_1, d_2) \\ &= \left((g_2^a)^{\frac{-1}{F(v)}} g_2^r, \ (g_2^a)^{\frac{-J(v)}{F(v)}} \left((g_2^b)^{F(v)} g_2^{J(v)} \right)^r \right). \end{aligned}$$

Observe that $d_1 = g_2^{\tilde{r}}$ and $d_2 = g_2^{ab} \left(g_2^{bF(v) + J(v)} \right)^{\tilde{r}}$, where $\tilde{r} = r - \frac{a}{F(v)}$.

Once $\mathcal{A}$ declared the target user U_t (with credentials G_t and R_t) whom she would like to impersonate, the simulator needs to send to $\mathcal{A}$ a message of the form $(g^x, \mathsf{rep}_1(G_t || R_t)^x)$. Let v^* denote the string $G_t || R_t$. $\mathcal{A}'$ first checks whether $x' + \sum_{i \in V^*} x_i = km$. If the equality does not hold, the simulator aborts and submits a random guess. Otherwise, $F(v^*) \equiv 0 \ (\text{mod } p)$ (which means that $\mathsf{rep}_1(v^*) = g_1^{J(v^*)}$), and $\mathcal{A}'$ sends $(g_1^c, (g_1^c)^{J(v^*)})$ to $\mathcal{A}$. Finally, $\mathcal{A}'$ submits the key that $\mathcal{A}$ outputs as the answer to her challenge. Note that if $\mathcal{A}$ is successful in computing the key, $\mathcal{A}'$ receives from it $e(g_1, g_2)^{abc}$.

A detailed analysis of the success probability of $\mathcal{A}'$ can be found in [43], and we omit it here. $\quad\square$

Proof of Theorem 1

Correctness: If the participants satisfy the rules of the handshake protocol, they will successfully share a common key, as was shown in Section 5.1.

Impersonator resistance: By Lemma 2.

Detector resistance: In the member detection game, the adversary $\mathcal{A}$ selects an (uncorrupted) user of her choice U_t and engages in a handshake with that user. The goal of the adversary is to distinguish between interaction with that user and a random simulator. Since in this scheme during the handshake U_t sends only $(g^x, (\mathsf{rep}_1(v))^x)$, where v is the expected identity of the other end, this message can be generated by anyone. Therefore, $\mathcal{A}$ cannot determine whether it was sent by U_t or a simulator. And by Lemma 1 $\mathcal{A}$ cannot recover v from this message to make her decision based on the string v.

Unlinkability: In the handshake protocol, the participants do not send any information about their own credentials, but instead they only send information about the credentials they request from the other party. This trivially implies that an adversary cannot tell apart requests by the same or different users.

However, since the credentials that users request from others and their own credentials are highly correlated, we require that it is not possible for the adversary to determine whether she was asked to provide the same or different credentials in the linking game. Lemma 1 shows that any adversary cannot do this with a non-negligible probability. $\quad\square$

B.3 Security of the secret handshake scheme with fuzzy matching

Similar to the previous scheme, we first show that, given messages destined for two recipients, it is not feasible to tell whether they were constructed using the same or different identities (i.e., sets of attributes).

Lemma 3 *Under the SXDH assumption, the fuzzy secret handshake scheme provides privacy of identities.*

Proof The proof is similar to the proof of Lemma 1, and the reduction proceeds in the same way. That is, assume that adversary $\mathcal{A}$ attacks privacy of the scheme. We construct $\mathcal{A}'$ who is given an instance of the decisional Diffie-Hellman problem $g, g_1, g_2, g_3 \in \mathbb{G}_1$ and uses $\mathcal{A}$ to make her decision. $\mathcal{A}'$ uses g to set up a fuzzy secret handshake scheme and publishes parameters $\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, n, d, G_1, \ldots, G_{n+1})$. $\mathcal{A}'$ adds users and lets $\mathcal{A}$ to interact with them and corrupt some of them. Then $\mathcal{A}$ receives two messages of the form $(g^x, \{T_1(u_i)^x\}_{i=1}^n)$ and is asked to decide whether they correspond to the same identity or not. To construct these messages, $\mathcal{A}'$ first chooses $x, y, s, t \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $T_2, \ldots, T_n \stackrel{R}{\leftarrow} \mathbb{G}_1$. Next, $\mathcal{A}'$ sets the first message to $(g^{sx}, g_1^{tx}, T_2^x, \ldots, T_n^x)$ and the second message to $(g_2^{sy}, g_3^{ty}, T_2^y, \ldots, T_n^y)$. If $\mathcal{A}$ says that they correspond to the same identity, $\mathcal{A}'$ replies to her challenge saying that there is an integer a such that $g_1 = g^a$

and $g_3 = g_2^a$; if $\mathcal{A}$ says they correspond to different identities, $\mathcal{A}'$ replies saying that such an integer does not exist. Thus, if $\mathcal{A}$ succeeds with a non-negligible probability, so does $\mathcal{A}'$, implying that this scheme provides privacy of identities. $\qquad\square$

Lemma 4 *Under the BDH and SXDH assumptions, the fuzzy secret handshake scheme provides impersonator resistance in the Fuzzy Selective-ID model.*

Proof Let $\mathcal{A}$ be an adversary who attacks impersonation resistance of the fuzzy secret handshake scheme and can impersonate a member in the Fuzzy Selective-ID model with a certain probability. Then we construct $\mathcal{A}'$ who uses $\mathcal{A}$ to solve an instance of the BDH problem. $\mathcal{A}'$ is given $g_1, g_1^a, g_1^b, g_1^c, g_2, g_2^a, g_2^b$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ and the challenge is to compute $e(g_1, g_2)^{abc}$.

The proof strategy we employ is similar to the one used in the security proof of fuzzy IBE scheme [36] (see also [7]), which we adapt to the case of asymmetric groups.

In the beginning of the game $\mathcal{A}'$ receives from $\mathcal{A}$ the challenge identity u^*, which consists of n elements of $\mathbb{Z}_p$.

The simulator $\mathcal{A}'$ sets $g = g_1$, $g_\alpha = g_1^a$, $\tilde{g} = g_2$, and $h = g_2^b$. $\mathcal{A}'$ then chooses a random n-degree polynomial $f(x)$ and computes another n-degree polynomial $u(x)$ such that $u(x) = -x^n$ for each $x \in u^*$ and $u(x) \neq -x^n$ for other x. $\mathcal{A}'$ sets $G_i = (g_1^b)^{u(i)} g_1^{f(i)}$ and $H_i = (g_2^b)^{u(i)} g_2^{f(i)}$ for $1 \leq i \leq n + 1$. Note that now $T_1(i) = (g_1^b)^{i^n + u(i)} g_1^{f(i)}$ and $T_2(i) = (g_2^b)^{i^n + u(i)} g_2^{f(i)}$, and all of G_i's and H_i's are chosen independently at random since $f(x)$ is a random polynomial. Finally, $\mathcal{A}'$ outputs public parameters params $= (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, g_\alpha, G_1, \ldots, G_{n+1})$.

$\mathcal{A}$ now can request private keys of users whose identities overlap with the challenge identity u^* by less than d attributes. When $\mathcal{A}$ corrupts users by querying their private keys, $\mathcal{A}'$ can answer those queries as follows. Suppose that the private key for identity w was requested. Let the set X consist of the overlapping elements $X = u^* \cap w$, the set X' be any set such that $X \subseteq X' \subseteq w$, where $|X'| = d - 1$, and $S = X' \cup \{0\}$.

For each $i \in X'$, the decryption key components D_i and d_i are computed as $D_i = (g_2^b)^{s_i} T_2(i)^{r_i}$ and $d_i = g_2^{r_i}$, where $r_i, s_i \xleftarrow{R} \mathbb{Z}_p$. For each $i \in w \setminus X'$, these values are computed as:

$$D_i = \left(\prod_{j \in X'} (g_2^b)^{s_j L_{j,S}(i)} \right) \times$$

$$\times \ \left((g_2^a)^{\frac{-f(i)}{i^n + u(i)}} \left((g_2^b)^{i^n + u(i)} g_2^{f(i)} \right)^{r'_i} \right)^{L_{0,S}(i)}$$

and

$$d_i = \left((g_2^a)^{\frac{-1}{i^n + u(i)}} g_2^{r'_i} \right)^{L_{0,S}(i)} .$$

In the above we have that $q(i) = s_i$, in addition to having $q(0) = a$. Also, the value of $i^n + u(i)$ will be non-zero for all $i \notin u^*$ including all $i \in w \setminus X'$.

If we let $r_i = \left(r'_i - \frac{a}{i^n + u(i)} \right) L_{0,S}(i)$, then we obtain that $D_i = (g_2^b)^{q(i)} T(i)^{r_i}$ and $d_i = g_2^{r_i}$ (see [36] for more details). This means that $\mathcal{A}'$ is able to construct a private key for identity w.

During the handshake protocol, $\mathcal{A}'$ needs to send to $\mathcal{A}$ a message of the form $(g^x, \{T_1(i)^x\}_{i \in u^*})$. $\mathcal{A}'$ forms it by sending:

$$\left(g_1^c, \{(g_1^c)^{f(i)}\}_{i \in u^*} \right).$$

Since $i^n + u(i) = 0$ for each $i \in u^*$, $(g_1^c)^{f(i)} = T_1(i)^c$ for all $i \in u^*$. Finally, when $\mathcal{A}'$ receives the key that $\mathcal{A}$ outputs, she submits the key as the answer to her own challenge. If $\mathcal{A}$ was able to construct the key correctly, $\mathcal{A}'$ receives exactly $e(g_1, g_2)^{abc}$. $\qquad\square$

Proof of Theorem 2

Correctness: Correctness of the protocol when the number of overlapping attributes is d or more was shown in Section 6.1.

Impersonator resistance: By Lemma 4.

Detector resistance: In this case we deal with an adversary who corrupts users, selects a target user U_t, engages in the handshake protocol with U_t and tries to learn whether U_t possesses a specific attribute or not. When adversary $\mathcal{A}$ engages in the handshake protocol with U_t without having the required d attributes in common, the protocol will fail in the first step, which could trivially be performed by a simulator and $\mathcal{A}$ would not be able to detect the difference.

But even beyond that point in the protocol, everything sent during its execution does not reveal any information about the attributes used. That is, Lemma 3 showed that it is impossible to distinguish whether two different executions of the protocol had different or the same credentials, which implies the difficulty of discovering what attributes were requested. Since the reduction in Lemma 3 used only a single attribute of the identity u, this result could be obtained for any given attribute of u, implying that no information about each individual attribute (not only the overall u) can be discovered.

Unlinkability: As in the secret handshake scheme with roles, the values exchanged during the protocol correspond not to the credentials of the sender but to the expected credentials of the receiver. However, because such values are related to the credentials that the players possess, we require that it is impossible to determine the attributes from the values exchanged and even tell whether they correspond to the same attribute or not. This is exactly what was shown in Lemma 3. $\qquad\square$