A Provably Secure Secret Handshake with Dynamic Controlled Matching

1. Introduction

Parties cooperating in hostile networked environments often need to establish an initial trust. Trust establishment can be very delicate when it involves the exchange of sensitive information, such as affiliation to a secret society or to an intelligence agency. Two mechanisms, *Secret Handshakes* and *Secure Matchmaking*, have tackled this problem, coming up with solutions for secure initial exchange between mistrusting principals. The relevance of this problem as a research topic is evidenced by the number of recent publications on the subject [1, 10, 11, 15, 16].

A *Secret Handshake*, first introduced by Balfanz et al. in [3], is a mechanism devised for two users to simultaneously prove to each other possession of a *property*, for instance membership to a certain group. The ability to prove and verify is strictly controlled by a certification authority, that issues *property credentials* and *matching references* respectively allowing to prove to another user, and to verify another user's, possession of a property. Users are not able to perform a successful handshake without the appropriate credentials and matching references; in addition protocol exchanges are often untraceable and anonymous. Most of the Secret Handshake schemes available in the literature only allow for the matching of own group membership: we will refer to this class of protocols as *own-group membership* secret handshakes.

Matchmaking protocols, presented first in [2], solve the same problem in a slightly different setting: users express "wishes" about the property expected from the other communicating party, and the communication is established only if both users' wishes are mutually matched. The main difference from Secret Handshakes, is the ability of a Matchmaking user to set credential and matching reference, thus freely choosing the properties object of the match.

Recently, Ateniese et al. presented in [1] a scheme that allows Secret Handshake with *dynamic matching*, lifting the own-group limitation by allowing to verify the presence of properties different from the user's own. This scheme is somewhat in between Secret Handshakes and Secure Matchmaking protocols. It inherits from secret handshake the need for credentials issued by an authority; however, the choice of the property to be verified in the other party is left at the discretion of the verifying user, as in Secure Matchmaking.

In this paper, we present the first Secret Handshake scheme with *dynamic controlled matching*: users are required to possess credentials and matching references issued by a trusted certification authority in order to be able to prove and to verify possession of a given property. Therefore the certification authority retains the control over who can prove what and who can disclose which credentials. However verification is dynamic, in that it is not restricted to own property, as opposed to [3, 7, 13, 16, 17].

This new scheme is of clear practical use. For instance, it fulfills the requirements identified by the EU Project R4EGov [9]. In one of the project's use cases, EU justice forces cooperate with one another in order to solve cross-boundary criminal cases. EU regulations define official processes that must imperatively be followed by operating officers: in particular, these processes mandate which institutions must cooperate upon each particular case. During such collaboration, for instance, a member of France's *Ministère de la Défense* must cooperate with a member of the *Bundesnachrichtendienst*, Germany's intelligence service, to investigate on an alleged internal scandal. The two officers may need to meet secretly, and authenticate themselves on-the-fly. Both are definitely reluctant to disclose their affiliation and purpose to anybody but the intended recipient.

It is evident that they cannot use matchmaking or plain secret handshake: the former does not offer any certification on the exchanged properties, the latter only allows matching within the same organization. Handshakes with dynamic matching too fall short of providing a suitable solution for the problem. The freedom of matching any property gives too much liberty to the officials, who must instead strictly abide by EU regulations that mandate which institution must cooperate on a case-by-case basis. Indeed, these officials are acting on behalf of the State and of the people: they must follow rules and ought not make personal choices.

To this end, we propose a novel cryptographic scheme, called SecureMatching, that allows *an authorized prover* to convince *an authorized verifier* that she owns a property (such as group membership). Our work thus addresses requirements that are not met by existing Secret Handshake and Matchmaking protocols, by combining the mandatory control of a third party over credentials and matching references – akin to Secret Handshakes – with the dynamic matching features of Matchmaking. In Section 4 we show, by means of reductionist proofs, that this primitive is secure under the random oracle model, under the assumption that the Bilinear Decisional Diffie-Hellman (BDDH) problem is hard. Finally, we show how to use SecureMatching to build a full-fledged Secret Handshake scheme with dynamic controlled matching.

2. Related Work

Secret Handshakes are first introduced in 2003 by Balfanz et al. [3] as mechanisms designed to prove group membership, and share a secret key, between two fellow group members. The purpose of these protocols is – as pointed out in [16] – to model in a cryptographic protocol the folklore of real handshakes between members of exclusive societies, or guilds.

Since this early work, many papers have further investigated the subject, considerably advancing the state of the art. New schemes have been introduced, achieving for instance reusable credentials (the possibility to generate multiple protocol exchanges out of a single credential with no loss in untraceability) and dynamic matchings (the ability to verify membership for groups different from one's own). Castelluccia et al. in [7] introduce the concept of CA-Oblivious encryption and show how to build a Secret Handshake scheme from such a primitive. Users are equipped with credentials and matching references (in this particular case embodied by a public key and a trapdoor) that allow them to pass off as a group member and to detect one. In [13], Meadows introduces a scheme that is similar to Secret Handshakes, despite the fact that the security requirements are slightly different – for instance, untraceability is not considered. In [10], Hoepman presents a protocol, based on a modified Diffie-Hellman key exchange, to test for shared group membership, allowing users to be a member of multiple groups. In [16], Vergnaud presents a secret handshake scheme based on RSA. In [17], Xu and Yung present the first secret handshake scheme that achieves unlinkability with reusable credentials: previous schemes had to rely upon multiple one-time credentials being issued by the certification authority. However, the presented scheme only offers a weaker anonymity. In [11], Jarecki, Kim and Tsudik introduce the concept of affiliation-hiding authenticated key exchange, very similar to group-membership secret handshakes; the authors study the security of their scheme under an interesting perspective, allowing the attacker to schedule protocol instances in an arbitrary way, thus including MITM attacks and the like. However their scheme is not suitable in our context, since it only allows to verify own group membership and does not consider untraceability of protocol exchanges.

A closely related topic is secure Matchmaking, introduced by Baldwin and Gramlich in [2]. In [18], Zhang and Needham propose a protocol for on-line matchmaking, based on an on-line database service available to all users. In [15], Shin and Gligor present a new matchmaking protocol based on password-authenticated key exchanges [5].

In [1], Ateniese et al. present the first Secret Handshake protocol that allows for matching of properties different from the user's own. Property credentials are issued by a certificate authority. However, the authors study the protocol in the Matchmaking setting, where the matching reference is a low entropy keyword that can be set at each user's discretion.

A related topic is represented by oblivious signature-based envelopes (OSBEs), introduced by Li et al. in [12]; using OSBE, a sender can send an envelope to a receiver, with the assurance that the receiver will only be able to open it if he holds the signature on an agreed-upon message. Nasserian and Tsudik in [14] argue – albeit with no proofs – that two symmetric instances of OSBE may yield a Secret Handshake. The scheme we introduce in Section 3.2 shares some similarities with OSBE, although some substantial differences are present: OSBE does not consider unlinkability and anonymity, as it requires the explicit agreement on a signature beforehand.

3. The Scheme

In this Section we introduce SecureMatching, a novel cryptographic scheme that allows a user to convince a verifier that she owns a given property. We afterward leverage on this building block to create a Secret Handshake protocol used to secure the mutual exchange of property credentials and to share a common key in case of mutual successful verification of properties.

3.1. Preliminaries

We assume that the system includes users from a set of users $\mathcal{U}$. Each user can possess properties drawn from a set of properties $\mathcal{P}$. Given a security parameter k, let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, *)$ be two groups of order q for some large prime q, where the bit-size of q is determined by the security parameter k. Our scheme uses a computable, non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ for which the *Computational Diffie-Hellman Problem (CDH)* is assumed to be hard. Modified Weil or Tate pairings on supersingular elliptic curves are examples of such maps. We recall that a bilinear pairing satisfies the following three properties:

- Bilinear: for $P, Q \in \mathbb{G}_1$ and for $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$
- Non-degenerate: $\hat{e}(P, P) \neq 1$ is a generator of $\mathbb{G}_2$
- Computable: an efficient algorithm exists to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$

We also introduce a one-way hash function $H : \mathcal{P} \to \mathbb{G}_1$. A suitable implementation is the MapToPoint function introduced in [6].

3.2. SecureMatching

SecureMatching is a prover-verifier protocol wherein a prover can convince a verifier that she owns a property. Provers receive credentials for a given property, allowing them to convince a verifier that they possess that property. Verifiers in turn receive matching references for a given property, which allow them to detect possession of that property after the protocol exchange.

Let $P \in \mathbb{G}_1$ be a random generator of $\mathbb{G}_1$. Let $r, s, t, v \in \mathbb{Z}_q^*$ be random values. We set $\tilde{P} \leftarrow rP$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow vrP$. The system public parameters are $\{q, P, \tilde{P}, S, T, V, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, H\}$. The system secret parameters are the values r, s, t and v.

When a user $u \in \mathcal{U}$ joins the system, a secret value $x_u \overset{R}{\leftarrow} \mathbb{Z}_q^*$ is drawn. Then, the value $X_u = x_u s^{-1} rP$ is issued to u through a secure channel; this value is kept secret by the user. Users receive their credentials and matching references through these algorithms, run by a certification authority:

- Certify is executed by the certification entity upon a user's request. The certification entity verifies that the supplicant user $u \in \mathcal{U}$ possesses the property $p \in \mathcal{P}$ she will later claim to have during the protocol execution; after a successful check, the certification entity issues to u the appropriate credential $cred_p = vH(p)$. The user verifies that $\hat{e}(cred_p, \tilde{P}) = \hat{e}(H(p), V)$. If the verification succeeds, she accepts the credential; otherwise she aborts;

- Grant is executed by the certification entity upon a user's request. First of all the certification entity verifies that – according to the policies of the system – the user u is entitled to verify that another user possesses property $p \in \mathcal{P}$. If the checking is successful, the certification entity issues the appropriate matching reference $match_{u,p} = t^{-1}r(cred_p + x_u P)$, where x_u is the secret value associated with user u; the user verifies that

$$\hat{e}(match_{u,p}, T) = \hat{e}(H(p), V) \cdot \hat{e}(X_u, S)$$

If the verification is not successful, she aborts;

Let A be a prover and B a verifier. A has $cred_{p_A}$ to prove possession of property p_A; B holds $match_{B,p_B}$ to detect property p_B. The protocol proceeds as follows:

1. B picks $n \overset{R}{\leftarrow} \mathbb{Z}_q^*$, and sends $N_1 = nP$ and $N_2 = n\tilde{P}$ to A;
2. A checks whether $\hat{e}(N_1, \tilde{P}) = \hat{e}(N_2, P)$; if so, she picks $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sends to B the tuple $disguisedCred_{p_A} =< r_1 cred_{p_A},\ r_2 N_2,\ r_1 r_2 S,\ r_1 r_2 T >$;
3. B checks whether
$$K = \frac{\hat{e}(r_1 cred_{p_A}, r_2 N_2)^{n^{-1}} \cdot \hat{e}(r_1 r_2 S, X_B)}{\hat{e}(r_1 r_2 T, match_{B,p_B})} \qquad (1)$$
equals to one; if so, B concludes that A possesses property p_B (or similarly that p_A and p_B are the same). X_B is the secret value associated to B.

3.3. From SecureMatching *to Secret Handshake*

In order to use SecureMatching to perform secret handshakes, we need two additional characteristics: (i) the capability of establishing a session key out of the protocol exchange and (ii) the assurance that the key is mutually established only if SecureMatching is successful at both sides. If the key is successfully shared by both users, each of them is certain that the other possesses the expected property as defined by the local matching reference. Note that the properties verified by both users need not be identical.

A $\longrightarrow$ B	$n_A P, n_A \tilde{P}$
A $\longleftarrow$ B	$n_B P, n_B \tilde{P}, r_{1B}(cred_{P2} + r_{3B} P), r_{2B}(n_A \tilde{P}), r_{1B} r_{2B} S, r_{1B} r_{2B} T$
A $\longrightarrow$ B	$r_{1A}(cred_{P1} + r_{3A} P), r_{2A}(n_B \tilde{P}), r_{1A} r_{2A} S, r_{1A} r_{2A} T$

Figure 1: Using SecureMatching to build a Secret Handshake

Let us assume two users, Alice and Bob, want to perform a Secret Handshake and share a key if the Handshake is successful. Alice owns the tuple $(cred_{P1}, match_{A,P2},\ X_A)$ and Bob owns the tuple $(cred_{P2}, match_{B,P1}, X_B)$.

Alice and Bob can draw four random values each, $r_{1A}, r_{2A}, r_{3A}, n_A$ for Alice and $r_{1B}, r_{2B}, r_{3B}, n_B$ for Bob. Then – as we can see in Figure 1 – each performs the steps of SecureMatching, with the only exception that Alice sends $r_{1A}(cred_{P1} + r_{3A}P)$ instead of sending $r_{1A}cred_{P1}$. The same applies to Bob, who sends $r_{1B}(cred_{P3} + r_{3B}P)$.

The addition of a random value to the credential, prevents Alice and Bob from checking whether K, as defined in Equation 1, equals to one in case of successful matching. Indeed, K_{Bob}, the value K computed by Bob, equals to $\hat{e}(P,P)^{r_{1A}r_{2A}r_{3A}r}$; similarly, K_{Alice}, the value K computed by Alice, equals to $\hat{e}(P,P)^{r_{1B}r_{2B}r_{3B}r}$.

However, Alice can compute the values $K' = (K_{Alice})^{r_{1A}r_{2A}r_{3A}}$; similarly, Bob can compute $K'' = (K_{Bob})^{r_{1B}r_{2B}r_{3B}}$, and – in case of successful simultaneous matching – $K' = K''$. This value can be subsequently used to derive a secret key, shared between Alice and Bob only if the matching is successful.

4. Security Analysis

The security requirements of the SecureMatching protocol and of the Secret Handshake based on it can be effectively resumed as follows. With the focus on properties, an attacker can perform three different types of actions: *linking*, *knowing* and *forging*. Linking refers to the ability of an attacker to recognize a common property in two separate instances of the protocol, without the appropriate matching references. Knowing refers to the unfeasibility of a verifier to detect a prover's property without the appropriate matching reference. Finally, forging refers to the unfeasibility of a prover to convince a verifier that she possesses a given property without the appropriate property credential.

Before the actual analysis, let us spend a few words on untraceability. Untraceability refers to the unfeasibility for a verifier to link any two protocol executions to the same prover. In particular, the verifier should not be able to tell apart users she has interacted with, by running a successful matching for a common property. The satisfaction of this requirement by the presented scheme is trivially proved: given any property $p \in \mathcal{P}$, the associated credential $cred_p$, from which the disguised credential is derived, does not contain any information other than the master secret v and the hash of the property $H(p)$. None of this information can be used to identify a user among those that possess the same property.

In the rest of this section we introduce three games, Trace, Detect and Impersonate, that capture the essence of the attacks mentioned above, and we show the impossibility of these attacks. A complete description of the security definition and attacker model can be found in [1, 3].

Notice that we prove the security of our scheme in the exact same setting as the one chosen in the closest state-of-the-art paper by Ateniese et al. [1], which in turn is similar to the one chosen by Balfanz et al. in [3]. To estimate the success probability of the attacker, we can use the same technique used by Balfanz et al. in [3]; we therefore omit the detailed probability estimation here. Before proceeding further, we state the well-known BDDH problem:

Definition 1 (*Bilinear Decisional Diffie-Hellman* Problem)**.** *We say that the Bilinear Decisional Diffie-Hellman Problem (BDDH) is hard if, for all probabilistic, polynomial-time algorithms B,*

$$\mathsf{AdvBDDH}_B := Pr[B(P, aP, bP, cP, xP) = \textit{ true if } x = abc] - \tfrac{1}{2}$$

is negligible in the security parameter.

This probability is taken over random choice of $P \in \mathbb{G}_1$, a, b, c and $x \in \mathbb{Z}_q^*$. This problem has been extensively used in the literature, for instance in [8]. The security proofs for the scheme follow from the hardness of the BDDH problem in the random oracle model, as introduced by Bellare and Rogaway in [4], whereby the hash function H is considered a truly random oracle.

4.1. Security of SecureMatching

In this Section we are going to investigate one by one the identified security requirements, showing proofs for each, focusing our analysis on SecureMatching.

4.1.1. Untraceability

Consider an adversary A whose goal is – given any two disguised credentials – to trace them to having been generated from the same credential, so as to prove possession of the same property. The attacker cannot decide whether there is a property that both credentials can be matched to.

A can receive valid credentials and matching references of his choice and can engage in SecureMatching protocol execution with legitimate users. A is then challenged as follows: she is given $disguisedCred_1$ and $disguisedCred_2$, for which she has not received a matching reference, and she returns true if she can decide that a property $p \in \mathcal{P}$ exists, to which both credentials can be matched to. This implies that $K = 1$ for both credentials with matching references in the set $S_{match,p} = \{match_{u_i,p} : u_i \in \mathcal{U}\}$. We call this game Trace.

Lemma 1. *If an adversary A has a non-null advantage*

$$\mathsf{AdvTrace}_A := Pr[A \textit{ wins the game } \mathsf{Trace}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve any given instance of the Bilinear Decisional Diffie-Hellman problem (BDDH).

Proof. We define B as follows. B is given an instance (P, aP, bP, cP, xP) of the BDDH problem and wishes to use A to decide if $x = abc$. The algorithm B simulates an environment in which A operates, using A's advantage in the game Trace to help compute the solution to the BDDH problem. In particular, B acts as an oracle for H.

Setup Here is a description of how the algorithm B works. B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$, sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. She then publishes the public parameter according to the rules of the protocol.

Queries At first, A queries B for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. The queries can be adaptive. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. If p_i has never been queried before, B picks $h_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p_i, h_i) in a table.

Then, B looks up in the table for the values h_i and x_{u_i}, and answers: $H(p_i) = h_iP$, $cred_{p_i} = vh_iP$, $X_{u_i} = x_{u_i}s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vh_i + x_{u_i})(bP)$. A can check that both $\hat{e}(cred_{p_i}, P) = \hat{e}(H(p_i), V)$ and $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold.

Challenge At the end of this phase, A inputs two nonce pairs $N_1 = n_1P, N_1' = n_1\tilde{P}$ and $N_2 = n_2P, N_2' = n_2\tilde{P}$ according to the specification of the protocol. B then produces two hidden credentials constructed as follows:

$$\begin{cases} disguisedCred_1 =< r_1v(aP),\ r_2N_1',\ r_1r_2S,\ r_1r_2T > \\ disguisedCred_2 =< v(xP),\ r_3N_2,\ r_3s(cP),\ r_3t(cP) > \end{cases}$$

where r_1, r_2, r_3 are random values $\in \mathbb{Z}_q^*$. Then, A outputs her decision.

Analysis of A's answer It is straightforward to verify that, if A wins the game, B can give the same answer to solve the BDDH problem. Indeed, if A wins the game, she is able to decide if $\exists \alpha \in \mathbb{Z}_q^*$ such that

$$\begin{cases} r_1 r_2 vab + r_1 r_2 b x_{u1} = r_1 r_2 b(x_{u1} + v\alpha) \\ r_3 vx + r_3 cb x_{u2} = r_3 cb(x_{u2} + v\alpha) \end{cases} \tag{2}$$

are both verified for any user $u_1, u_2 \in \mathcal{U}$. Since this system of equations is by definition valid for any value of x_{u1} and x_{u2}, we can rewrite 2 as

$$\begin{cases} r_1 r_2 vab = r_1 r_2 bv\alpha \\ r_3 vx = r_3 cbv\alpha \end{cases} \tag{3}$$

and solve the first equation as $\alpha = a$. If A wins the game and decides that the two disguised credentials can be matched to the same property, then we can solve the second equation as $x = abc$, which is the positive answer to BDDH. Conversely, $x \neq abc$, which is the negative answer to BDDH. $\square$

4.1.2. Detector Resistance

Consider an adversary A whose goal is to verify presence of a property of his choice without owning the corresponding matching reference. At first, A queries the system for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. She is free to engage in the SecureMatching protocol execution with legitimate users.

A then choses a property $p_* \in \mathcal{P}$, not yet queried in the previous phase, which will be the object of the challenge. She receives $H(p_*)$ and $cred_{p_*}$. Finally she receives a disguised credential. She is then challenged to tell whether K, as defined in Equation 1, equals to one for any matching reference in the set $S_{match,p_*} = \{match_{u_i,p_*} : u_i \in \mathcal{U}\}$ for the property $p_* \in \mathcal{P}$ object of the challenge. A clearly does not posses any of the matching references in S_{match,p_*}. We call this game Detect.

Lemma 2. *If an adversary A has a non-null advantage*

$$\mathsf{AdvDetect}_A := Pr[A \textit{ wins the game } \mathsf{Detect}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve any given instance of the Bilinear Decisional Diffie-Hellman problem (BDDH).

Proof. We define B as follows. B is given an instance (P, aP, bP, cP, xP) of the BDDH problem and wishes to use A to decide if $x = abc$. The algorithm B simulates an environment in which A operates, using A's advantage in the game Detect to help compute the solution to the BDDH problem. In particular, B will run for A an oracle for the hash function H.

Setup Here is a high-level description of how the algorithm B will work. B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. She then publishes the public parameter according to the rules of the protocol.

Queries At first, A queries B for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. The queries can be adaptive. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. If p_i has never been queried before, B picks $h_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p_i, h_i) in a table.

Then, B looks up in the table for the values h_i and x_{u_i}, and answers: $H(p_i) = h_iP$, $cred_{p_i} = vh_iP$, $X_{u_i} = x_{u_i}s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vh_i + x_{u_i})(bP)$. A can check that both $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$ and $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold.

Challenge A then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones not queried in the previous phase. She then queries B for $H(p_*)$ and $cred_{p_*}$. B's response is $H(p_*) = (aP)$ and $cred_{p_*} = v(aP)$. A can check that $\hat{e}(cred_{p_*}, P) = \hat{e}(H(p_*), V)$ holds.

Then A sends to B a pair of nonces $N_1 = nP, n_2 = n\tilde{P}$ according to the specifications of the protocol. B answers by sending the disguised credential

$$disguisedCred =< v(xP), r_1N_1, r_1s(cP), r_1t(cP) > \tag{4}$$

Analysis of A's answer Let's assume $x = abc$. For every user $u_* \in \mathcal{U}$, we can then write

$$K = \frac{\hat{e}(v(abcP), r_1nP)^{n^{-1}} \cdot \hat{e}(r_1s(cP), X_{u_*})}{\hat{e}(r_1t(cP), t^{-1}(cred_{p_*} + x_{u_*})(bP))} = 1 \tag{5}$$

which implies a successful matching for the disguised credential of Expression 4. Indeed

$$r_1vx + r_1bcx_{u_*} - r_1c(vab + x_{u_*}b) = 0 \tag{6}$$

is satisfied $\forall x_{u_*} \in \mathbb{Z}_q^*$ if and only if $x = abc$.

Therefore, if A wins the game and is able to match the disguised credential, thus detecting property p_*, B can give the same answer to the BDDH. □

4.1.3. Impersonation Resistance

An adversary A has as its goal to impersonate a user owning a given credential, which she does not dispose of. At first, A queries the system for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. She is free to engage in SecureMatching protocol execution with legitimate users.

A then choses a property $p_* \in \mathcal{P}$, not yet queried in the previous phase, which will be the object of the challenge. A queries the system for many matching references for property p_* and users $u_j \in \mathcal{U}$ of his choice. A is then challenged in the following way: she receives a nonce value, and she has to produce a valid handshake message, able to convince a user $u_* \in \mathcal{U}$, among the ones not queried before, with a valid matching reference for property p_*, that she owns the credential $cred_{p_*}$. We call this game Impersonate.[1]

Lemma 3. *If an adversary A has a non-null advantage*

$$\mathsf{AdvImpersonate}_A := Pr[A \textit{ wins the game } \mathsf{Impersonate}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve a given instance of the Bilinear Decisional Diffie-Hellman Problem (BDDH).

Proof. We define B as follows. B is given an instance (P, aP, bP, cP, xP) of the BDDH problem and wishes to use A to decide if $x = abc$. The algorithm B simulates an environment in which A operates: B will in particular act as an oracle for H.

Setup B picks random values r, s, t and $v \in \mathbb{Z}_q^*$ and sets $\tilde{P} = rP$, $S = sP$, $T = t(bP)$ and $V = vr(bP)$. She then publishes the public parameter according to the rules of the protocol.

Queries At first, A queries B for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. The queries can be adaptive. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. If p_i has never been queried before, B picks $h_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p_i, h_i) in a table.

Then, B looks up in the table for the values h_i and x_{u_i}, and answers: $H(p_i) = h_iP$, $cred_{p_i} = vh_i(bP)$, $X_{u_i} = x_{u_i}rs^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}r(vh_iP + x_{u_i}P)$. A can check that both $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$ and $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold.

A then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones not queried in the previous phase. She then queries B for $H(p_*)$.

[1]Notice that this game does not prevent an attacker from stealing legitimate users' credentials and claiming to possess their properties. This is common to many Secret Handshakes schemes in the literature, for instance [1]. We could require credentials to be stored on password-protected, tamper resistant hardware; an algorithmic solution however would require an efficient revocation method, which we do not investigate here and leave as a major item for future work.

B's response is aP. A choses many users $u_j \in \mathcal{U}$ of her choice and asks B for $match_{u_j,p_*}$. After picking the values x_{u_j} as in the previous phase, B's response is $match_{u_j,p_*} = t^{-1}r(v(aP) + x_{u_j}P)$ along with $X_{u_j} = x_{u_j}rs^{-1}(bP)$. A can easily check that it is a valid matching reference by verifying that the equivalence $\hat{e}(T, match_{u_j,p_*}) = \hat{e}(H(p_*), V) \cdot \hat{e}(X_{u_j}, S)$ holds.

Challenge After this phase, B sends to A nonces $cP, r(cP)$ according to the protocol, and challenges A to produce $disguisedCred_{p_*}$ for which K of Equation 1 equals to one with matching reference $match_{u_*,p_*}$ and X_{u_*} of a user $u_* \in \mathcal{U}$ not queried in the previous phase.

A answers the challenge with $(A, B, C, D) \in \mathbb{G}_1^4$, and wins the game if K equals to one, which implies $\hat{e}(A, B)^{c^{-1}} \cdot \hat{e}(X_{u_*}, C) = \hat{e}(D, match_{u_*,p_*})$.

Analysis of A's response Let us write $A = \alpha P$, $B = \beta P$, $C = \gamma P$ and $D = \delta P$. Let us assume that A wins the game; then we can write

$$\alpha\beta c^{-1} + \gamma s^{-1} r x_{u_*} b = \delta(t^{-1}rva + t^{-1}rx_{u_*}) \tag{7}$$

If A wins the game, she should be able to convince a user u_* that she owns the credentials for property p_*. B can choose any value for x_{u_*}, since user u_* has never been object of queries before, and this value is unknown to A. Consequently, $\alpha\beta c^{-1}$ and $\delta t^{-1}rva$ must be independent of x_{u_*}. We can then rewrite Equation 7 as

$$\begin{cases} \alpha\beta c^{-1} = \delta t^{-1} rva \\ \gamma s^{-1} r x_{u_*} b = \delta t^{-1} r x_{u_*} \end{cases} \tag{8}$$

Solving the second equation as $\delta = \gamma s^{-1} tb$ and substituting the resulting expression of δ in the first, yields $\alpha\beta = \gamma s^{-1} rvabc$. Therefore if A wins the game, B can decide whether $x = abc$ based on the outcome of $\hat{e}(A, B)^{sr^{-1}v^{-1}} = \hat{e}(C, xP)$. $\square$

4.2. Security of Secret Handshake

In this Section we focus our attention on the security of the secret handshake scheme presented in Section 3.3. We omit the proof for unlinkability because it is a trivial adaptation of the proof of Lemma 1.

As for detection and impersonation resistance instead, we present two new games, ImpersonateSH and DetectSH, inspired on Impersonate and Detect, covering these attacks in the Secret Handshake scenario; in particular, instead of asking the adversary to perform the detection of the property or the impersonation of a user owning a credential, challenger and adversary engage in a Secret Handshake protocol run, at the end of which the adversary receives a key; the adversary is then asked to tell whether the key is the correct key associated to that instance of the handshake or not. This approach is the standard approach used in the proof of authenticated key exchange schemes, requiring key indistinguishability. We stress that this requirement is the strongest possible; other works such as [1, 3] prove resilience to attacks with weaker adversaries: in particular, in the games presented in [1, 3], the attacker is required to actually produce the correct key instead of only distinguish it from a random value.

4.2.1. Impersonation Resistance

An adversary A has as its goal to impersonate a user owning a given credential, which she does not dispose of. At first, A queries the system for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. She is free to engage in Secret Handshakes protocol execution with legitimate users.

A then choses a property $p_* \in \mathcal{P}$, not yet queried in the previous phase, which will be the object of the challenge. A queries the system for many matching references for property p_* and users $u_j \in \mathcal{U}$ of his choice. A also picks a property $p_\circ$; this property is the one the challenger will use to generate its side of the handshake. This is required because what is being tested is the ability of the adversary to impersonate, so the detection part of the handshake should be successful. A is then challenged in the following way: she has to engage in secret handshake with the challenger, she receives a key and she has to tell whether it is the key linked to a successful detection of $p_\circ$ by the adversary and a successful detection of p_* by the challenger, or a random bit string of the same length. We call this game ImpersonateSH.

Lemma 4. *If an adversary A has a non-null advantage*

$$\mathsf{AdvImpersonateSH}_A := Pr[A \textit{ wins the game } \mathsf{ImpersonateSH}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve a given instance of the Bilinear Decisional Diffie-Hellman Problem (BDDH).

Proof. We define B as follows. B is given an instance (P, aP, bP, cP, xP) of the BDDH problem and wishes to use A to decide if $x = abc$. The algorithm B simulates an environment in which A operates: B will in particular act as an oracle for H.

Setup B picks random values r, s, t and $v \in \mathbb{Z}_q^*$ and sets $\tilde{P} = rP$, $S = sP$, $T = t(bP)$ and $V = vr(bP)$. She then publishes the public parameter according to the rules of the protocol.

Queries At first, A queries B for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. The queries can be adaptive. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. If p_i has never been queried before, B picks $h_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p_i, h_i) in a table.

Then, B looks up in the table for the values h_i and x_{u_i}, and answers: $H(p_i) = h_iP$, $cred_{p_i} = vh_i(bP)$, $X_{u_i} = x_{u_i}rs^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}r(vh_iP + x_{u_i}P)$. A can check that both $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$ and $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold.

A then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones not queried in the previous phase. She then queries B for $H(p_*)$. B's response is aP. A choses many users $u_j \in \mathcal{U}$ of her choice and asks B for $match_{u_j,p_*}$. After picking the values x_{u_j} as in the previous phase, B's response is $match_{u_j,p_*} = t^{-1}r(v(aP) + x_{u_j}P)$ along with $X_{u_j} = x_{u_j}rs^{-1}(bP)$. A can easily check that it is a valid matching reference by verifying that the equivalence $\hat{e}(T, match_{u_j,p_*}) = \hat{e}(H(p_*), V) \cdot \hat{e}(X_{u_j}, S)$ holds. Finally, A chooses the property $p_\circ$ that B will use in his matching reference.

Challenge After this phase, A and B engage in a secret handshake instance; in particular A sends nonces nP and $n\tilde{P}$; B verifies that the nonces are compliant, sends to A nonces $cP, r(cP)$ according to the protocol, sends the handshake tuple $(r_1(vH(p_\circ) + r_3cP), r_2n\tilde{P}, r_1r_2S, r_1r_2T)$ and challenges A to produce $disguisedCred_{p_*}$; A answers the challenge with $(A, B, C, D) \in \mathbb{G}_1^4$. B then sends A a key formed as follows:

$$K = \left(\frac{\hat{e}(A, B)}{\hat{e}(C, xP)^{s^{-1}rv}} \right)^{r_1r_2r_3}$$

and challenges A to distinguish the correct key from a random string of the same length. Correct key means the key the challenger computes with matching reference $match_{u_*,p_*}$ and X_{u_*} of a user $u_* \in \mathcal{U}$ not queried in the previous phase. A answers the challenge with a bit b; A wins the game if $b = 0$ iff the key is a random bit string, and $b = 1$ iff the key is correct.

Analysis of A's response Let us write $A = \alpha P$, $B = \beta P$, $C = \gamma P$ and $D = \delta P$. Let us assume that A wins the game and his answer is $b = 1$; B can then write

$$K = \left(\frac{\hat{e}(A, B)}{\hat{e}(C, xP)^{s^{-1}rv}} \right)^{r_1r_2r_3} = \left(\frac{\hat{e}(A, B)^{c^{-1}} \cdot \hat{e}(C, X_{u_*})}{\hat{e}(D, match_{u_*,p_*})} \right)^{r_1r_2r_3c} \tag{9}$$

This follows from how the key in the handshake is computed, see Section 3.3. From Equation 9 we can rewrite

$$\alpha\beta - \gamma rs^{-1}vx = \alpha\beta + \gamma x_{u_*}rs^{-1}bc - \delta t^{-1}rc(va + x_{u_*}) \tag{10}$$

Notice that, as in the proof of Lemma 3, B can choose any value for x_{u_*}, since user u_* has never been object of queries before, and this value is unknown to A. Consequently, $\gamma rs^{-1}vx$ and $\delta t^{-1}rvac$ must be independent of x_{u_*}. We can then rewrite Equation 7 as

$$\begin{cases} \gamma rs^{-1}vx = \delta t^{-1}rvac \\ \gamma x_{u_*}rs^{-1}bc = \delta t^{-1}rcx_{u_*} \end{cases} \tag{11}$$

Solving the second equation as $\delta = \gamma s^{-1}tb$ and substituting the resulting expression of δ in the first, yields $x = abc$, which is the positive answer to the BDDH problem. If instead A answers $b = 0$, through the same calculation we conclude that $x \neq abc$, which is the negative answer to the BDDH problem. Therefore if A wins the game, B can solve the BDDH problem by answering with b. $\square$

4.2.2. Detector Resistance

Consider an adversary A whose goal is to verify presence of a property of his choice without owning the corresponding matching reference. At first, A queries the system for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. She is free to engage in Secret Handshakes protocol execution with legitimate users.

A then choses a property $p_* \in \mathcal{P}$, not yet queried in the previous phase, which will be the object of the challenge. She receives $H(p_*)$ and $cred_{p_*}$. A also picks a property $p_\circ$; this property is the one the adversary will use to generate its side of the handshake. This is required because what is being tested in this game is the ability of the adversary to detect, so the impersonation part of the handshake should be successful. A is then challenged in the following way: she has to engage in secret handshake with the challenger and she receives a key, and she has to tell whether it is the key linked to a successful detection of $p_\circ$ by the challenger and a successful detection of p_* by the adversary, or just a random bit string of the same length A clearly does not posses any of the matching references in S_{match,p_*}. We call this game DetectSH.

Lemma 5. *If an adversary A has a non-null advantage*

$$\mathsf{AdvDetectSH}_A := Pr[A \textit{ wins the game } \mathsf{DetectSH}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses A's advantage to solve any given instance of the Bilinear Decisional Diffie-Hellman problem (BDDH).

Proof. We define B as follows. B is given an instance (P, aP, bP, cP, xP) of the BDDH problem and wishes to use A to decide if $x = abc$. The algorithm B simulates an environment in which A operates, using A's advantage in the game Detect to help compute the solution to the BDDH problem. In particular, B will run for A an oracle for the hash function H.

Setup Here is a high-level description of how the algorithm B will work. B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. She then publishes the public parameter according to the rules of the protocol.

Queries At first, A queries B for an arbitrary number of tuples $< H(p_i)$, $cred_{p_i}$, X_{u_i} and $match_{u_i,p_i} >$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. The queries can be adaptive. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. If p_i has never been queried before, B picks $h_i \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p_i, h_i) in a table.

Then, B looks up in the table for the values h_i and x_{u_i}, and answers: $H(p_i) = h_iP$, $cred_{p_i} = vh_iP$, $X_{u_i} = x_{u_i}s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vh_i + x_{u_i})(bP)$. A can check that both $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$ and $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold.

A then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones not queried in the previous phase. She then queries B for $H(p_*)$ and $cred_{p_*}$. B's response is $H(p_*) = (aP)$ and $cred_{p_*} = v(aP)$. A can check that $\hat{e}(cred_{p_*}, P) = \hat{e}(H(p_*), V)$ holds.

Challenge After this phase, A and B engage in a secret handshake instance; in particular, A sends to B a pair of nonces $N_1 = nP, N_2 = n\tilde{P}$ according to the specifications of the protocol. B verifies that the nonces are compliant, sends to A nonces $n'P, n'\tilde{P}$ according to the protocol, sends the handshake tuple $(v(xP) + r_1bP, r_2N_1, r_2s(cP), r_2t(cP))$ and receives $disguisedCred_{p_\circ} = (A, B, C, D) \in \mathbb{G}_1^4$ from A; B then sends A a key formed as follows:

$$K = K'^{r_1r_2} = \hat{e}(P, \tilde{P})^{r_1r_2\mu} \quad \text{such that} \quad K' = \hat{e}(P, \tilde{P})^{\mu} = \frac{\hat{e}(A, B)^{n'^{-1}} \cdot \hat{e}(C, X_{u_\circ})}{\hat{e}(D, match_{u_\circ,p_\circ})}$$

and challenges A to tell whether K is the correct key or a random string of the same length. Correct key means the key the adversary computes detecting property p_*. A answers the challenge with a bit b; A wins the game if $b = 0$ iff the key is a random bit string, and $b = 1$ iff the key is correct.

Analysis of A's answer Let us at first notice that B computes the key by generating at first the value K' by matching property $p_\circ$ from $disguisedCred_{p_\circ}$, and then raising the result to the power r_1r_2.

If A wins the game and answers $b = 1$, it means that the key B generated was correct. Every user $u_* \in \mathcal{U}$ owning a matching reference $match_{u_*,p_*}$ should be able to compute the same key; we can therefore write

$$\left(\frac{\hat{e}(v(xP) + r_1bP, r_2nP)^{n^{-1}} \cdot \hat{e}(r_2s(cP), x_{u_*}s^{-1}(bP))}{\hat{e}(r_2t(cP), t^{-1}(va + x_{u_*})(bP))}\right)^{\mu} = K = \hat{e}(P, \tilde{P})^{r_1r_2\mu}$$

From this expression we notice that

$$(r_2vx + r_1r_2b + r_2x_{u_*}bc - r_2bc(va + x_{u_*}))\mu = r_1r_2b\mu \tag{12}$$

is satisfied $\forall x_{u_*} \in \mathbb{Z}_q^*$ if and only if $x = abc$. Therefore, if A wins the game B can give the same answer to the BDDH. $\square$

4.3. A Word on Man-In-The-Middle Attacks

In this section we focus on man-in-the-middle attacks and investigate their applicability to the secret handshake scheme. At first let us notice that, upon a successful Secret Handshake protocol run, two users Alice and Bob establish a common secure channel. This channel ensures for Alice that at the other hand there is a user whose credentials match her matching reference; the same holds for Bob. No man-in-the-middle attack can break this assurance: this has been demonstrated by the proofs of Lemma 4 and 5, that assure that an adversary – without the appropriate credential and matching reference – cannot distinguish between a correct key and a random bit string.

However the protocol *does not* give any information about the identities of the communicating parties. Let us see with a few examples the consequences of this feature. Let us imagine that Alice, Bob and Mallory are equipped with credentials and matching references for property p. This scenario is consistent with group-only secret handshakes introduced in [3]. In this scenario the following attack is possible: Alice and Bob try to run a secret handshake on a channel controlled by Mallory; Mallory runs two parallel secret handshakes, one with Alice and another with Bob, establishing two keys – and consequently two channels. At this point, Mallory is in the condition of acting as the man-in-the-middle in the conversation between Alice and Bob.

First of all, let us underline that this attack does not compromise what the protocol guarantees: at the end of the handshake, both Alice and Bob are indeed on a secure channel with another member of their group. In addition, group pressure would ensure that a group member would not mount such an attack to a fellow group member: indeed group membership tokens (credentials and matching references) are issued after a check of the compliance of the user to the group policies, so it can be refused to not-trusted members.

This problem is usually thwarted including identity enforcement in the protocol; this however requires drastic changes to the protocol, because Secret Handshake by definition requires untraceability, unlinkability and anonymity, whereas if the credentials carry a reference to the identity of their possessor, these requirement cannot be easily fulfilled. Indeed, either users reveal their identities upfront, thus losing their anonymity, or they do not know who they are interacting with until the secret handshake is successful.

An apparent solution is represented by the use of pseudonyms together with an efficient revocation mechanism: this way the anonymity of users is not violated and the certification authority can still revoke credentials of misbehaving users. This is the approach adopted by Balfanz et al. [3]. This approach is still not perfect since pseudonyms by definition do not provide users with information on the real identity of the carrier of the credentials; in addition, detecting misbehaving users is not a straightforward task, given that these types of attack are quite stealthy; we argue that the pressure of secret groups is a strong enough argument in favor of the security of our scheme, given that in addition, the most viable alternative solution, pseudonym, does not solve completely the problem.

A different scenario occurs when our scheme is used in a dynamic controlled matching scenario, where users are equipped with credentials and matching references potentially for different properties. The man-in-the-middle attack presented earlier on in this section does not apply any longer; let us see it with an example: Alice is equipped with a credential for p_1 and a matching reference form p_2; Bob is equipped with a credential for p_2 and a matching reference form p_1. To perpetrate the same attack as before, Mallory would be required to possess credentials for both p_1 and p_2 and matching references for both p_1 and p_2. The certification authority can restrict this kind of attacks by refusing to issue credentials and matching references for two different properties; or it can explicitly allow this to particular users that are allowed to act as proxies in the communications of two other users.

5. Conclusion and Future Work

In this paper we have proposed a prover-verifier protocol and a two-party Secret Handshake protocol using bilinear pairings. Our work studies the problem of Secret Handshakes under new requirements, different than the ones considered before in the state of the art, thus completing the landscape of available techniques in the field. As future work, we intend to extend the protocol, allowing the certification authority to revoke credentials formerly issued, in order to cope with compromised users and we intend to study the security of the protocol in the more complete setting suggested in [11].

References

[1] G. Ateniese, M. Blanton, and J. Kirsch. Secret handshakes with dynamic and fuzzy matching. In *Network and Distributed System Security Symposuim*, pages 159–177. The Internet Society, 02 2007. CERIAS TR 2007-24.

[2] R. W. Baldwin and W. C. Gramlich. Cryptographic protocol for trustable match making. In *IEEE Symposium on Security and Privacy*, Los Alamitos, CA, USA, 1985. IEEE Computer Society.

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

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

[5] S. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society, May 1992.

[6] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[7] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[8] H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[9] Europol and Eurojust and Thomas Van Cangh and Abdelkrim Boujraf. Wp3-cs2: The Eurojust-Europol Case Study. http://www.r4egov.eu/resources, 2007.

[10] J.-H. Hoepman. Private handshakes. In F. Stajano, C. Meadows, S. Capkun, and T. Moore, editors, *ESAS*, volume 4572 of *Lecture Notes in Computer Science*. Springer, 2007.

[11] S. Jarecki, J. Kim, and G. Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In *CT-RSA*, pages 352–369, 2008.

[12] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *22nd ACM Symposium on Principles of Distributed Computing (PODC 2003)*, pages 182–189. ACM Press, 2003.

[13] C. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, pages 134–137, 1986.

[14] S. Nasserian and G. Tsudik. Revisiting oblivious signature-based envelopes: New constructs and properties. In *Financial Cryptography and Data Security (FC06)*, 2006.

[15] J. S. Shin and V. D. Gligor. A new privacy-enhanced matchmaking protocol. In *Network and Distributed System Security Symposuim*. The Internet Society, 02 2007.

[16] D. Vergnaud. Rsa-based secret handshakes. In *WCC*, pages 252–274, 2005.

[17] S. Xu and M. Yung. k-anonymous secret handshakes with reusable credentials. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*.

[18] K. Zhang and R. Needham. A private matchmaking protocol. http://citeseer.nj.nec.com/71955.html, 2001.