Federated Secret Handshakes with Support for Revocation

Alessandro Sorniotti[1,2] and Refik Molva[2]

[1] IBM Research
Zurich Research Laboratory
CH8803 Rüschlikon
[2] Institut Eurécom
2229, Route des Crêtes
06560 Valbonne, France
{first.last}@eurecom.fr

Abstract. Secret Handshakes are well-established cryptographic primitives that help two mistrusting users to establish initial trust by proving and verifying possession of given properties, such as group membership. All the Secret Handshake schemes to date assume the existence of a single, centralized Certification Authority (CA). We challenge this assumption and create the first Secret Handshake scheme that can be managed by a federation of separate and mistrusting CAs, that collaborate in the setup of the scheme yet retaining strict control over subsets of the property in the system. The security of the scheme is proved without random oracles.

1 Introduction

A *Secret Handshake* is a distinct form of greeting which conveys membership to club, group or fraternity [1]. Usually a Secret Handshake involves conducting the handshake in a special way so as to be recognizable as such by fellow members while seeming completely normal to non-members. The need for such a secretive initial exchange is motivated by the existence in society of gatherings of individuals, revolving around sensitive topics and therefore secret by nature.

With the increasing role over the past half century of electronic communications in our society, it is natural to expect that the discipline of cryptography should capture the essence of Secret Handshakes and model it into protocols that can be automatically executed by electronic devices. It has indeed been the case, as witnessed by the numerous papers on the subject [3, 4, 9–11, 16–19, 21].

One common trait of all these schemes is that they all rely on a *single* centralized entity, that we shall refer to as certification authority or CA, that is in charge of generating public parameters and of handing over cryptographic tokens to users. While the assumption of a single CA can be justified in simple scenarios, it becomes arguably unrealistic in scenarios where more dynamic matching is possible. Indeed, within such a setting, the same CA may be required to generate Credentials for competing groups or secret agencies of different countries.

M. Soriano, S. Qing, and J. López (Eds.): ICICS 2010, LNCS 6476, pp. 218–234, 2010.

The objective of this paper is therefore the creation of a Secret Handshake scheme that relaxes the requirement on a single centralized CA. To the best of our knowledge, ours is the first effort in this direction. Instead of creating a brand new scheme, we have chosen to extend one of the schemes in the state-of-the-art in order to support the federation of independent and mistrusting CAs.

The choice of what scheme to extend has lead us to conduct an extensive survey of the state of the art which also constitutes a valuable contribution to the literature.

2 A Primer on Secret Handshakes

Secret Handshakes belong to a very specific and yet very complex family of cryptographic protocols. A new Secret Handshake protocol can be better understood by reference to its functional and security requirements. The task of drafting a taxonomy for Secret Handshakes however has, to date, not yet been undertaken. Therefore in this Section, starting from a toy protocol, we introduce all the orthogonal dimensions in the family of Secret Handshakes and describe its design space, by identifying a set of characteristics for these protocols. We then move on to the analysis of the numerous Secret Handshake protocols in the state-of-the-art, explaining what they achieve and how they position themselves within the identified taxonomy.

Secret Handshakes consist of users engaging in a protocol in order to exchange information about a *property*. There are two actions that each user performs during a Secret Handshake: *proving* and *verifying*. Proving means convincing the other party that one possesses the property object of the handshake. Verifying in turn means checking that the other party actually possesses the property object of the handshake.

The core objective of Secret Handshakes can be defined as follows:

Definition 1 (Secret Handshake). *A Secret Handshake is a protocol wherein two users u_i and u_j belonging to a universe of users $\mathcal{U}$ authenticate as possessors of a common property p_* belonging to a universe of properties $\mathcal{P}$.*

A simple toy protocol achieving this objective is the following: users u_i and u_j receive a secret value K_{p_*} associated with property p_*. The two users exchange n_i and n_j, two nonces randomly chosen by each user. After the two nonces are exchanged, each user can compute a value $k = MAC_{K_{p_*}}(n_i||n_j)$, using a message authentication code such as for instance HMAC; both users will compute the same value k only if they both posses the correct secret value K_{p_*}. A proof of knowledge that the same value has been computed by both users accomplishes the proving and verifying actions.

A limitation of the above protocol is that the actions of proving and verifying cannot be separated since they are both accomplished at the same time through the proof of knowledge of k; in turn, k is a function of the nonces and of K_{p_*}: therefore, in the simple protocol described above, the knowledge of K_{p_*} grants at the same time the right to prove and to verify for property p_*. Let us then define the concept of separability:

Definition 2 (Separability). *A Secret Handshake protocol is separable if the ability to prove can be granted without the ability to verify (and vice versa).*

According to Definition 2, the simple toy protocol described above is non-separable. Separability in particular translates into splitting secrets associated with a property – such as K_{p_*} in our previous example – into two separate components: *Credentials* and *Matching References* . Credentials grant the ability to prove to another user the possession of a property. Matching References in turn grant the ability to verify whether another user possesses a property. Now that we have formally introduced Credentials and Matching References, we can underline the fact that, in Secret Handshakes, only legitimate bearers of Credentials should be able to prove possession of a property, and only legitimate bearers of Matching References should be able to verify possession of a property. We can thus refine Definition 1 as follows:

Definition 3 (Secret Handshake). *A Secret Handshake is a protocol wherein two users u_i and u_j belonging to a universe of users $\mathcal{U}$ authenticate as possessors of a common property p_* belonging to a universe of properties $\mathcal{P}$. The authentication is successful if both users possess legitimate Credentials and Matching References for p_*.*

The legitimacy of Credentials and Matching References depends on the particular way in which these are generated. Indeed, different Credentials and Matching Reference generation policies play a crucial role on the control over *"who can prove possession of a property"* and *"who can verify possession of a property"*. We shall refer to *proof-control* and *verification-control* respectively, to refer to these two concepts. For instance, if a certification authority generates Credentials and gives them away only to selected users, it retains the control over the ability to prove.

Let us now investigate the amount of information leaked to an observer from a Secret Handshake execution. At first, we will state a few definitions, taken from [15].

Definition 4 (Anonymity). *Anonymity of a user means that the user is not identifiable within a set of user, the user set.*

Definition 5 (Unlinkability). *Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an observers perspective means that within the system (comprising these and possibly other items), the observer cannot sufficiently distinguish whether these IOIs are related or not.*

We say that a Secret Handshake scheme guarantees Anonymity if the identifiers of the involved users are not revealed throughout its execution. Unlinkability of users instead relates to the ability of an observer to link the same user throughout multiple instances of Secret Handshake. We can therefore say that a Secret Handshake protocol guarantees Unlinkability of users if – upon executing two separate instances of Secret Handshake – an observer is not able to tell whether he is interacting with the same user or two different ones. As far as properties

are concerned, we say that a Secret Handshake protocol guarantees Unlinkability of properties if – upon executing two separate instances of Secret Handshake – an observer is not able to tell whether he is interacting with users holding Credentials for the same property or users holding Credentials for different ones; naturally, this requirement should hold only in case of failed handshake, since in case of success, linking properties is possible by definition.

Let us now introduce the concept of fairness, according to Asokan's definition [2] and understand its relationship with Secret Handshakes.

Definition 6 (Fairness). *An exchange protocol is considered fair if at its end, either each player receives the item it expects or neither player receives any additional information about the other's item.*

In a Secret Handshake scenario, this definition translates to the requirement that either both users learn that they both possess a given property, or they do not learn anything at all. As we have seen, proving knowledge of the computed key to one another is what allows users to learn of a successful handshake. Therefore fairness can be achieved if users can execute a protocol that allows them to exchange fairly the results of a proof of knowledge of the two keys, for instance a challenge-response protocol. Unfortunately, a result from Pagnia and Gärtner [14] shows that fairness in exchange protocols is impossible to be achieved without a trusted third party. Secret Handshake protocols however can achieve some more limited form of fairness. Let us define the following predicate

$\mathfrak{P}$:= "*both participants to the Secret Handshake protocol possess the property object of the handshake*"

We can then introduce the notion of fairness in Secret Handshakes:

Definition 7 (Fairness in Secret Handshake). *Upon termination of a Secret Handshake protocol after either a complete or incomplete execution, either at least one party learns* $\mathfrak{P}$*, or no one learns any information besides* $\neg\mathfrak{P}$*.*

where by $\neg\mathfrak{P}$ we mean the negation of the predicate $\mathfrak{P}$. Definition 7 acknowledges the unfairness of Secret Handshakes, but allows one of the two users, p_{adv}, to have an advantage over the other only under specific circumstances. Indeed, in order for p_{adv} to learn $\mathfrak{P}$, p_{adv} must possess the property object of the handshake. p_{adv} can only learn $\neg\mathfrak{P}$ otherwise.

Thanks to the definitions that we have given so far, we will now go through the Secret Handshakes protocols presented in the literature, underlining how they relate to the dimensions highlighted so far and gradually introducing new features.

2.1 Classic Secret Handshakes

In 2003 [4], Balfanz and colleagues first introduced the notion of Secret Handshake, presenting a scheme based on bilinear pairings. The scheme introduced in the paper is, according to our definitions, a non-separable protocol, since it

is impossible for a user to only verify the membership of another user without proving its own. The protocol guarantees Anonymity thanks to the use of pseudonyms; Unlinkability of users is achieved by providing users with a large number of pseudonyms and by asking users to never reuse them: however, although Unlinkability of users is indeed guaranteed, the solution is suboptimal since it trades off the number of Credentials provided with the number of unlinkable handshakes that a user can perform.

In order to mitigate this issue, Xu and Yung have presented in [21] the concept of k-anonymous Secret Handshakes and of reusable Credentials. Let us start with the latter:

Definition 8 (Reusable Credentials). *A Secret Handshake scheme supports reusable Credentials if some form of Anonymity and Unlinkability are guaranteed and users receive a single Credential.*

Clearly Balfanz *et al.*'s scheme does not support reusable Credential. Xu and Yung's scheme is the first one to support reusable Credentials. This is achieved at the expense of full Anonymity, since a user is only effectively anonymous within a population of $k < |\mathcal{U}|$

In [19], Vergnaud presents three Secret Handshake protocols whose security is based on the RSA assumption. The scheme is similar to Balfanz *et al.*'s, and in particular also does not ensure Unlinkability of users with reusable Credentials.

In [16] Shin and Gligor present a privacy-enhanced matchmaking protocol that shares several features with Secret Handshakes. The protocol operates as follows: users receive anonymous Credentials and run a password-based authenticated key exchange (PAKE, see [5, 6, 8]), where instead of the password, they use self-generated communication wishes, as in matchmaking protocol. This suggests that users may retain proof and verification control; however, after a successful matching of the communication wish through the PAKE, users are requested to show certificates linking the pseudonym that has been declared upfront with the wish that they claim they possessed/were interested in.

In [12] Jarecki and Liu underline the fact that schemes proposed so far either support limited nuances of Unlinkability or support reusable Credentials. Therefore they propose an unlinkable version of Secret Handshake, affiliation/policy hiding key exchanges, wherein Credentials are reusable and yet Secret Handshake executions do not leak the nature of the properties linked with Credentials (called affiliation) and Matching References (called policies). The scheme is based on public-key group-management schemes.

In [11], the same authors strengthen the concept of affiliation-hiding key exchanges to include perfect forward secrecy; the authors also investigate the amount of information leaked in the case of an attacker able to compromise sessions (thus learning if the two users belonging to the session do belong to the same group) and users (thus learning the group that user belongs to). The scheme however relies on pseudonyms and therefore gives up Unlinkability of users.

2.2 Secret Handshake with Dynamic Matching

The concept of *Dynamic Matching* allows users to prove and verify possession of two distinct properties during the execution of a Secret Handshake, as opposed to Secret Handshakes introduced so far, which allowed users to prove and verify the matching of a unique property, that is, membership to a single, common group; we shall refer to the latter type of Secret Handshake as to classic Secret Handshakes from here on.

Definition 9 (Secret Handshake with Dynamic Matching). *A Secret Handshake with Dynamic Matching is a protocol wherein two users u_i and u_j belonging to a universe of users $\mathcal{U}$ authenticate if two conditions are satisfied: (i) u_i has a Credential for the same property p_* for which u_j has a Matching Reference; and (ii) u_j has a Credential for the same property $p_\circ$ for which u_i has a Matching Reference.*

The introduction of Secret Handshake with Dynamic Matching in Definition 9 requires to revisit the concept of fairness in Secret Handshakes introduced in Definition 7; in particular we need to rephrase the predicate $\mathfrak{P}$ as follows:

$\mathfrak{P}$:= "*both participants to the Secret Handshake protocol possess Credentials for the property object of the other's Matching Reference* "

The concept of Dynamic Matching has been introduced in [3] by Ateniese and colleagues; however the earlier work of Castelluccia *et al.* [9] already gave the same ability to users, although the fact has not been stressed by the authors in their paper. The protocol of Ateniese and colleagues [3] is compliant with Definition 9. The protocol is separable: users receive Credentials from the certification authority – which retains the proof control – whereas users can freely create Matching References without the intervention of the CA; thus, verification control is in the hands of users. The protocol is innovative also because it is the first one supporting reusable Credentials and guaranteeing Anonymity and Unlinkability of users and of properties.

2.3 Secret Handshake with Dynamic Controlled Matching

In [17], we have introduced the concept of *Dynamic Controlled Matching*. Dynamic Controlled Matching draws its motivation from the observation that among the two schemes supporting separable Credentials, namely the work of Castelluccia *et al.* [9] and the work of Ateniese *et al.* [3], none allows the CA to maintain verification control; indeed in both schemes, the user has the freedom of choosing the property to be matched from the other party, and the CA can exercise no control over it.

In Secret Handshake schemes that support 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 verify which Credentials. However verification is dynamic, in that it is not restricted to a single, common property, as opposed to the approaches suggested in [4, 13, 16, 19, 21].

It is important also to notice that Secret Handshake with Dynamic Controlled Matching is a generalization of both classic Secret Handshake and Secret Handshake with Dynamic Matching; in order to create classic Secret Handshakes the CA can grant a Matching Reference to a user only if the latter has the corresponding Credential. This way, users are only allowed to execute successful Secret Handshake proving and verifying possession of a common property. Conversely, in order to create Secret Handshakes with Dynamic Matching, the CA can grant Matching References for every property. This way, users can choose autonomously the Matching Reference to use upon each Secret Handshake, thus effectively keeping control over verification.

3 The Scheme

In this Section we introduce a modification of the scheme that we have presented in [18]; within our new, modified protocol, Credentials are distributed by multiple, independent CAs that trust one another but still want to maintain the control over properties falling in their realm. The choice of extending this scheme in particular is that, as can be deduced from what discussed in the previous Section, it supports separability, reusable credentials and revocation. In addition, the CA retains proof and verification control, thus allowing for the more generic concept of Dynamic Controlled Matching.

Within a multiple CA scenario, a handshake between two users A and B can be successful if A has a Credential for property p_1 issued from CA_1 and a Matching Reference for property p_2 issued from CA_1 and B has a Credential for property p_2 issued from CA_1 and a Matching Reference for property p_1 issued from CA_1. However a handshake can be successful even in hybrid situations in which for instance A has a Credential for property p_1 issued from CA_1 and a Matching Reference for property p_2 issued from CA_2 and B has a Credential for property p_2 issued from CA_2 and a Matching Reference for property p_1 issued from CA_1.

3.1 Description of the Scheme

In this Section we introduce the Secret Handshake scheme. The active parties in the scheme are essentially users and a number of mistrusting entities that we will call certification authority (CA). The various CAs jointly engage in the CASetup algorithm to generate the common public and secret parameters. Each CA then executes independently the Setup algorithm to generate public and secret parameters for the single CA.

Users then receive from given CAs Credentials and Matching References for a given property. In case of compromised Credentials, the CA adds a value called Revocation Handle to a publicly available revocation list: this way, verifiers may refuse to interact with users bearing revoked Credentials.

At first, let us describe the notation used in the sequel of the Chapter. Given a security parameter k, let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of order q for some large prime q, where the bitsize of q is determined by the security parameter k. Our scheme uses a computable, non-degenerate bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ for which the *Symmetric External Diffie-Hellman (SXDH)* problem is assumed to be hard. The SXDH assumption in short allows for the existence of a bilinear pairing, but assumes that the Decisional Diffie-Hellman problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$ (see [3] for more details).

Next, we describe how we represent strings into group elements. Following [7, 20], let $g \xleftarrow{R} \mathbb{G}_1$; let us also choose $n+1$ random values $\{y_i\}_{i=0}^{n} \xleftarrow{R} \mathbb{Z}_q^*$; we assign $g_0 = g^{y_0}, g_1 = g^{y_1}, \ldots, g_n = g^{y_n}$. If $v \in \{0,1\}^n$ is an n-bit string, let us define $h(v) = y_0 + \sum_{i \in V(v)} y_i$, where $V(v)$ represents the set of indexes i for which the i-th bit of v is equal to 1. We also define $H(v) = g_0 \prod_{i \in V(v)} g_i = g^{h(v)} \in \mathbb{G}_1$. The scheme is composed of the following algorithms:

- CASetup this algorithm corresponds to the general setup of the system, to which all CAs participate; according to the security parameter k, $g, \tilde{g}$ are selected, where g and $\tilde{g}$ are random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Then the values $W = g^w$ and $\tilde{g}^{w^{-1}}$ are chosen, so that the value w is unknown to all CAs[1]. Then, values $\{y_i\}_{i=0}^{n} \xleftarrow{R} \mathbb{Z}_q^*$ are randomly drawn and assigned as $g_0 \leftarrow g^{y_0}, g_1 \leftarrow g^{y_1}, \ldots, g_n \leftarrow g^{y_n}$. Notice that with these parameters, $H(p)$ is computed as $g_0 \prod_{i \in V(p)} g_i = g^{h(p)}$. The system's parameters are $\{q, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, W, g_0, \ldots, g_n\}$; the values $y_0, \ldots, y_n$ and $\tilde{g}^{w^{-1}}$ are kept secret among the CAs;
- Setup this algorithm corresponds to the setup of a single CA; upon execution of this algorithm, the CA picks $t_{CA} \in \mathbb{Z}_q^*$ and publishes $T_{CA} = \tilde{g}^{t_{CA}}$; finally, the CA maintains its own function $f_{CA}(p)$; f_{CA} is implemented maintaining a list of pairs $(p \in \mathcal{P}, f_{CA}(p) \in \mathbb{Z}_q^*)$, which is filled as follows: if p is not in the list, the CA picks a random number $r \in \mathbb{Z}_q^*$ and inserts the pair (p, r) in the list. If p is already in the list, the CA looks up the pair (p, r) and sets $f_{CA}(p) = r$;
- Certify this algorithm is executed by a given CA when a user $u \in \mathcal{U}$ queries that CA for a Credential for property $p \in \mathcal{P}$; if p falls within the set of properties that the queried CA is responsible for, the queried CA verifies that the supplicant user $u \in \mathcal{U}$ possesses the property $p \in \mathcal{P}$; after a successful check, the CA issues to u the appropriate Credential, which is made of two separate components: an Identification Handle, later used for revocation, and the actual Credential. To hand out the Identification Handle for a given pair (u, p), the CA picks the Identification Handle $x_{u,p} \xleftarrow{R} \mathbb{Z}_q^*$, randomly drawn upon each query, and gives it to the supplicant user. The CA then forms the Credential as a tuple $cred_{u,p} = \langle C_{u,p,1}, C_{u,p,2}, C_{u,p,3} \rangle$ where $C_{u,p,1} = g^{zw(x_{u,p} + t_{CA} f_{CA}(p) h(p))}$, $C_{u,p,2} = \tilde{g}^{(zw)^{-1}}$ and $C_{u,p,3} = \tilde{g}^{z^{-1}}$, where $z \in \mathbb{Z}_q^*$ is randomly drawn upon each query. To allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{CA}(p)}$ and $\tilde{g}^{t_{CA} h(p)}$. The user first

[1] The CAs can achieve this for instance by using an external dealer or by engaging in a secure multi-party computation.

verifies that $\hat{e}\left(H(p), T_{CA}\right) = \hat{e}\left(g, \tilde{g}^{t_{CA}h(p)}\right)$; if this first verification succeeds, the user verifies that $\hat{e}(C_{u,p,1}, C_{u,p,2}) = \hat{e}(g^{x_{u,p}}, \tilde{g}) \cdot \hat{e}(g^{f_{CA}(p)}, \tilde{g}^{t_{CA}h(p)})$;

- Grant this algorithm is executed by a given CA when a user $u \in \mathcal{U}$ queries that CA for a Matching Reference for property $p \in \mathcal{P}$; the CA verifies that – according to the policies of the CA – the supplicant user is entitled to verify that another user possesses property $p \in \mathcal{P}$. If the checking is successful, the CA issues the appropriate Matching Reference $match_p = \tilde{g}^{t_{CA}f_{CA}(p)h(p)}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{CA}(p)}$ and $\tilde{g}^{t_{CA}h(p)}$. The user first verifies that $\hat{e}\left(H(p), T_{CA}\right) = \hat{e}\left(g, \tilde{g}^{t_{CA}h(p)}\right)$; if this first verification succeeds, the user verifies that $\hat{e}(g, match_p) = \hat{e}(g^{f_{CA}(p)}, \tilde{g}^{t_{CA}h(p)})$;
- Revoke if the Credential for property p of user $u \in \mathcal{U}$ is to be revoked, the CA adds the so-called *Revocation Handle* $rev_{u,p} = \tilde{g}^{x_{u,p}}$ to a publicly available revocation list L_{rev}. It is worth noting that the Identification Handle $x_{u,p}$ and the corresponding Revocation Handle $rev_{u,p} = \tilde{g}^{x_{u,p}}$ are tightly related;
- Handshake is a probabilistic polynomial-time two-party algorithm executed by two users; the algorithm is composed of four sub-algorithms:
  - Handshake.Init the user picks $m \xleftarrow{R} \mathbb{Z}_q^*$ and produces $\tilde{g}^m$;
  - Handshake.RandomizeCredentials the user picks random values $r, s \xleftarrow{R} \mathbb{Z}_q^*$; then, given the Credential $cred_{u,p} = \langle C_{u,p,1}, C_{u,p,2}, C_{u,p,3} \rangle$ and the Identification Handle $x_{u,p}$, the user produces the tuple $\left\langle g^r, (C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}, (C_{u,p,3})^{s^{-1}} \right\rangle$. The user also computes $K = \left(\hat{e}\left(g, \tilde{g}^{m'}\right)\right)^{rx_{u,p}}$, where $\tilde{g}^{m'}$ is the nonce received from the other party;
  - Handshake.CheckRevoked the user parses the tuple SH as $\left\langle g^r, (C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}, (C_{u,p,3})^{s^{-1}} \right\rangle$. The user verifies whether SH contains a revoked Credential by checking if the following identity

$$\hat{e}\left((C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}\right) = \hat{e}\left(g^r, match_p \cdot rev\right) \tag{1}$$

    is verified with any of the Revocation Handles rev in the list L_{rev}. $match_p$ is the Matching Reference the user will use when performing Handshake.Match. If the check is successful, the user discards the current handshake instance;
  - Handshake.Match the users parses SH, the handshake message received from the remote user, as $\left\langle g^r, (C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}, (C_{u,p,3})^{s^{-1}} \right\rangle$. The user checks whether

$$\hat{e}\left(g, (C_{u,p,3})^{s^{-1}}\right) = \hat{e}\left(W, (C_{u,p,2})^{s^{-1}}\right) \tag{2}$$

    and computes

$$K = \left(\frac{\hat{e}\left((C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}\right)}{\hat{e}\left(g^r, match_p\right)}\right)^m \tag{3}$$

    $match_p$ is a Matching Reference;

Let us assume that two users, Alice and Bob, want to perform a Secret Handshake and share a key if the Handshake is successful. Alice owns the tuple $\langle cred_{A,p_1}, match_{p_2}, \ x_{A,p_1} \rangle$ and Bob owns $\langle cred_{B,p_2}, match_{p_1}, \ x_{B,p_2} \rangle$. Figure 1 shows how the handshake is carried out.

Alice :	pick $r, s, m \overset{R}{\leftarrow} \mathbb{Z}_q^*$
Alice $\longrightarrow$ Bob :	$\left\langle g^r, (C_{A,p_1,1})^{rs}, (C_{A,p_1,2})^{s^{-1}}, (C_{A,p_1,3})^{s^{-1}}, \tilde{g}^m \right\rangle$
Bob :	pick $r', s', m' \overset{R}{\leftarrow} \mathbb{Z}_q^*$
Bob $\longrightarrow$ Alice :	$\left\langle g^{r'}, (C_{B,p_2,1})^{r's'}, (C_{B,p_2,2})^{s'^{-1}}, (C_{B,p_2,3})^{s'^{-1}}, \tilde{g}^{m'} \right\rangle$
Alice :	check that Equation 2 holds, otherwise abort
Alice :	check that Equation 1 is not satisfied with any $rev \in L_{rev}$, otherwise abort
Alice :	compute $K_1 = \left(\hat{e}\left(g, \tilde{g}^{m'}\right)\right)^{rx_{A,p_1}}$
Alice :	compute $K_2 = \left(\dfrac{\hat{e}\left((C_{B,p_2,1})^{r's'}, (C_{B,p_2,2})^{s'^{-1}}\right)}{\hat{e}\left(g^{r'}, match_{p_2}\right)}\right)^m$
Bob :	check that Equation 2 holds, otherwise abort
Bob :	check that Equation 1 is not satisfied with any $rev \in L_{rev}$, otherwise abort
Bob :	compute $K_1 = \left(\dfrac{\hat{e}\left((C_{A,p_1,1})^{rs}, (C_{A,p_1,2})^{s^{-1}}\right)}{\hat{e}\left(g^r, match_{p_1}\right)}\right)^{m'}$
Bob :	compute $K_2 = \left(\hat{e}\left(g, \tilde{g}^m\right)\right)^{r'x_{A,p_1}}$
Alice $\longleftrightarrow$ Bob:	mutual proof of knowledge of K_1 and K_2

Fig. 1. Secret Handshake with Dynamic Controlled Matching

At the completion of the protocol, Alice and Bob share the same keypair if and only if each user's Credential matches the other user's Matching Reference. If not, one of the two keys, or both, will be different. By requiring them to prove to one another knowledge of both keys simultaneously, either both users learn of a mutual matching, or they do not learn anything at all. In particular, they do not learn – in case of a failed handshake – if just one of the two matchings have failed, and if so which one, or if both did fail.

Let us describe a practical scenario to understand the scheme better: let us assume that two national CAs, CA_1 and CA_2, are issuing Credential and Matching References to justice enforcement officials of their respective countries. CA_1 can therefore for instance issue Credentials for *"case agent XYZ"* or *"case supervisor XYZ"*; the same can be done by CA_2. Then, if agents assigned to the same case need to cooperate on an international investigation, they can receive Matching References from the CA of the other country, making them able to run a Secret Handshake, authenticate and secure their communications.

Notice that both CAs could in principle generate Credentials for the same property *"case agent XYZ"*; however, thanks to the separate functions f_{CA} and the different values T_{CA}, none of the CAs can generate Credentials (Matching References) that would match Matching References (Credentials) associated with properties under the jurisdiction of another CA.

4 Security Analysis

This Section analyzes the security of the protocol. The proofs do not rely on random oracles, albeit the function f_{CA} can be mistaken by one: random oracles are functions that users of the system (and attackers) can compute on their own, whereas f is comparable to a master secret that changes for the different properties, whose value is given to users only to allow them to perform checks on Credentials.

It could be debatable whether or not it is opportune to hand out the value $g^{f_{CA}(p)}$ for each property at the time of the execution of Setup amongst the other public parameters, instead of giving them only upon the execution of Certify and Grant. In any case, from the security point of view, the adversary has knowledge of all these values in all the games.

The security requirements of the scheme can be effectively resumed as follows:

1. *Impersonator Resistance*: given property p_*; let us assume two users, A and B, engage in Handshake; B has a Matching Reference for p_*; then, it is computationally infeasible for A – without a non-revoked Credential for p_* – to engage in Handshake with B and output the correct key, linked to a successful proof of possession of p_* by A and a successful detection of p_* by B;
2. *Detector Resistance*: given property p_*; let us assume two users, A and B, engage in Handshake; B has a Credential for p_*; then, it is computationally infeasible for A – without the appropriate Matching Reference for p_* – to distinguish between the key A computes executing Handshake and a random value;
3. *Unlinkability of Users*: it is computationally unfeasible for a user – engaging in two executions of Handshake – to tell whether he was interacting with the same user or two different ones;
4. *Unlinkability of Properties*: it is computationally unfeasible for a user – engaging in two executions of Handshake without the appropriate Matching References – to tell whether he was interacting with users having Credentials for the same property or for different ones;

Notice that in the impersonation resistance game, the adversary is required to produce the successful key instead of requiring key indistinguishability. However we stress that the same requirement is considered for instance in the works of Balfanz *et al.* [4] and of Ateniese *et al.* [3].

Throughout our analysis, we shall consider two separate types of adversary: type I represents a common user of the system. For this type of attacker we assume that certification entities cannot be compromised: this means that the

adversary will not receive the secret system parameters $\tilde{g}^{w^{-1}}, y_0, \ldots, y_n$ and the CA-specific parameters t_{CA} and $f_{CA}(p)$.

Type II is instead represented by a malicious CA, whose objective is to successfully engage in a Secret Handshake and carry out detection and impersonation for properties under the control of another CA; this type of adversary has access to all the information available to CAs, but clearly not to CA-specific information such as t_{CA} and $f_{CA}(p)$ for other CAs.

Due to space restrictions, Lemmas and proofs establishing the security of the scheme cannot be included in this paper, and will be included in its extended version.

4.1 Security against Adversary Type I

We consider the same adversarial type as the one adopted in numerous closely-related works such as [3, 4, 9], wherein the adversary can always obtain Credentials and Matching References for properties at his will, except of course for properties being object of challenges: in particular, the adversary cannot get a Credential (resp. Matching Reference) for the property he is trying to impersonate (resp. detect); also, the adversary for Unlinkability requirements is not limited to passively observing protocol instances[2], but can actively engage in protocol instances and even receive the correct key at the end.

The adversary is allowed to access a number of oracles managed by the challenger in order to interact with the system, in particular $\mathcal{O}_{\mathsf{Setup}}$ is invoked when the adversary wants to create a new certification authority by calling Setup; $\mathcal{O}_{\mathsf{Certify}}$ is invoked when the adversary wants to receive a Credential for a given property through the execution of Certify; $\mathcal{O}_{\mathsf{Grant}}$ is invoked when the adversary wants to receive a Matching Reference for a given property through the execution of Grant; and finally $\mathcal{O}_{\mathsf{Revoke}}$ is invoked when the adversary wants to receive a Revocation Handle for a given Credential through the execution of Revoke. Each of the proofs of this Section assume that the algorithm $\mathcal{O}_{\mathsf{CASetup}}$ has already been executed by the challenger prior to the beginning of the game. Notice that this is not a limiting factor, since the adversary that we are addressing now is a simple user of the system who – we assume – has no access to the output of $\mathcal{O}_{\mathsf{CASetup}}$. This assumption will be lifted in the next Section when we consider the resilience of the scheme against a type II adversary.

Notice also that this adversarial model is weaker than the one adopted by Jarecki and colleagues in [11]. Under this model, as opposed to the one used in this and several other works [3, 4, 9], the adversary can access a $\mathcal{O}_{\mathsf{Handshake}}$ oracle through which it can initiate arbitrary concurrent Secret Handshake instances and reveal the outcome of some of them. Quoting Jarecki and colleagues, we focus our analysis only on the *"security of isolated protocol instances"*.

Unlinkability of Properties. Consider an adversary $\mathcal{A}$ whose goal is to check if two handshake tuples contain the same property. $\mathcal{A}$ can access the oracles of

[2] Adversaries trying to detect/impersonate are by nature active ones.

the system. $\mathcal{A}$ is then challenged as follows: $\mathcal{A}$ chooses a property p_* for which no call to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted; he is then given SH_1 and SH_2 generated by two calls to Matching.RandomizeCredentials and is required to return *true* if he can decide that both SH_1 and SH_2 refer to p_*. To make the adversary as powerful as possible, the challenger will also give to the adversary the key that it computes when executing Matching.RandomizeCredentials. We call this game TraceProperty.

Unlinkability of Users. Consider an adversary $\mathcal{A}$ whose goal is to check if two handshake tuples come from the same user. Let us first of all notice that there are two values that can deanonymize a user, the Identification Handle $x_{u,p}$, and z, the random number drawn at each call to Certify and used to salt the Credentials. Between the two, $x_{u,p}$ is the only one that can be traced over two different handshake tuples. Indeed, tracing the value z is impossible, since over successive handshake tuples, it always appears multiplied by a different random value.

$\mathcal{A}$ can access the oracles of the system. Eventually $\mathcal{A}$ receives two handshake tuples containing the same property, and returns true if he can decide that upon both protocol instances he was interacting with the same user. We call this game TraceUser.

There are two separate situations where we want to prove that Unlinkability of users holds: (i) where a user uses Credentials that have not yet been revoked, for which the adversary has a corresponding Matching Reference; and (ii) where the user uses Credentials that have already been revoked, in which case Unlinkability of users holds only if the adversary does not have the corresponding Matching Reference. We remind the reader that users are clearly traceable to an adversary who has both the correct Matching Reference and the Revocation Handle for that Credential.

Therefore we present two separate games, TraceUser1 and TraceUser2: the first challenges the adversary's capability to trace a non-revoked user, having the appropriate Matching Reference for the user's Credential; the second challenges the adversary's ability to trace a revoked user without the appropriate Matching Reference for the user's (revoked) Credential.

Detection Resistance. Let $\mathcal{A}$ be an adversary whose goal is to engage in Secret Handshake protocol instances and detect the other user's property, without owning the appropriate Matching Reference. We call detector resistance the resilience to such kind of an attacker. At first, the adversary can access the oracles of the system. At the end of the query phase, $\mathcal{A}$ picks a property p_* for which no call to $\mathcal{O}_{\mathsf{Grant}}$ has been made. The adversary then engages in a protocol execution with the challenger, and is asked at the end to distinguish the correct key that Handshake.Match would output with the correct Matching Reference from a random value of the same length. We call this game Detect.

Impersonation Resistance. The analysis of the impersonation resistance requirement is slightly more complex than the analysis of other requirements. Before venturing in the actual analysis, we shall give an overview of how we approach it. At first we define a broad game, called Impersonate, where the attacker has to be able to conduct a successful Secret Handshake for a given property, having access to an arbitrary number of Credentials that are revoked before the challenge phase.

Then, we create two sub-games, Impersonate1 and Impersonate2: each game is the same as Impersonate with an additional requirement that the adversary needs to satisfy. The additional requirement (namely the satisfaction of an equality) creates a clear cut between the two games, whose union generates Impersonate. Then we present two separate proofs for the hardness of the Impersonate1 and Impersonate2 games. These two games, while representing valid reductions, have the inconvenience that they are strategy-dependent since they make assumptions on the behaviour of the attacker. In order to fix this inconvenient, we show how the two reductions can be joined in a single reduction for the strategy-independent initial game Impersonate.

Let us now introduce the Impersonate game, where the attacker has to be able to conduct a successful Secret Handshake for a given property, having access to an arbitrary number of Credentials that are revoked before the challenge phase. The adversary can access the oracles of the system. $\mathcal{A}$ eventually decides that this phase of the game is over. The challenger then revokes each Credential handed out to the attacker in the previous phase. $\mathcal{A}$ then declares $p_* \in \mathcal{P}$ which will be the object of the challenge; $\mathcal{A}$ is then challenged to engage in Handshake with the challenger, and has to be able to convince that he owns a Credential for property p_*. $\mathcal{A}$ is then asked to output the key computed. In order to successfully win the game, it must not be possible for the challenger to abort the handshake due to the fact that the Credentials used by the attacker have been revoked.

We then construct two sub-games, as follows: at the end of the query phase of the Impersonate game, $\mathcal{A}$ receives a nonce $\tilde{g}^m$ and is then asked to produce the handshake tuple $\langle g^\alpha, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta \rangle$ and the key e^k computed by the algorithm Handshake.RandomizeCredentials. If the attacker is successful, the challenger should be able to compute the same key using Handshake.Match and the Matching Reference for p_*.

The challenger checks $\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, match_{p_*})}\right)^m = \left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, \tilde{g}^{t_{CA} f_{CA}(p_*) h(p_*)})}\right)^m =$ e^k and that $\hat{e}\left(g, \tilde{g}^\delta\right) = \hat{e}\left(g^w, \tilde{g}^\gamma\right)$. Let us set $\alpha = r$, $k = r x_{u_*,p_*} m$ and $\delta = s^{-1}$, for some integers $r, x_{u_*,p_*}, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, we can write $\gamma = (ws)^{-1}$ and $\beta = rsw(x_{u_*,p_*} + t_{CA} f_{CA}(p_*) h(p_*))$.

Recall that the attacker receives a number of Credentials during the query phase. The attacker can win the game in two ways: (i) forge a brand new Credential or (ii) use an old Credential yet circumventing the revocation check, notably Equation 1 of the Handshake.CheckRevoked sub-algorithm. Let us set $X_{u,p} = x_{u,p} + t_{CA} f_{CA}(p) h(p)$. When the attacker is challenged, we have seen that he produces the value $g^{rsw(x_{u_*,p_*} + t_{CA} f_{CA}(p_*) h(p_*))} = g^{rs X_{u_*,p_*}}$. If we define the set

$Q_{\mathcal{A}} = \{X_{u,p} \in \mathbb{Z}_q^* : \mathcal{A}$ *has received* $g^{zwX_{u,p}}, \tilde{g}^{(zw)^{-1}}, \tilde{g}^{z^{-1}}$ *from a* Certify *query*$\}$, then (i) implies $X_{u_*,p_*} \notin Q_{\mathcal{A}}$ and (ii) implies $X_{u_*,p_*} \in Q_{\mathcal{A}}$. X_{u_*,p_*} is the value the attacker uses in the challenge handshake instance. We then define two different games: Impersonate1, the aforementioned Impersonate game when $X_{u_*,p_*} \notin Q_{\mathcal{A}}$, and Impersonate2 when $X_{u_*,p_*} \in Q_{\mathcal{A}}$.

4.2 Security against Adversary Type II

In this Section we focus on colluding CAs, whose purpose is to engage in a successful Secret Handshake carrying out a successful detection or impersonation of a property under the control of another target CA_*, without owning the appropriate Credential or Matching Reference. In the rest of this Section we will tackle the analysis of the security against this other type of adversary, by presenting two games, CAImpersonate and CADetect, similar to the aforementioned Impersonate and Detect games, with the difference that the adversary is now another CA; the adversary then also obtains the values $\tilde{g}^{w^{-1}}$ and $y_0, \ldots, y_n$. In particular, we give the adversary the ability to invoke the $\mathcal{O}_{\mathsf{CASetup}}$ oracle and receive its output: the adversary is therefore free to either generate and maintain its own CAs, or to invoke the $\mathcal{O}_{\mathsf{Setup}}$ oracle and have the challenger generate a CA under its control. The adversary will eventually attempt at impersonation or detection of a property under the control of the CA controlled by the challenger.

CA Detection Resistance. Let $\mathcal{A}$ be a malicious CA whose goal is to use the advantage held in the role of CA to engage in Secret Handshake protocol instances and attempt at the detection of a property whose Matching References are issued by another CA, without owning the appropriate Matching Reference. We call CA detector resistance the resilience to this type of attacker. We assume – with no loss in generality – that there are only two CAs in the system, the adversary and the one simulated by the challenger.

At first, $\mathcal{A}$ can access the oracles of the system, including $\mathcal{O}_{\mathsf{CASetup}}$. At the end of the query phase, $\mathcal{A}$ decides a property p_*, under the control of the CA simulated by the challenger, for which no call to $\mathcal{O}_{\mathsf{Grant}}$ has been made. $\mathcal{A}$ is then challenged to engage in a protocol execution with the challenger, and asked at the end to distinguish the correct key that Handshake.Match would output with the correct Matching Reference from a random value of the same length. We call this game CADetect.

CA Impersonation Resistance. To address the analysis of this requirement, we follow the same strategy adopted in Section 4.1; in particular, we define two sub-games, CAImpersonate1 and CAImpersonate2 and then join them together under a broader CAImpersonate game.

Let $\mathcal{A}$ be a malicious CA whose goal is the impersonation of a user owning a Credential for a given property, under the control of another CA. $\mathcal{A}$ can access $\mathcal{A}$ can access the oracles of the system, including $\mathcal{O}_{\mathsf{CASetup}}$. We assume – with no loss in generality – that there are only two CAs in the system, the adversary

and the one simulated by the challenger. $\mathcal{A}$ eventually decides that this phase of the game is over. The challenger then revokes each Credential handed out to the attacker in the previous phase. $\mathcal{A}$ then declares a property $p_* \in \mathcal{P}$ under the control of the CA simulated by the challenger, which will be the object of the challenge; the adversary $\mathcal{A}$ is then challenged to engage in Handshake with the challenger, and has to be able to convince that he owns a non-revoked Credential for property p_*. $\mathcal{A}$ is then asked to output the key computed. In order to successfully win the game, it must not be possible for the challenger to abort the handshake due to the fact that the Credentials used by the attacker have been revoked. We call this game CAImpersonate.

5 Conclusion

The focus of this paper has been the study of Secret Handshakes that do not rely on a centralized certification entity; to this end, we have presented the first scheme whereby a coalition of multiple, independent CAs can associate: each CA maintains proof and verification control over the properties falling under its realm. Users can conduct successful Secret Handshakes even in hybrid scenarios, with Credentials and Matching References from different CAs. The scheme supports Secret Handshake with Dynamic Controlled Matching and allows for revocation of Credentials. We have studied the security of the scheme through game-based security without relying on random oracles.

References

1. Wikipedia, the free encyclopedia (2010), `http://www.wikipedia.org`
2. Asokan, N.: Fairness in electronic commerce. PhD thesis, Waterloo, Ont., Canada (1998)
3. Ateniese, G., Kirsch, J., Blanton, M.: Secret handshakes with dynamic and fuzzy matching. In: NDSS (2007)
4. Balfanz, D., Durfee, G., Shankar, N., Smetters, D.K., Staddon, J., Wong, H.-C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy, pp. 180–196 (2003)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: ACM Conference on Computer and Communications Security, pp. 244–250 (1993)
7. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
8. Boyko, V., MacKenzie, P.D., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)

9. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret handshakes from ca-oblivious encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
10. Jarecki, S., Kim, J., Tsudik, G.: Authentication for paranoids: Multi-party secret handshakes. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 325–339. Springer, Heidelberg (2006)
11. Jarecki, S., Kim, J., Tsudik, G.: Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 352–369. Springer, Heidelberg (2008)
12. Jarecki, S., Liu, X.: Unlinkable secret handshakes and key-private group key management schemes. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 270–287. Springer, Heidelberg (2007)
13. Meadows, C.: A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In: IEEE Symposium on Security and Privacy, pp. 134–137 (1986)
14. Pagnia, H., Gärtner, F.C.: On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology (March 1999)
15. Pfitzmann, A., Hansen, M.: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology, v0.31 (February 2008), `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`
16. Shin, J.S., Gligor, V.D.: A new privacy-enhanced matchmaking protocol. In: NDSS (2008)
17. Sorniotti, A., Molva, R.: A provably secure secret handshake with dynamic controlled matching. In: SEC, pp. 330–341 (2009)
18. Sorniotti, A., Molva, R.: Secret handshakes with revocation support. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 274–299. Springer, Heidelberg (2010)
19. Vergnaud, D.: Rsa-based secret handshakes. In: Ytrehus, Ø. (ed.) WCC 2005. LNCS, vol. 3969, pp. 252–274. Springer, Heidelberg (2006)
20. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
21. Xu, S., Yung, M.: k-anonymous secret handshakes with reusable credentials. In: ACM Conference on Computer and Communications Security, pp. 158–167 (2004)