Thèse

présentée pour obtenir le grade de docteur

de TELECOM ParisTech

Spécialité : Informatique et Réseaux

Alessandro SORNIOTTI

Protocoles pour Poignées de Main Secrètes

Soutenue le 29 Juin 2010 devant le jury composé de

PhD Thesis

TELECOM ParisTech

Computer Science and Networks

Alessandro SORNIOTTI

Secret Handshake Protocols

Defense date: June 29th, 2010. Committee in charge:

Pascal Urien	Chairman
Giuseppe Ateniese	Reporter
Claude Castelluccia	Reporter
Jan Camenisch	Examiner
Volkmar Lotz	Examiner
Refik Molva	Advisor

Ai miei genitori, a cui devo tutto.
To my parents, to whom I owe everything.
A mes parents, à qui je dois tout.

Acknowledgements

I would like to start off by thanking my parents: as the dedication of this manuscript reads, I owe them everything. They have taught me, and are teaching me a lot every day, they have raised me with the perfect mixture of strictness and love, I have always felt protected and challenged. It is safe to say that they have played a significant part in all the good things in my life. For the bad ones, I proud myself to having done everything on my own.

My gratitude and love goes to Nancy, the greatest gift that these three years have given me. She has been very patient with me, with my mood swings when things seemed to go wrong, with my temper when proofs insisted on telling me that a scheme was flawed.

I am profoundly grateful to prof. Refik Molva. It is through his lectures that I have become interested in security and research. He has been an extraordinary mentor, always available to discuss, to explain, to brainstorm, to help me. I have learned so much from him and this thesis would not have been possible without him.

Thanks to my friends, Matteo and Chiara, Daniele, Frédéric and Jean-François. During these past three years, we have shared our experiences with research, our frustrations and results. More importantly, we have partied a lot and although we all live in different parts of the world, we have always managed to stay in touch as good friends do. Thanks to my "local" friends, Michael and Tünde, Luca and Cécile, Gerald and Julia, Stuart: we have had a lot of fun together, which has considerably lightened up these three years of hard work. Moreover, thanks to you guys I have discovered the beauty of running, and managed to run 42.195 kilometres on a cold November morning.

I would like to thank my colleagues at SAP, in particular Konrad, Annett, Laurent, Luca, Florian, Volkmar, JC, Cédric, Slim: thank you all for your support and especially for creating a good working environment.

Thanks to the staff at EURECOM, in particular to Gwenäelle for helping me and being really cool.

Finally, thanks to prof. Pascal Urien for agreeing to chair the defense committee. Thanks to prof. Giuseppe Ateniese, prof. Claude Castelluccia, dr. Jan Camenisch and Volkmar Lotz for agreeing to be reporters and examiners.

Abstract

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. The mechanism of Secret Handshakes tackles this problem, providing a solution for secure initial exchange between mistrusting principals. A Secret Handshake is a protocol that allows two users to mutually verify one another's properties, and in case of simultaneous matching, to share a key used to secure subsequent communications. The protocol assures that an outsider, or an illegitimate group member, does not learn anything by interacting with a legitimate user or by eavesdropping on protocol exchanges.

In this thesis, we present several novel protocols, aimed at providing new features or at fixing shortcomings of existing protocols in the literature. At first, we focus on a new concept of Secret Handshake, called Dynamic Controlled Matching, generalizing other Secret Handshake variants. We then address the challenging task of revocation in Secret Handshakes, presenting an approach through which we can achieve revocation for each of the different variants of Secret Handshake known in the literature. Furthermore, we study two decentralized Secret Handshake protocols, one where a number of separate mistrusting entities can federate to create a Secret Handshake scheme and another one where the scheme is self-managed by its users. Finally we investigate two use-cases for Secret Handshake protocols, the first involving online social networks and the second addressing supply chain management.

Résumé

Les utilisateurs qui coopèrent sur des réseaux non fiables ont souvent besoin d'établir une confiance initiale; cette étape peut etre très délicate si elle implique l'échange d'informations sensibles, telles l'affiliation à une société secrète ou à des services de renseignement. Le mécanisme de la poignée de main secrète fournit une solution à ce problème. La poignée de main secrète est un protocole qui permet à deux utilisateurs de vérifier mutuellement des propriétés et, en cas de correspondance simultanée, de partager une clé pour sécuriser les communications. Le protocole assure qu'aucune entité malveillante n'apprenne quelque-chose en interagissant avec des utilisateurs légitimes ou en écoutant les échanges du protocole.

Dans cette thèse, nous présentons plusieurs nouveaux protocoles visant à fournir de nouvelles fonctionnalités ou à fixer les lacunes des protocoles existants dans la littérature. Dans un premier temps, nous nous concentrons sur un nouveau concept de poignée de main secrète appelé vérification dynamique contrôlée, qui généralise les autres variantes du protocole. Nous étudions ensuite le problème de la révocation des titres pour la poignée de main secrète, en présentant une approche grâce à laquelle nous pouvons obtenir la révocation pour chacune des différentes variantes du protocole connues dans la littérature. En outre, nous étudions deux protocoles de poignées de main secrète décentralisés: l'un où un certain nombre d'entités distinctes peut gérer le protocole, et l'autre où le système est auto-géré par ses utilisateurs. Enfin, nous étudions deux cas d'utilisation de ce protocole, le premier impliquant des réseaux sociaux en ligne et le second couvrant la gestion d'une chaîne d'approvisionnement.

Table of Contents

List of Figures

List of Tables

List of Publications

[GLR$^+$08] Laurent Gomez, Annett Laube, Vincent Ribière, Alessandro Sorniotti, Christophe Trefois, Marco Valente, and Patrick Wetterwald. Encryption-based access control for building management. In *MobiQuitous*, 2008.

[GLS08] Laurent Gomez, Annett Laube, and Alessandro Sorniotti. Design guidelines for integration of wireless sensor networks with enterprise systems. In *MOBILWARE*, page 12, 2008.

[GLS09] Laurent Gomez, Annett Laube, and Alessandro Sorniotti. Trustworthiness assessment of wireless sensor data for business applications. In *AINA*, pages 355–362, 2009.

[KS09] Florian Kerschbaum and Alessandro Sorniotti. Rfid-based supply chain partner authentication and key agreement. In *WISEC*, pages 41–50, 2009.

[KS10] Florian Kerschbaum and Alessandro Sorniotti. Extending searchable encryption for outsourced data analytics. In *ACNS*, 2010.

[SEKG$^+$09] Alessandro Sorniotti, Paul El Khoury, Laurent Gomez, Anjel Cuevas, and Annett Laube. A security pattern for untraceable secret handshakes. In *SECURWARE 2009, 3rd International Conference on Emerging Security Information, Systems and Technologies, June 18-23, 2009, Athens/Glyfada, Greece*, 06 2009.

[SGWO07] Alessandro Sorniotti, Laurent Gomez, Konrad Wrona, and Lorenzo Odorico. Secure and trusted in-network data processing in wireless sensor networks: a survey. *JIAS, Journal of Information Assurance and Security, Volume 2, Issue 3 (Special Issue), September 2007*, 2007.

[SM09a] Alessandro Sorniotti and Refik Molva. A provably secure secret handshake with dynamic controlled matching. In *SEC*, pages 330–341, 2009.

[SM09b] Alessandro Sorniotti and Refik Molva. A provably secure secret handshake with dynamic controlled matching. *Computers & Security, Elsevier*, In Press, Corrected Proof, 2009.

[SM09c] Alessandro Sorniotti and Refik Molva. Secret handshakes with revocation support. In *ICISC 2009, 12th International Conference on Information Security and Cryptology, December 2-4, 2009, Seoul, Korea*, 12 2009.

[SM10] Alessandro Sorniotti and Refik Molva. Secret interest groups (sigs) in social networks with an implementation on facebook. In *SAC*, pages 621–628, 2010.

[SMG08] Alessandro Sorniotti, Refik Molva, and Laurent Gomez. Efficient access control for wireless sensor data. In *PIMRC*, pages 1–5, 2008.

[SMG09a] Alessandro Sorniotti, Refik Molva, and Laurent Gomez. Efficient access control for wireless sensor data. *Ad Hoc & Sensor Wireless Networks*, 7(3-4):325–336, 2009.

[SMG+09b] Alessandro Sorniotti, Refik Molva, Laurent Gomez, Christophe Trefois, Annett Laube, and Piervito Scaglioso. Efficient access control for wireless sensor data. *International Journal of Wireless Information Networks, Springer, Vol.16 N°3, September 2009*, 2009.

[TWS+10] Slim Trabelsi, Eric Weil, Alessandro Sorniotti, Stuart Short, and Michele Bezzi. Privacy-aware policy matching. In *ISCC*, 2010.

Chapter 1

Introduction

The motivation for the work of this thesis originates from an analysis of the use-cases presented by the CoBIs European project in [Cob07]. In one scenario, drums containing chemicals are stocked in a warehouse; however, safety regulations impose restrictions on the dispositions of these barrels. For instance, barrels containing reactive chemicals cannot be stored close to each other: a small leak of the chemicals could have potentially disastrous consequences.

The devised solution included equipping each barrel with a wireless device. Each device was exchanging information about the content of its associated barrel in cleartext: this allowed to perform some inference on the disposition of the barrels and take countermeasures in case of forbidden combinations.

The risk assessment of such scenario showed many shortcomings: the matching of complementary chemicals was possible only due to the fact that the content of the barrels was broadcast in cleartext. The fact that transmissions were in cleartext however may have led to attacks linked to terrorism or to industrial espionage.

Many cryptographic solutions can be thought of, or are existing, to address this simple matching problem: a study of possible solutions shows that they are very similar and yet they present subtle differences that in the end create completely different protocols, achieving greatly different results.

The focus of this thesis is therefore an analysis of the family of these protocols, called Secret Handshake.

1.1 Secret Handshake

A *Secret Handshake* is a distinct form of greeting which conveys membership in club, group or fraternity [wik10]. 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 computer science should capture the essence of Secret Handshakes and model it into protocols that can be automatically executed by electronic devices. In particular, given the secret and sensitive nature of the scenarios motivating these protocols – namely secret groups and concealed fraternities – ought to become cryptographic protocols, accounting for the existence of misbehaving users and attackers.

1.1.1 Scenarios

In this Section we justify Secret Handshakes by presenting a broad range of different scenarios where these protocols may be required.

Consider a secret agent on a mission, needing to authenticate to a fellow agent or to a server belonging to the agency. Agents are bound to follow the agency's policy never to disclose their Credentials, unless they are certain to be dealing with fellow agents or with agency's servers. The same policy is applied by the servers too. The interesting consequence of the interactions of individuals following such policies is called a policy deadlock: none of the agents will accept to reveal his Credential first, so the communication comes to a standstill.

Let us now turn our attention to a user, Alice, who lives in a country with a questionable human-rights record. She is a member of a pro-democracy movement. Members periodically gather at secret meetings, where Alice often meets new alleged members, whom she has never met before. Consequently she is worried that she might be dealing with members of the secret police of that state, whose aim is to round up members of the pro-democracy movement and arrest them. Nonetheless, legitimate members need to interact with one another in order to carry on the activities of the movement.

Consider now justice forces of a federation of states, needing to cooperate with one another in order to solve cross-boundary criminal cases. Regulations of the federation define official processes that must imperatively be followed by operating officers: in particular, these processes mandate which institutions must cooperate upon each particular case. For instance, a member of an agency of one state must cooperate with a member of the corresponding agency of another state, 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.

Imagine now a newly formed project consortium whose members want to securely add one another as friends on a social network and use the social network infrastructure as a collaboration tool. The consortium members require means to secure the friendship invitation process: this helps to avoid false negatives, refusing a request from a legitimate consortium member, or false positives, accepting the invitation from a rogue user and consequently interacting with it. In addition, the project consortium may require additional security for fear of industrial espionage.

These different scenarios share some common requirements: in each of these examples, users are interested in conducting an authentication protocol and willing to disclose their allegiance, provided that this happens only when they are dealing with the intended remote party, a secret agent in the first example, a member of the pro-democracy movement in the second, a justice official of the intended state in the third and a consortium member in the last. If these conditions do not apply, users require that no information is leaked on their actual membership, except that the matching is not successful.

1.1.2 Characteristics

The design space of Secret Handshake protocols is governed by two main dimensions. The first and most prominent one is the dimension of security requirements; security requirements aim at giving assurance that Secret Handshake can be used in the presence of adversaries, whose aim is to subvert its operations and increase their return from its execution. Common security requirements include resistance to openly subverting the protocol: for instance, with reference to the first example of the previous Section, somebody who is not a member of the agency must not be able to authenticate as one. Additional security requirements address the amount of information that is leaked by each protocol instance upon unsuccessful executions, or that is leaked to passive adversaries whose aim is to read a number of protocol transcripts and derive information about the nature of users executing them: the information being derived may be the identity of the user, its membership or even the mere information that the same user or members of the same agency have executed the protocol twice.

The second dimension relates to functional requirements of the protocol: for instance, these requirements encompass considerations on the existence of a central authority[1] and its role in the protocol. Indeed the protocol can either be managed by a central authority, in charge of creating cryptographic tokens that allow users to conduct the authentication; as an alternative, the protocol can be self-managed by user, without the requirement for a central authority. If a central authority is required, it can exercise variable degrees of control over the capabilities of users, which represents as well another design choice. Another example of functional requirement addresses the actual type of cryptographic token(s) required for the execution of the protocol, as to whether a single type of token is sufficient, whether it can be reused and whether it can be self-generated or not.

Further requirements mix functional and security aspects, such as – referring to the same example as before – the support for disqualification from the status of agency member, also known as revocation.

1.1.3 Contributions and Organization

This thesis makes several contributions:

- We perform a detailed analysis of the literature on Secret Handshakes; we build a taxonomy of the Secret Handshake protocol family, examining gradually more complex protocols and introducing new features; at the end of this analysis we present a snapshot of the state-of-the art and we highlight a number of missing features;

[1]In the sequel of the manuscript, we shall refer to this actor as certification authority (CA) or certification entity (CE).

- We suggest six new protocols, each of which contributes to completing the spectrum of available techniques in the field. In particular we focus on:
  - a new concept of Secret Handshake whereby the aforementioned central authority exercises a strong control over the capabilities of users;
  - bringing revocation support to a number of schemes that either completely lacked this support or could only cover a limited range of scenarios;
  - decentralized Secret Handshake schemes, with a scheme that requires no central authority and a second one where several independent authorities can federate but maintain their independence;
- We present a first use-case where the techniques discussed in this thesis can be leveraged by users of a social network to spontaneously create a secret group and subsequently secure their interactions on the social network; we design a framework that achieves this objective, and implement part of it in the ever growing Facebook platform [fac]; we also discuss some of the implementation challenges;
- We present a second use-case where companies that are members of a common supply chain can exchange batches of goods equipped with RFID tags; each partner can then perform a Secret Handshake-like authentication with another partner to authenticate on the grounds of having handled a common tag at some point during the life-cycle of the supply chain;

Each new protocol comes with a detailed security analysis, conducted using state-of-the-art techniques. The analyses show mastery of the complex art of proving security protocols, since, as it shall be seen in the sequel of this document, in some occasion they require turning to advanced techniques or resorting on complex approaches.

The reminder of this manuscript is split into two parts. The first part contains contributions in terms of new cryptographic protocols, generic building blocks that can be used to address several Secret Handshake scenarios. This part of the thesis starts with Chapter 2, that introduces the context of the work, through an informal introduction to the discipline of cryptography, to cryptographic protocols, elliptic-curve cryptography and bilinear pairings, some of the foundations of our work. The subsequent Chapter,

Chapter 3 presents a detailed analysis of the state of the art of the Secret Handshake protocol family; through a critical study of the works in the literature, we are able to present a taxonomy of the various protocols and highlight a number of missing features. In the following Chapters we build up on this initial study and we focus on the design of protocols representing solutions for the highlighted gaps; in particular Chapter 4 presents a new type of Secret Handshake with a different role of the certification authority; Chapter 5 focuses on the challenging problem of revocation for Secret Handshake protocols; finally Chapter 6 constitutes a first effort toward decentralized Secret Handshake schemes.

The second part of this manuscript presents real-world use-cases where the protocols presented in the previous part serve as practical cryptographic solutions; in particular, in Chapter 7 we introduce an ad-hoc framework that can be used to create secret user groups and use such groups to secure the interaction of users in online social networks; Chapter 8 instead presents a solution for an authentication protocol between partners of a supply chain.

Finally, Chapter 9 presents the global conclusions of the manuscript and gives an outline of the future work.

Part I

Cryptographic Protocols

Chapter 2

Preliminaries

2.1 Introduction

In this Chapter we give the reader an overview of the fields of Security and Cryptography. At first we define what a cryptographic protocol is; then we discuss how it can proved to be secure, referring to the common approach of game-based security proofs and related security models such as the random oracle model and the generic group model. Finally we introduce elliptic curve cryptography and bilinear pairings as most protocols presented in this thesis will leverage on them.

2.2 Security and Cryptography

The definition of the science of *Information Security* can only be done under the assumption that an information system has some "enemies", a set of users whose objectives are to misuse it or disrupt its standard operations. Under this assumption, we can define information security as the branch of computer science that deals with the protection of information and information systems from a number of unauthorized, malicious or accidental uses. By extension, information security is also the science that studies the means of detecting and preventing such unauthorized, malicious or accidental uses.

A number of requirements have historically been identified as *confidentiality*, requiring information to be disclosed only to the intended receiver; *integrity*, detecting and preventing malicious or accidental modifications of information or of an information system; *availability* referring to the degree to which an information system is in an operating state; *authentication* dealing with the corroboration of the identity of an actor of an information system, such as the identity of the sender of a message.

Under certain circumstances, some of the aforementioned requirements can be met without recurring to cryptography: for instance, confidentiality and integrity of the transmission of a message can be ensured by building a dedicated transmission line physically secured against any tampering. However, under most circumstances, cryptographic solutions are required.

According to Menezes and colleagues [MvOV96], cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication.

2.2.1 Cryptographic Protocols

Over the centuries, an elaborate set of protocols and mechanisms has been created to deal with information security issues ranging from the simple encryption schemes devised in roman times up to the most recent and sophisticated encryption and authentication schemes.

In order to understand what a cryptographic protocol is, let us first focus on *algorithms*: an algorithm is a finite set of instructions whose execution attempts at the solution of a problem.

A *cryptographic protocol* is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective.

To define a protocol, at first the actors taking part in its execution need to be defined. Each actor performs a particular algorithm in response to an event such as the receipt of a message, or to trigger the execution of the protocol or part of it.

In the design of a cryptographic protocol, usually assumptions are made on the behavior of actors; some actors faithfully execute the protocol as it is their interest to assure the fulfillment of the security objective at stake; some others may abide by the rules of the protocols and carry out additional actions to increase their return from the execution of the protocol, for instance to learn additional information.

Every security protocol has by definition a set of adversaries. Adversaries are actors of the protocol, who know the algorithms composing it and may execute parts of them or interact with other actors. However, adversaries do not necessarily abide by the rules of the protocol (i.e. apply faithfully its algorithms): adversary are only constrained by time and complexity of algorithms.

2.3 Provable Security

To be usable in practice, any security protocol needs evidence that it can actually fulfill the security requirements that it promises. To this end, security proofs are usually provided to support a newly proposed protocol.

The proofs usually require assumptions on the capabilities of the attacker, especially the type of information accessible to him; for instance, an attacker may be able to not only execute the various algorithms of the protocol, but also to access some secret information, that is otherwise unaccessible to standard users.

Depending on the considered assumptions, one can define a more or less powerful attacker; if a protocol is proved secure against a very powerful attacker, it implies that its security is guaranteed in practice even if the rules of the protocol are not completely respected.

For instance, an encryption protocol may include a “Setup” algorithm wherein a set of secret parameters is generated. These parameters are supposed to be known in practice only by a specific actor, a “Key Generation Centre”. Nonetheless, the protocol may be proved secure even in presence of an attacker who is able to disclose some of this secret information.

In the early days of modern cryptography, a new scheme would be equipped with an informal security analysis, where the author(s) of the scheme would underline the unfeasibility of selected attacks due to the hardness of a particular operation required to perpetrate them.

One drawback of this approach is that it makes assumptions on the strategy followed by the attacker to mount an attack; this approach is limited, since the proof does not guarantee that the attack is not feasible, but shows that a particular strategy to mount it is. This approach clearly does not give any guarantee against different strategies to achieve the same attack: it may be sufficient for simple protocols, but for more complex ones, some unforeseen attacks can still lay undetected.

2.3.1 Game-based Security

To overcome the limits expressed in the previous Section, the notion of game-based security has been established in the cryptographic community. Essentially, this approach consists of the following steps: at first, a generic security property is defined, together with the existence of an *adversary* A, whose objective is to break the security property.

The adversary however does not interact directly with real instances of the protocol actors; instead a *challenger* C simulates internally all other actors. The adversary can interact with the simulated actors through so-called *oracles*. Usually there is one-to-one mapping between the algorithms defining the protocol and the oracles available for query to the adversary. Additional oracles can also be available to model additional capabilities of the attacker, such as compromising legitimate users or disclosing some otherwise secret material. The more oracles are available to the adversary, the more powerful the adversary is. As a consequence, if the protocol is proved secure against a powerful adversary, it clearly enjoys a stronger security.

After defining the oracles, the challenger sets rules on how the adversary can interact with them: usually, a set of *phases* is defined. Phases normally include a query phase, wherein the adversary can freely interact with the oracles; a challenge phase, whereby the adversary is asked to perform a particular action (send a message, give an answer) possibly subject to restrictions linked with the queries of the previous phase, in order to avoid a trivial success of the attacker; this phase can be followed by another round of queries by the attacker.

A *security game* is identified by the aforementioned oracles and phases together with the attacker definition. Examples of security games are for instance the various games that are now well-established to prove the security of encryption schemes, such as OW-CPA, IND-CCA1 and IND-CCA2 [BDPR98].

The game is set up with the objective of showing that such an attacker, with the capabilities specified through the defined oracles, cannot exist. To this end, the challenger sets up the environment where the attacker can perform its queries and answer the oracle queries using the inputs of a problem whose solution is currently assumed to be hard, i.e. unfeasible in polynomial time. Examples of these problems are for instance the RSA problem [RSA83], the discrete logarithm problem or the Diffie-Hellman problem [DH03].

Of particular importance for this manuscript is the Decisional Diffie-Hellman problem, that we hereby state:

Definition 1 (Hardness of the *Decisional Diffie-Hellman* Problem)**.** *We say that the Decisional Diffie-Hellman Problem (DDH) is hard if, for all probabilistic, polynomial-time algorithms $\mathcal{B}$,*

$$\mathsf{AdvDDH}_B := Pr[\mathcal{B}(g, g^a, g^b, g^x) = \top \text{ } \textit{if } x = ab] - \tfrac{1}{2}$$

is negligible in the security parameter. We assume a random choice of g, a, b; x is equal to ab with probability $\frac{1}{2}$ and is otherwise equal to a random value different from ab with the same probability.

Then a reduction is performed, showing that the complexity of performing the attack modeled in the game is equivalent to that of solving the hard problem. Informally, if the adversary manages to break the protocol, it is “tricked” into providing an answer to the hard problem.

The advantage of this approach is the exhaustive coverage of all possible attacks since the proof does not assume any strategy for the adversary.

2.3.2 The Random Oracle Model

Some security protocols may use cryptographic hash functions. Cryptographic hash function are usually functions defined on bitstrings of arbitrary size to bitstrings of a fixed size, say, n. These functions often also enjoy properties such as collision resistance, preimage resistance and second preimage resistance.

To prove the security of protocols using hash functions, it is often required to model them as ideal hash functions, also referred to as random oracles. Concretely, if the challenger controls an oracle simulating some hash function H used by the cryptographic scheme, then the outputs of the oracle, which are returned to the adversary, are supposed to be computationally indistinguishable from a truly random output source [Bag06].

This is modeled by replacing H with a member of the family of all truly random functions with same domain and codomain, chosen uniformly at random. When queried on the same input, the oracle must be defined to produce the same output, since the hash function will behave this way in the real world. Hence, in the random oracle model, the adversary cannot take advantage of the structure of the real hash function.

First formulated by Bellare and Rogaway in [BR93], the random oracle model is considered as a widely accepted assumption in the community. Provided that the adversary has no insight into the hash function, using this black box idealized approach to model hash functions clearly captures the security essence of the overall cryptographic scheme. Moreover, the abstraction allows designing cryptographic schemes whose efficiency cannot be achieved if the security proofs are performed without any ideal assumption.

As presented in [BR93], the random oracle model provides thus a bridge between cryptographic theory and cryptographic practice. Critics argue that no single deterministic polynomial time function can provide a good implementation of random oracles in the real world. In other words, they argue that the random oracle methodology is flawed. We refer the reader to [CGH04] for a critical look at the relationship between the security of cryptographic schemes in the random oracle model and the security of the schemes that result from implementing the random oracle by real hash functions.

2.3.3 The Generic Group Model

The generic group model is a theoretical framework for the analysis of the success of algorithms in groups where the representation of the elements reveals no information to the attacker. The most popular generic group model is the one presented by Victor Shoup [Sho97]. In this model, the attacker is not given direct access to group elements, but rather to the images of group elements under a random one-to-one mapping. The only operations the attacker can perform are therefore equality testing by a bitwise comparison on the images. Group operations can be computed by the attacker through a series of oracles. It is clear that in this situation, the attacker can gain no advantage in solving a computational problem from the representation of the group element.

The generic group is usually not adopted directly to prove the security of a protocol; instead, it is used to provide evidence as to the hardness of computational problems; for example [ACHdM05; BB08; LRSW99].

Internally, the simulator represents the elements of a group as their discrete logarithms relative to a chosen generator. To represent the images of the elements of the group for the attacker, a random one-to-one mapping is used from the considered group – say, of order q – to a codomain represented by random bitstrings of length n, such that $n = \lceil log_2 q \rceil$. For instance, the group element g^a is represented internally as a, whereas the attacker is given the external string representation $\xi_1(a)$, where ξ represents the aforementioned mapping.

The adversary communicates with the oracles using the string representation of the group elements exclusively. The proof then can show a lower-bound for the complexity of solving the particular problem at hand, under the assumption that the structure of the group does not allow for any operation other than the ones modeled through the set of oracles.

2.4 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested by Neal Koblitz [Kob87] and Victor S. Miller [Mil85]. To date, all the known algorithms to solve the discrete logarithm problem on elliptic curves are exponential. This allows, in systems using elliptic curves, to use keys of much smaller size than in the systems relying on the hardness of integer factoring or discrete logarithms in finite fields. There have been several efforts aimed at designing theoretical special purpose computers that would implement the existing attack algorithms far faster than general computing resources. Unlike the RSA and Diffie-Hellman cryptosystems, which slowly succumbed to increasingly strong attack algorithms, elliptic curve cryptography has remained at its full strength since it was first presented in 1985 and it has also been an active area of study in academia.

The US National Institute for Standards and Technology (NIST) has recommended that 1024-bit systems [1] are sufficient for use until 2010. After that, NIST recommends that they be upgraded to systems providing an increased security level. This further enhancement to public key systems can be obtained in two main ways. One option is to simply increase the public key parameter size to a level appropriate for another decade of use. Another option, the one we describe in this Section, is to take advantage of the past 30 years of public key research and analysis and move from first generation public key algorithms to elliptic curves. Calculations on this kind of curves provide a more different asymmetry between legitimate users and non-users: these calculations run indeed polynomial time for users and exponential time for non-users. The length of

[1]The majority of public key systems in use today use indeed 1024-bit parameters.

a key, in bits, for a conventional encryption algorithm is a common measure of security. One can see that, as symmetric key sizes increase, the required key sizes for RSA and Diffie-Hellman increase at a much faster rate than the required key sizes for elliptic curve cryptosystems [HMV03]. Hence, elliptic curve systems offer more security per bit increase in key size than either RSA or Diffie-Hellman public key systems.

Security is not the only attractive feature of ECC. Elliptic curve cryptosystems also are more computationally efficient than the first generation public key systems, RSA and Diffie-Hellman [GPW+04]. Although elliptic curve arithmetic is slightly more complex per bit than either RSA or DH arithmetic, the added strength per bit more than makes up for any extra compute time.

The NIST has standardized on a list of 15 elliptic curves of varying sizes. Ten of these curves are for what are known as binary fields and 5 are for prime fields. For protecting both classified and unclassified National Security information, the National Security Agency (NSA) has decided to move to elliptic curve based public key cryptography. Where appropriate, NSA plans to use the elliptic curves over finite fields with large prime moduli (256, 384, and 521 bits) published by NIST.

In conclusion, ECC provides greater security and more efficient performance than the first generation public key techniques (RSA and Diffie-Hellman) now in use. As vendors look to upgrade their systems they should seriously consider the elliptic curve alternative for the computational and bandwidth advantages they offer at comparable security.

2.4.1 ECC mathematical basis

We now propose a brief overview of the mathematical basis under Elliptic Curve Cryptography. For more details, please refer to [HMV03].

Elliptic curves for cryptography are defined over finite algebraic structures such as finite fields. Let us define F_q as a finite field of characteristic p, p being a prime and q a power of this prime. F_q will be referred to as the base group. The fields F_q^k with k a positive integer are extensions of the base group.

The general equation of an elliptic curve E over a field F_q is:

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5,$$

where the coefficients a_i belong to the field F_q. We consider only smooth curves, meaning a curve with no singular points, in other words no points where both partial derivate in x and y vanish.

We denote by $E(F_q)$ the set of pairs (x,y) such that (x,y) are solutions of the previous equation. To have the points on E to form a group, an extra point denoted by θ is included. This extra point is called the point at infinity and can be formulated as $\theta = (\infty, \infty)$.

If the characteristic of the field F_q is neither 2 nor 3, then a simple admissible change of variables allows us to simplify the elliptic curve equation to:

$$y^2 = x^3 + ax + b \text{ (}\textbf{Weierstrass form}\text{)}$$

We write the set of all such points on E as:

$$E(F_q) = \{P = (x, y) | x, y \in F_q \text{ solved from } E : y^2 = x^3 + ax + b\} \cup \{\theta\}$$

The obtained set of points form an Abelian group under the additive group operation, which is conventionally written using the notation "+".

The order of $E(F_q)$ is the number of points that lie on the elliptic curve $E(F_q)$. The order of a point on the curve $E(F_q)$ is the smallest integer m (if it exists) such that $mP = \infty$. If such an integer does not exist the point is said to have infinite order.

2.4.2 Supersingular curves

E is said to be supersingular over F_q if $E[p] = \theta$, with p being the characteristic of F_q. Concretely, this means that E has no point of order p, since θ is of order 1.

Supersingular elliptic curves are sometimes used in cryptography because computations (of pairings for example) can be made in an easier way, and determining the group order is also simpler.

2.4.3 Operations on points

We define the addition of two points in the following way:

- Let $P \in E(F_q)$. Then $P + \theta = P$ and $\theta + P = P$. So θ serves as the identity for the additive group. If $P = \theta$ then we define $-P = \theta$.

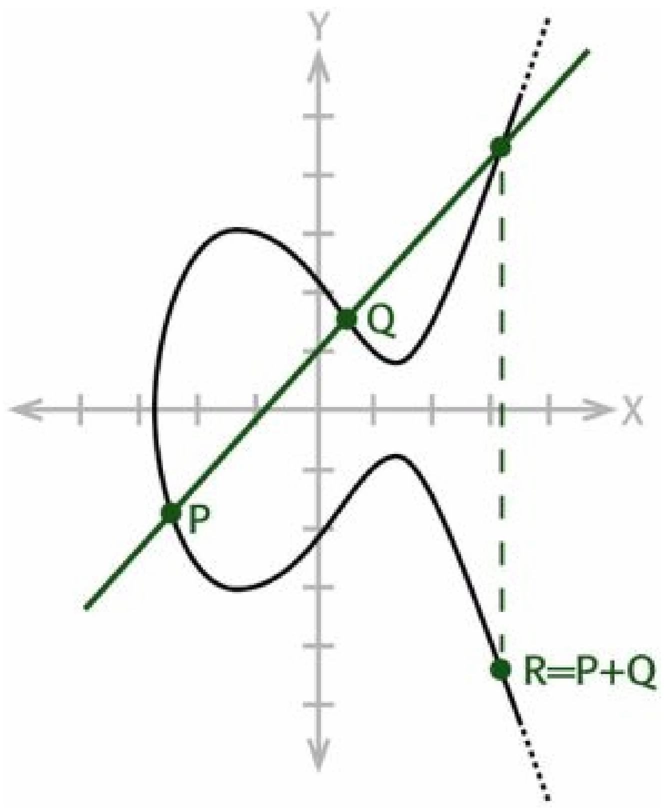


Figure 2.1: Example of elliptic curve.

- Let $P := (x, y) \in E(F_q)^*$ $(E(F_q)^* = E(F_q) - \theta)$. Then $-P = -(x, y) = (x, -y)$ and $P + (-P) = \theta$. So the inverse of P is -P.

- Let $P := (x, y) \in E(F_q)^*$ and $Q := (x_0, y_0) \in E(F_q)^*$. If $x \neq x_0$, then $P + Q = -R$, where -R is a reflection of R in the x-axis and R is the point of intersection of the line joining P and Q with E.

- Let $P := (x, y) \in E(F_q)^*$. Then $P + P = -R$, here -R is also a reflection of R in the x-axis where R is the point of intersection of the tangent at P with E. This particular operation is called point doubling.

This addition law can be shown to be commutative and associative, making the group $(E(F_q)^*, +)$ an Abelian group.

The addition operation can be repeated $P + \ldots + P$ (m times). We note this scalar multiplication with mP, where m is a positive integer. When m is negative, then mP

stands for (-m)(-P). By doing so it is possible, starting from a given point P, to quickly generate other points, by computing *2P*, *3P* and so on. This will generate new points until $kP = P$ for some integer k.

2.4.4 Torsion points

Torsion points are points of finite order. To be more precise, P is said to be a r-torsion point (where r is a positive integer) if $rP = \infty$.

If the curve is defined over a finite field $\mathbf{F}_q$, then all rational points are torsion points, since their order divides the order of the curve and that the order of the curve is finite.

2.5 Bilinear Pairings

Recently, elliptic curve also finds new and exciting applications in cryptography, in particular in cryptographic protocols. This is from so-called bilinear pairing of points over certain elliptic curves. The intuition behind pairings is the construction of a mapping between two well-defined groups. This mapping allows the design of new cryptographic schemes whose security is based on the reduction of one problem in the first group to a different and usually easier problem in the second group.

Today, the known implementations of such mappings, namely the Weil and the Tate pairings, use groups over elliptic curves, while the most popular pairing-based cryptographic scheme is the identity-based encryption scheme proposed by Boneh and Franklin in 2001 [BF03a]. We refer in the following to this document, presenting its definition of map.

Let us consider **G1** and **G2** two groups of order q for some large prime p, and a **bilinear map** $\hat{e} : \mathbf{G1} \times \mathbf{G1} \rightarrow \mathbf{G2}$ between these two groups. The map must satisfy the following properties:

1. Bilinear: a map $\hat{e} : \mathbf{G1} \times \mathbf{G1} \rightarrow \mathbf{G2}$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbf{G1}$ and all $a, b \in Z$

2. Non-degenerate: the map does not send all pairs in $\mathbf{G1} \times \mathbf{G1}$ to the identity in **G2**.

3. Computable: there is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbf{G1}$

The existence of the bilinear map has two direct implications: the **MOV reduction** and the **easiness of the DDH problem**.

Menezes, Okamoto, and Vanstone [MOV93] show that the discrete log problem in $\mathbf{G1}$ is no harder than the discrete log problem in $\mathbf{G2}$. To see this, let $P, Q \in \mathbf{G1}$ be an instance of the discrete log problem in $\mathbf{G1}$ where both P, Q have order q. We wish to find an $\alpha \in Z_q$ such that $Q = \alpha P$. Let $g = \hat{e}(P, P)$ and $h = \hat{e}(Q, P)$. Then, by bilinearity of $\hat{e}$ we know that $h = g^{\alpha}$. By non-degeneracy of $\hat{e}$ both g,h have order q in $\mathbf{G2}$. Hence, the discrete log problem in $\mathbf{G1}$ is reduced to a discrete log problem in $\mathbf{G2}$. It follows that for discrete log to be hard in $\mathbf{G1}$, the discrete log in $\mathbf{G2}$ has to be hard too.

The Decision Diffie-Hellman problem (DDH) in $\mathbf{G1}$ is to distinguish between the distributions $\langle P, aP, bP, abP\rangle$ and $\langle P, aP, bP, cP\rangle$ where a,b,c are random in Z_q^* and P is random in $\mathbf{G1}^*$. Joux and Nguyen [JN03] point out that the DDH in $\mathbf{G1}$ is easy. Given $P, aP, bP, cP \in \mathbf{G1}^*$ we have:

$$c = ab \bmod q \iff \hat{e}(P, cP) = \hat{e}(aP, bP).$$

The Computational Diffie-Hellman problem (CDH) in $\mathbf{G1}$ can still be hard (CDH in $\mathbf{G1}$ is to find abP given random $\langle P, aP, bP\rangle$. Joux and Nguyen [JN03] give examples of mappings $\hat{e} : \mathbf{G1} \times \mathbf{G1} \to \mathbf{G2}$ where CDH in $\mathbf{G1}$ is believed to be hard even though DDH in $\mathbf{G1}$ is easy.

Since the DDH in $\mathbf{G1}$ is easy, security systems have to be based on a variant of the CDH called Bilinear Diffie-Hellman Assumption(BDH).

The Bilinear Diffie-Hellman problem in $\langle \mathbf{G1}, \mathbf{G2}, \hat{e}\rangle$ is defined as follows: given $\langle P, aP, bP, cP\rangle$ for some $a, b, c \in Z_q^*$, compute $W = \hat{e}(P, P)^{abc} \in \mathbf{G2}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving BDH in $\langle \mathbf{G1}, \mathbf{G2}, \hat{e}\rangle$ if

$$\Pr\left[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc}\right] \geq \epsilon$$

where the probability is over the random choice of a,b,c in Z_q^*, the random choice of $P \in \mathbf{G1}^*$ and the random bits of $\mathcal{A}$.

Let $\mathcal{G}$ be a BDH parameter generator. We say that an algorithm $\mathcal{A}$ has advantage $\epsilon(k)$ in solving the BDH problem for $\mathcal{G}$ if for sufficiently large k:

$$Adv_{\mathcal{G},\mathcal{A}}(k) = \Pr[\mathcal{A}(q, \mathbf{G1}, \mathbf{G2}, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc} \mid$$
$$\langle q, \mathbf{G1}, \mathbf{G2}, \hat{e}\rangle \leftarrow \mathcal{G}(1^k), P \leftarrow \mathbf{G1}^*, a, b, c \leftarrow Z_q^*] \geq \epsilon(k)$$

We say that $\mathcal{G}$ satisfies the BDH assumption if for any randomized polynomial time (in k) algorithm $\mathcal{A}$ we have that $Adv_{\mathcal{G},\mathcal{A}}(k)$ is a negligible function. When $\mathcal{G}$ satisfies the BDH assumption, BDH is hard in groups generated by $\mathcal{G}$.

2.6 Conclusions

In this Chapter, we have presented the discipline of cryptography and the main notions to which we will refer in the sequel of the manuscript. In particular, we briefly presented cryptographic protocols and the foundations of the concept of provable security, including the reductionist proof strategy, the random oracle model as well as the generic group model. Moreover, we gave an overview of bilinear pairings over elliptic curves, which are used as a building block for the proposed cryptographic schemes.

Chapter 3

About Secret Handshakes

3.1 Introduction

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.

The purpose of this Chapter is therefore to survey the Secret Handshake protocol family. 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.

In Section 3.5.1, while wrapping up the conclusions of the Chapter, we highlight a number of shortcomings of the solutions available in the state of the art. The latter step then serves as a natural launching pad to introduce the protocols that we present in the subsequent Chapters of this Thesis: these protocols, as we shall see, complete the landscape of available protocols in the Secret Handshake family.

3.2 A Primer on Secret Handshakes

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 2 (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 protocol achieving this objective is shown in Figure 3.1. 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 [BCK96]; both users will compute the same value k only if they both posses the correct secret value K_{p_*}.

$$\begin{array}{ll} u_i \longrightarrow u_j & n_i \\ u_j \longrightarrow u_i & n_j \\ u_j \longleftrightarrow u_i & \text{prove knowledge of } k = MAC_{K_{p_*}}(n_i||n_j) \end{array}$$

Figure 3.1: A simple protocol for Secret Handshake.

First of all we can see that the output of the protocol is a value, k. A proof of knowledge that the same value has been computed by both users accomplishes the proving and verifying actions. This value can also possibly be used by the two users to derive a key used to secure further communication.

A limitation of the protocol of Figure 3.1 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 of Figure 3.1, 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 3 (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 3, the protocol described in Figure 3.1 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 2 as follows:

Definition 4 (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. The same happens for Matching References.

3.2.1 Anonymity and Unlinkability

In this Section we investigate the amount of information leaked to an observer from a Secret Handshake execution. At first, we will state a few definitions, taken from [PH08].

Definition 5 (Anonymity)**.** *Anonymity of a user means that the user is not identifiable within a set of user, the user set.*

Definition 6 (Unlinkability)**.** *Unlinkability of two or more items of interest (IOIs, e.g., subjects, messages, actions, ...) from an observer's perspective means that within the system (comprising these and possibly other items), the observer cannot sufficiently distinguish whether these IOIs are related or not.*

At first, let us notice that Anonymity always relates to users and their identifications, whereas Unlinkability relates to general items of interest, which are not necessarily restricted to users. Let us nonetheless focus on users. The user set is the universe of users $\mathcal{U}$ introduced in Definition 2. Then, 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. In order to make the observer as powerful as possible, we assume that the observer is one of the two users engaging in a protocol. 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.

Let us now turn our attention to Unlinkability of properties. Following the same approach, 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.

3.2.2 A Word on Fairness

Let us now introduce the concept of fairness, according to Asokan's definition [Aso98] and understand its relationship with Secret Handshakes.

Definition 7 (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 [PG99] 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 8 (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 8 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. The full impact of this will be clearer later on in this Chapter.

3.3 The state-of-the-art of Secret Handshakes and related protocols

Thanks to the definitions that we have given in the previous Section, 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.

3.3.1 Matchmaking

The seminal work that introduced Secret Handshakes has been presented by Balfanz *et al.* in 2003 [BDS$^+$03]. Before this paper, a few other works have described protocols with similar objectives. In this Section we will describe these protocols, highlighting some new features whose understanding will be essential for the comprehensive study of the Secret Handshake design space.

In 1985, Baldwin and Gramlich introduced the concept of Matchmaking [BG85]. In Matchmaking protocols, the concept of property that we have referred to as part of Definition 4 become user-generated wishes; for instance, the authors present an example involving a company hiring a high-level manager from within the workforce of other – possibly competing – companies. On the one hand, a manager is reluctant to expose his willingness to leave his current employer; on the other hand, a company is not keen on publicizing that they need a replacement in the current management workforce. Given the sensitiveness of the topics, both users express communication “wishes” on the nature of the other party; wishes can be – as it is the case in the example – fairly sensitive; therefore, users want to interact only with another user who has the same wish. The paper presents a set of protocols involving users and an external matchmaker, whose role is to guarantee fairness in the system. The matchmaker is assumed to be an honest-but-curious third party.

Matchmaking, as described in the paper, is a non-separable protocol that guarantees fairness (according to Definition 7) thanks to the use of a trusted third party. Since the protocol is non-separable, Credentials and Matching References are fused into a single token, namely the self-generated wish. The protocol can be summarized as follows: users have certified public-private key pairs associated with their identities; users write in a public database a number of wish key-pairs associated with a pseudonym; users can then lookup a particular wish, fetching the respective keys. A user would then try to match another user under a particular wish by trying to compute a shared key out of the fetched ones; the key computation involves the public key of both users. If two users are trying to match one another under the same wish, they will end up computing the same key; the key will be published in the matchmaker's database, associated with the pseudonyms users have chosen: this way, the interested users know of a matching while for the matchmaker and for other users remain oblivious.

The protocol preserves Unlinkability and Anonymity of users against the matchmaker and against users engaging in unsuccessful matchmaking protocols. However the protocol fails to guarantee Unlinkability of properties against the matchmaker as observer. In this scheme, both proof and verification control are under the control of users who received a certificate for their keypair. In addition, although wishes can be generated autonomously by users, the fact that the key computation involves certified public keys associated with identities, prevents fraudulent matching attempts: assume that u_i tries to match the wish w off u_j: then, only u_j (i.e. a user chosen and supposedly trusted by u_i), thanks to its private key, will be able to engage in a successful matching.

In [ZN01], Zhang and Needham present another protocol, similar to Baldwin and Gramlich's. This protocol relies on an untrusted matchmaker, acting as a public database where users publish encrypted versions of their wishes. In this case however, there is no proof and verification control: any user is able to search for a match and fake one; in addition, the encryption of wishes is deterministic, and therefore dictionary attacks are not prevented. Last, for the same reason, the protocol does not guarantee Unlinkability of properties.

In [Mea86], Meadows presents a new Matchmaking protocol. The starting point of this work is the fact that Baldwin and Gramlich's solution requires an online third party, which is arguably too strong a requirement in many scenarios. The protocol builds on top of the Diffie-Hellman key exchange [DH03]. The protocol operates as follows: a series of organization federate by exchanging public keys. Each organization can then issue Credentials associated with properties. When two users engage in a protocol run, they exchange Credentials and can tell whether the other party has the same Credential or not. On top of this binary answer, they exchange a key they can use to secure later communications.

Despite referencing Baldwin and Gramlich's work as closest related work, Meadows' protocol achieves significantly different results. The protocol is non-separable as Baldwin and Gramlich's, but it achieves fairness according to Definition 8. Other differences include the fact that, although Anonymity and Unlinkability of properties are preserved, Unlinkability of users is not, since users exchange all the time the same Credential. The biggest change however resides in proof and verification control; since the protocol is non-separable, there is a single token acting as both Credential and Matching Reference. This token is generated by an organization acting as certification authority, which therefore keeps the control over who can prove and who can verify which property.

In [Hoe08], Hoepman presents a very simple protocol – similar to Meadows' [Mea86] – and also based on the Diffie-Hellman key exchange.

3.3.2 Classic Secret Handshakes schemes

In 2003 [BDS$^+$03], Balfanz and colleagues first introduced the notion of Secret Handshake, presenting a scheme based on bilinear pairings. Balfanz *et al.*'s is also the first scheme whose security is formally proved. The scheme introduced in the paper is, according to our definitions, a non-separable protocol: users of the scheme receive from a Group Authority – upon registration – two separate lists of tokens, a list of pseudonyms and a list of secret points associated with each pseudonym. The tokens act as Credentials and Matching References for membership to a group: therefore proof and verification control are under the control of the Group Authority. Upon execution of the Secret Handshake, the users exchange nonces and pseudonyms and perform a local computation. Thanks to the properties of bilinear pairings, the two users share the same key if they belong to the same group. Despite the fact that users receive Credentials and Matching References separately (which is a first step toward separability as we shall see later), the protocol is admittedly still non-separable: indeed it is impossible for a user to only verify the membership of another user without proving its own. This is a consequence of the fact that the protocol openly provides fairness according to Definition 8: therefore, either both users prove and verify membership to the same group, or they only learn of a failed handshake.

The protocol guarantees Anonymity thanks to the use of pseudonyms; Unlinkability of properties is proved by reduction to the Bilinear Diffie-Hellman problem (in the paper, this property is referred to as indistinguishability to eavesdroppers). 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 [XY04] the concept of k-anonymous Secret Handshakes and of reusable Credentials. Let us start with the latter:

Definition 9 (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 as follows: upon a Secret Handshake execution, a user hides its handshake message among the handshake messages that could plausibly have been generated by other $k-1$ users with (possibly different) Credentials. The algorithm used to draft the other users that help hiding the handshake message makes sure that only the correct handshake message will be considered, which guarantees correctness. Thus, with a single Credential, users can execute an arbitrary number of Secret Handshake at the expense of full Anonymity.

In [Ver05], 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 [SG08] 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 [BM93; BMP00; BPR00]), 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. Thus in fact, proof and verification control is under the control of the certification authority. The protocol guarantees fairness according to Definition 8.

In this scheme there is a situation in which an attacker may retrieve some information in an unauthorized way, as follows: let user A without a legitimate Credential for a property – say p_* – interact with user B who does have such Credential; the PAKE – with p_* as a communication wish – would be successful. However, since the misbehaving user would not be able to present an anonymous Credential for p_*, the other user could present the transcript of the protocol and have the attacker's Credential revoked (more on revocation will come in Section 3.4). However the adversary, user A, would still have discovered valuable information on the Credential of the other party.

In [JL07] 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 [JKT08], 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.

3.3.3 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 in Section 3.3.2 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 10 (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 10 requires to revisit the concept of fairness in Secret Handshakes introduced in Definition 8; 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* "

Let us first of all notice that this definition of Secret Handshakes with Dynamic Matching encompasses classic Secret Handshakes: indeed, by simply equipping users of a Secret Handshake with Dynamic Matching scheme with Credentials and Matching

Reference referring to the same property, one can easily obtain classic Secret Handshakes. Therefore, Secret Handshakes with Dynamic Matching are a generalization of classic Secret Handshakes.

The concept of Dynamic Matching has been introduced in [AKB07] by Ateniese and colleagues; however an earlier work already gave the same ability to users, although the fact has not been stressed by the authors in their paper.

In [CJT04], Castelluccia *et al.* introduce the concept of CA-Obliviousness and show how to build Secret Handshakes using CA-Oblivious PKI system. CA-obliviousness can be explained as follows: imagine a sender who wants to send a message to a receiver. The sender's keypair has been certified by a CA. The receiver starts by sending its identity and its certificate to the sender; the sender extracts the public key from the receiver's certificate and encrypts the message destined to the receiver under the extracted public key. The system is CA oblivious if the sender cannot tell which CA certified the receiver's public key, and if the receiver cannot tell which CA the sender assumed upon extracting the public key from the receiver's certificate. Clearly there is one situation in which sender and receiver will disclose one another's CA: this happens of course when both CAs are equal. The CA in this setting plays the role of group manager handing out Credentials (the certificates). Two users can perform a Secret Handshake by exchanging their certificates, extracting public keys, exchanging encrypted messages and showing one another proofs that the decryption has been successful.

$A \longleftarrow B$	ID_B, ω_B
A	computes PK_B assuming CA
$A \longrightarrow B$	$E_{PK_B}(r_A), ID_A, \omega_A, ch_A$
B	computes PK_A assuming CA
$A \longleftarrow B$	$E_{PK_A}(r_B), ch_B, resp_B$
A	checks that $resp_B = H(r_A, r_B, ch_A)$
$A \longrightarrow B$	$resp_A$
B	checks that $resp_A = H(r_A, r_B, ch_B)$

Figure 3.2: Secret Handshakes from CA-Oblivious Encryption.

The protocol can be seen in Figure 3.2: A and B are the two users, ID_A and ID_B are their identities (or pseudonyms), ω_A and ω_B are the certificates of the two public keys PK_A and PK_B. The protocol is successful if and only if ω_A is a certificate issued by CA on PK_A and ID_A and ω_B is a certificate issued by the same CA on PK_B and ID_B. The protocol achieves the same results in terms of Anonymity, Unlinkability, fairness and proof control as Balfanz *et al.*'s. However verification control is under the control of users: this can be seen by the fact that each user can choose arbitrarily the CA that they assume the other user has a Credential from.

Consequently two users can obtain a Secret Handshake with Dynamic Matching as follows: suppose that A is equipped with a certificate ω_A issued from CA_*, the CA responsible for property p_*; suppose also that B is equipped with a certificate ω_B issued from $CA_\circ$, the CA responsible for property $p_\circ$. Also assume that there exist a public map associating the public key of each CA with the property it is responsible for. Then suppose that A chooses to use the public key of $CA_\circ$ in order to compute PK_B, effectively trying to match $p_\circ$ from B; suppose also that B chooses to use the public key of CA_* in order to compute PK_A, effectively trying to match p_* from A. In this case, the Secret Handshake would be successful: A and B would have effectively conducted a Secret Handshake with Dynamic Matching as described in Definition 10.

The concept of Secret Handshake with Dynamic Matching has been introduced in 2007 by Ateniese and colleagues in [AKB07]. The protocol is compliant with Definition 10. 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 under the control 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.

3.3.4 Other Works

In [LDB05], Li Du and Boneh introduce the concept of Oblivious Signature-Based Envelopes (OSBEs). An OSBE is an envelope that is sent by a sender to a receiver, with the assurance that the receiver will only be able to "open" it if he holds a signature on a pre-defined message. The protocol guarantees that (i) the sender will remain oblivious whether the receiver has or has not been able to open the envelope and (ii) the receiver will not be able to read it unless he has the correct signature.

An OSBE round between a sender and a receiver may be seen as a single side of a Secret Handshake: indeed in Secret Handshakes a prover wants to reveal some information to a verifier only under certain circumstances, as a sender of an OSBE wants to reveal its content only to a receiver holding a particular signature.

OSBEs have been studied also in [LDB05], where the authors hint to the possibility of using two symmetric OSBE instances to achieve a Secret Handshake. The Secret Handshake scheme resulting from the two OSBE instances would only support classic Secret Handshakes (in particular, no support for Dynamic Matching); fairness according to Definition 8 would be guaranteed provided that knowledge of the two keys exchanged in the envelope is proved simultaneously. Anonymity and Unlinkability of properties is guaranteed thanks to the obliviousness property; however the protocol would fail to guarantee Unlinkability of users, since Credentials are not reusable.

In [JKT06] Jarecki *et al.* investigate multi-party Secret Handshakes: a group of users will share a key only if all its members belong to a common group. The key agreement scheme is affiliation-hiding as defined in [JL07], and therefore no information about the property object of Credentials and Matching References is leaked. However Credentials are not reusable and therefore Unlinkability of users is guaranteed only if Credentials are never re-used.

3.4 Revocation in Secret Handshakes

Revocation represents an interesting challenge for Secret Handshakes: on the one hand, as we have stressed in Section 3.2.1, a strong requirement for Secret Handshakes is Unlinkability of both users and properties. On the other hand, revocation usually requires means of tagging Credentials in order to single out the revoked ones and refuse any interaction with users bearing them, which in principles requires some degree of linkability.

There are normally three main ways of addressing revocation of cryptographic tokens: short-lived Credentials, black lists and white lists. Short-lived Credentials are Credentials with an expiration date embedded in them; this way, after a pre-defined amount of time, Credentials automatically expire and the certification authority can decide whether to renew them. Black lists, also called revocation lists, are public lists, whose content is signed by the certification authority; these lists contain identifiers for all Credentials that have been revoked. A consequence of choosing this approach is the fact that Credentials need to carry a univocal identifier that can be published in the revocation list. White lists are based on cryptographic accumulators: in short, an accumulator is a public cryptographic token which can be used to test if a value belongs to the accumulator or not. To serve for revocation purposes, the accumulator is initialized by the certification authority, who adds all the identifiers of the valid Credentials. If a user wants to prove that he has a valid Credential, he needs to reveal the identifier of the Credential and anyone can check if it belongs to the accumulator. Camenisch *et al.* have shown in [CL02] how dynamic accumulators [Pfi97; BA93] can be used to achieve efficient revocation for anonymous Credentials, by turning the aforementioned check into a zero knowledge proof-of-knowledge: this way, the verifying user does not need to know the value of the identifier. However dynamic accumulators, quoting Balfanz [BDS$^+$03], are ill-suited for Secret Handshakes, mainly due to the fact that when a verifier has checked that a prover's witness belongs to the accumulator, he has already disclosed the prover's affiliation and can then selfishly refuse to reveal his own witness. Turning accumulator based asymmetric membership verification into symmetric handshakes is indeed an interesting open challenge.

In the rest of this Section we are going to revisit Secret Handshake schemes with respect to their coverage of revocation. Most schemes follow a black-list approach for revocation. The scheme by Jarecki and Liu [JL07] is to the best of our knowledge the only one following a revocation approach based on epochs and short-lived Credentials.

In the seminal work by Baldwin and Gramlich [BG85], the certification authority does not exercise any control on proof and verification; the only option for the CA is to revoke the public key, thus revoking the possibility of executing matchmaking altogether. The protocol described in [ZN01] on the contrary does not support any type of revocation, since it is entirely based on non-certified information. In the work by Meadows [Mea86], the certification authority has two options: either the revocation of the signatures on the user information or a complete rekeying. The latter on the contrary is the only option in the protocol presented by Hoepeman in [Hoe08].

The seminal work by Balfanz and colleagues [BDS$^+$03] supports revocation of single Credentials; the CA can publish user pseudonyms on a public CRL and users can refuse to engage in Secret Handshake if the pseudonym used by the other party belongs to the list. This allows to selectively revoke the power to prove; notice that all pseudonyms look alike and are published on a common list: therefore the nature of the properties of revoked users is not revealed even after revocation. Additionally, since the scheme uses one-time Credentials, a user does not lose its Unlinkability after revocation, provided that the publication of pseudonyms in the list are not performed in batches of one single user at a time, which would inevitably reveal that all of the published Credentials did belong to a single user. The same approach is followed by Jarecki *et al.* in [JKT08] and [JKT06].

The scheme of Vergnaud [Ver05] is based on RSA public key cryptography; Credentials are represented by the knowledge of a pair of encryption/decryption exponents; as a consequence, Credentials (i.e. public keys) can be revoked through a standard revocation list, whereby only the indices of revoked Credentials are published. Similarly, also the work by Xu and Yung [XY04], based on public-key encryption, supports revocation of Credentials through standard revocation of certificates.

As we have mentioned before, revocation is a critical feature of Shin and Gligor's scheme [SG08]; indeed, only by threatening users to revoke their anonymous Credentials if they misbehave, can the certification authority retain proof and verification control.

The seminal work on Dynamic Matching by Ateniese and colleagues does not support revocation. On the contrary, the work by Castelluccia *et al.*, also supporting Secret Handshake with Dynamic Matching, does support revocation of Credentials through revocation of one-time Credentials.

As for OSBE, which we have listed among related protocols, the study of revocation has not been addressed in [LDB05; NT06]. However, given the operation of an OSBE exchange, we can list two different possibilities of revocation of Credentials: one consists in periodically updating the pre-defined message that sender and receiver have to agree upon, for example by concatenating an expiration date. The second approach for revocation is based upon publishing the randomizer of signatures that need to be revoked on a revocation list.

3.5 A Taxonomy of Secret Handshake protocols

Let us recap here the main requirements that we have identified:

- *Separability* addresses the possibility of granting the right to prove separately from the right to verify, and vice versa.

- *Proof-control* relates to the entity which can grant the right to prove possession of a property.
- *Verification-control* relates to the entity which can grant the right to verify possession of a property.
- *Anonymity* requires that a user – upon a Secret Handshake execution – is not identifiable within the user set.
- *Unlinkability of Users* requires that an observer cannot link two Secret Handshake executions to a single user.
- *Unlinkability of Properties* requires that an observer cannot link two Secret Handshake executions to a single property.
- *Fairness* is guaranteed by a Secret Handshake scheme if, at the hand of the protocol, one or both users learn either that both users possess Credentials matching each other's Matching References or neither learns anything except of a failed match.
- *Reusable Credentials* correspond to the fact that user get a single Credential, which they can use an arbitrary number of time, yet preserving Anonymity and Unlinkability of users and properties.
- *Revocation Support* relates to the fact that Credentials can be revoked, dealing with the possibility that users lose their Credentials, or that attackers can steal them.

In Table 3.1 we briefly summarize Secret Handshake and related protocols that we have analyzed in this Chapter, highlighting how they relate to the aforementioned requirements.

3.5.1 Highlighting the Gaps

From an analysis of Table 3.1, we can notice that the protocols in the literature fail to cover two protocol scenarios:

- among the two schemes supporting separable Credentials, namely the work of Castelluccia *et al.* [CJT04] and the work of Ateniese *et al.* [AKB07], 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 all the schemes in the literature, the support for revocation seems to be in contrast with the ability to support more advanced features such as reusable Credentials and/or separability; as an example, the scheme of Castelluccia *et al.* [CJT04] supports revocation but does not support reusable Credentials; the scheme put forward by Jarecki *et al.* [JL07] supports revocation, reusable Credentials but does not support Dynamic Matching;

These two observations motivate the work of the next two Chapters; in Chapter 4 we try to overcome the first of the two limitations by introducing the concept of Dynamic Controlled Matching. In Chapter 5 instead, we address the second point and focus our attention on the topic of revocation for Secret Handshake schemes supporting more complete flavours of Secret Handshake than simple classic Secret Handshake, or meeting advanced requirements such as reusable Credentials.

A third observation instead motivates the work of the last Chapter of this first part of this manuscript: all the schemes in the state-of-the-art require a centralized entity to hand out either or both Credentials and Matching References. In Chapter 6 we make a first attempt to relax this assumption.

3.6 Conclusions

In this Chapter we have investigated the design space of Secret Handshakes and related protocols. Starting from a toy protocol, we have listed a series of requirements and target properties for Secret Handshake protocols. We then analyzed the protocols in the state-of-the-art based on this criteria.

Through this analysis we have identified a number of gaps, of missing features amongst the protocols in the literature. In the subsequent Chapters, we shall focus our attention on researching solutions for those.

Table 3.1: Review of Secret Handshake protocols in the literature.

Protocol	Separability	Proof control	Verification control	Anonymity	Unlinkability of Users	Unlinkability of Properties	Fairness	Reusable Credentials	Revocation Support
Baldwin *et al.* [BG85]	No	Users	Users	Yes	Yes	No	Yes[a]	N/A	Yes[b]
Zhang *et al.* [ZN01]	N/A	None	None	Yes	Yes	No	No	N/A	No
Meadows [Mea86]	No	CA	CA	Yes	No[c]	Yes	Yes	No	No
Hoepman [Hoe08]	No	CA	CA	Yes	Yes	Yes	Yes	No	No
Balfanz *et al.* [BDS$^+$03]	No	CA	CA	Yes	No[c]	Yes	Yes	No	Yes
Xu *et al.* [XY04]	No	CA	CA	Yes[d]	Yes[d]	Yes	Yes	Yes	Yes
Vergnaud [Ver05]	No	CA	CA	Yes	No[c]	Yes	Yes	No	Yes
Shin *et al.* [SG08]	No	CA[e]	CA[e]	Yes	No[c]	Yes	Yes	No	Yes
Jarecki *et al.* [JL07]	No	CA	CA	Yes	Yes	Yes	Yes	Yes	Yes
Jarecki *et al.* [JKT08]	No	CA	CA	Yes	No[c]	Yes	Yes	No	Yes
Castelluccia *et al.* [CJT04]	Yes	CA	Users	Yes	No[c]	Yes	Yes	No	Yes
Ateniese *et al.* [AKB07]	Yes	CA	Users	Yes	Yes	Yes	Yes	Yes	No

[a] This protocol uses a semi-trusted matchmaker, so perfect fairness can be achieved.
[b] What is being revoked here is not the right to prove possession of a property, but the right to participate in the protocol altogether.
[c] Unlinkability of Users can only be achieved by using multiple one-time Credentials.
[d] A user is only anonymous/unlinkable within a group of other k users, k being a parameter of the system.
[e] Users can misbehave but they can be traced and reported to the CA, which can revoke the Credentials.

Chapter 4

Secret Handshake with Dynamic Controlled Matching

4.1 Introduction

In this Chapter we introduce the concept of Dynamic Controlled Matching. Dynamic Controlled Matching fills one of the gaps in the state-of-the-art, helping to complete the landscape of available Secret Handshake solutions. A Secret Handshake scheme with Dynamic Controlled Matching allows for separability yet leaving proof and verification control to the CA. A look at Table 3.1 tells us that no scheme in the literature supports such setting.

The contributions of this Chapter are manifold: in Section 4.2 we formally introduce the concept of Dynamic Controlled Matching, highlighting its flexibility and showing how it can cover both classic Secret Handshakes and Secret Handshakes with Dynamic Matching. In order to build a practical scheme for Secret Handshake with Dynamic Controlled Matching, we proceed in steps: at first, in Section 4.3, we present a simpler scheme that accomplishes only a single round of a Secret Handshake, and prove its security. Then in Section 4.4 we show how this simpler protocol can be used as a building block to create a Secret Handshake scheme that supports Dynamic Controlled Matching, and demonstrate its security.

4.2 Dynamic Controlled Matching

In this Chapter we present the first Secret Handshake scheme with Dynamic Controlled Matching. 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 [BDS$^+$03; Ver05; SG08; Mea86; XY04].

Let us first of all point out that this new scheme is of clear practical use. For instance, it fulfills the requirements identified by the EU Project R4EGov [Eur07]. 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.

It is important also to stress that Secret Handshake with Dynamic Controlled Matching is a generalization of both classic Secret Handshake and Secret Handshake with Dynamic Matching, as we shall see later on in this Chapter.

4.2.1 Syntactic Definition

In a Secret Handshake with Dynamic Controlled Matching the actors are represented by users drawn from a set of users $\mathcal{U}$ and a single CA. Each user can possess properties drawn from a set of properties $\mathcal{P}$. Users are interested in conducting handshakes in order to mutually prove and verify possession of properties. The CA – upon system bootstrap – executes a Setup algorithm to generate public and private system parameters. In a Dynamic Controlled Matching scenario, users need to receive Credentials and Matching References from the CA – which is the only entity able to compute them – in order to conduct a successful Secret Handshake. To this end, the CA exposes two algorithms, $\mathsf{Certify}$ and Grant that users can invoke to receive Credentials and Matching References respectively. Finally, two users can engage in a $\mathsf{Handshake}$ protocol; the protocol is successful if the first user has a Credential for the property associated with the second user's Matching Reference and if the second has a Credential corresponding to the first user's Matching Reference.

A Secret Handshake with Dynamic Controlled Matching is defined by the following algorithms:

- $\mathsf{Setup}(k) \rightarrow (\mathsf{param}, \mathsf{secret})$ is a probabilistic polynomial-time algorithm executed by the certification authority taking a security parameter k as input and producing public parameters param and private parameters secret;

- $\mathsf{Certify}(u, p, \mathsf{secret}) \rightarrow (cred_p)$ is a probabilistic polynomial-time algorithm executed by the certification authority upon a user's request. The certification entity verifies that the supplicant user $u \in \mathcal{U}$ possesses the property $p \in \mathcal{P}$ whose possession will later be proved during the protocol execution; after a successful check, the certification entity issues to u the appropriate Credential $cred_p$. The user can verify the correctness of the Credential. If the verification succeeds, the user accepts the Credential, and aborts otherwise;

- $\mathsf{Grant}(u, p, \mathsf{secret}) \rightarrow (match_{u,p}, X_u)$ is a probabilistic polynomial-time algorithm 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 $p \in \mathcal{P}$. Upon successful verification, the certification entity issues the appropriate Matching Reference formed by the pair $(match_{u,p}, X_u)$ associated with the user; the user can verify the correctness of the Matching Reference. If the verification succeeds, the user accepts the Matching Reference and aborts otherwise;

- $\mathsf{Handshake}$ is a probabilistic polynomial-time two-party algorithm executed by two users, u_i and u_j; the algorithm is composed of three sub-algorithms:
  - $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}) \rightarrow (n, \mathsf{state}_{\mathsf{upd}})$ outputs a nonce n and updates the internal state state;
  - $\mathsf{Handshake.RandomizeCredentials}(\mathsf{param}, cred_p, \mathsf{state}, n) \rightarrow (SH, \mathsf{state}_{\mathsf{upd}})$ takes as input the system parameters, the user Credentials $cred_p$, the nonce n received from the other party and produces the Secret Handshake message SH; it also updates the internal state state;
  - $\mathsf{Handshake.Match}(\mathsf{param}, SH, (match_{u,p}, X_u), \mathsf{state}) \rightarrow (K)$ takes as input the system parameters, the Secret Handshake message SH received from the other user, the local state and the Matching Reference $(match_{u,p}, X_u)$ and outputs a key K;

The algorithm operates as follows:

$$\boxed{\begin{array}{lcll}
u_i & & & : \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_i) \rightarrow (n_i, \mathsf{state}_{i,\mathsf{upd}}) \\
u_i & \longrightarrow & u_j & : n_i \\
u_j & & & : \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_j) \rightarrow (n_j, \mathsf{state}_{j,\mathsf{upd}}) \\
u_j & \longrightarrow & u_i & : n_j \\
u_j & & & : \mathsf{Handshake.RandomizeCredentials}(cred_{p_1}, \mathsf{state}_j, n_i) \\
 & & & \quad \rightarrow (SH_j, \mathsf{state}_{j,\mathsf{upd}}) \\
u_j & \longrightarrow & u_i & : SH_j \\
u_i & & & : \mathsf{Handshake.RandomizeCredentials}(cred_{p_2}, \mathsf{state}_i, n_j) \\
 & & & \quad \rightarrow (SH_i, \mathsf{state}_{i,\mathsf{upd}}) \\
u_i & \longrightarrow & u_j & : SH_i \\
u_i & & & : \mathsf{Handshake.Match}(SH_j, (match_{u_i,p_3}, X_{u_i}), \mathsf{state}_i) \rightarrow (K_i) \\
u_j & & & : \mathsf{Handshake.Match}(SH_i, (match_{u_j,p_4}, X_{u_j}), \mathsf{state}_j) \rightarrow (K_j)
\end{array}}$$

Figure 4.1: The $\mathsf{Handshake}$ algorithm executed by two users u_i and u_j.

The two keys K_i and K_j are the same under the following conditions: $p_1 = p_3$ and $p_2 = p_4$. This implies that u_i's Credential is for the same property as u_j's Matching Reference and similarly, that u_j's Credential is for the same property as u_i's Matching Reference. Otherwise at least one of the two keys is a random value with overwhelming probability.

4.2.2 Creating classic Secret Handshakes and Secret Handshakes with Dynamic Matching

In this Section we highlight the fact that Secret Handshakes with Dynamic Controlled Matching includes both classic Secret Handshakes and Secret Handshakes with Dynamic Matching, thus being a more general and flexible protocol.

In order to create classic Secret Handshakes, the CA can use Setup, Certify, Grant and Handshake with one amendment: the policy followed by the CA upon the execution of Grant is equivalent to granting 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 use Setup, Certify, Grant and Handshake with one amendment: the policy followed by the CA upon the execution of Grant is equivalent to granting 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. Notice that this approach only incurs in limited overhead, since the universe of properties $\mathcal{P}$ is not expected to be too large.

4.2.3 Security Requirements

The security requirements of Secret Handshake with Dynamic Controlled Matching can be effectively resumed as follows:

1. *Correctness*: if two users A and B engage in Handshake, and if A possesses Credentials for B's Matching References and B possesses Credentials for A's Matching References, they both output the same key;

2. *Impersonator Resistance*: given any two properties p_* and $p_\circ$; let us assume two users, A and B, engage in Handshake; B possesses a Credential for $p_\circ$ and A possesses the corresponding Matching Reference. B also has a Matching Reference for p_*; then, it is computationally infeasible for A – without the appropriate Credential for p_* – to distinguish a random value from the key B computes executing Handshake;

3. *Detector Resistance*: given any two properties p_* and $p_\circ$; let us assume that two users, A and B, engage in Handshake; B possesses a Matching Reference for $p_\circ$ and A possesses the corresponding Credential. B also has a Credential for p_*; then, it is computationally infeasible for A – without the appropriate Matching Reference for p_* – to distinguish a random value from the key B computes executing Handshake;

4. *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;

5. *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 Anonymity is not listed among the desired properties since it is intrinsically implied by the fact that we require the stronger requirement of Unlinkability of Users, that includes Anonymity.

We consider the same adversarial type as the one adopted in numerous closely-related works such as [AKB07; CJT04; BDS$^+$03], 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);

- the adversary for Unlinkability requirements is not limited to passively observing protocol instances[1], 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 as follows:

- $\mathcal{O}_{\mathsf{Setup}}$ 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 $\mathsf{Certify}$;
- $\mathcal{O}_{\mathsf{Grant}}$ is invoked when the adversary wants to receive a Matching Reference for a given property through the execution of Grant;

Additionally, we also define the oracle $\mathcal{O}_{\mathsf{H}}$; this oracle can be accessed by the adversary and corresponds – in the real implementation of the scheme – to a hash function, which we assume to be a truly random function. Therefore, our security proofs will take place in the so-called Random Oracle Model [BR93].

Notice that this adversarial model is weaker than the arbitrary composition model adopted by Jarecki and colleagues in [JKT08]. Under this model, as opposed to the one used in this and several other works [AKB07; CJT04; BDS$^+$03], 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"*.

Let us now see how the security requirements listed above can be modeled with the oracles that we have introduced in this Section.

Resistance to Impersonation Attacks An adversary $\mathcal{A}$'s goal is to impersonate a user owning a given Credential, without possessing it. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$; $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$ for which no call to $\mathcal{O}_{\mathsf{Certify}}$ has been made (i.e. $\mathcal{A}$ does not have a Credential for p_*). $\mathcal{A}$ also chooses a property $p_\circ$; then, challenger and adversary engage in the execution of $\mathsf{Handshake}$ as shown in Figure 4.2.

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

$$\boxed{\begin{array}{cccll}
c & & & : & \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_c) \to (n_c, \mathsf{state}_{c,\mathsf{upd}}) \\
c & \longrightarrow & \mathcal{A} & : & n_c \\
\mathcal{A} & \longrightarrow & c & : & n_{\mathcal{A}} \\
c & & & : & \mathsf{Handshake.RandomizeCredentials}(cred_{p_\circ}, \mathsf{state}_c, n_{\mathcal{A}}) \\
 & & & & \to (SH_c, \mathsf{state}_{c,\mathsf{upd}}) \\
c & \longrightarrow & \mathcal{A} & : & SH_c \\
\mathcal{A} & \longrightarrow & c & : & SH_{\mathcal{A}} \\
c & & & : & \mathsf{Handshake.Match}(SH_{\mathcal{A}}, (match_{c,p_*}, X_c), \mathsf{state}_c) \to (K_0) \\
c & & & : & K_1 \overset{R}{\leftarrow} \{0,1\}^n \\
c & & & : & b \overset{R}{\leftarrow} \{0,1\} \\
c & \longrightarrow & \mathcal{A} & : & K_b
\end{array}}$$

Figure 4.2: Challenge for an adversary attempting to break impersonation resistance.

The adversary is the challenged to output a guess for b. n corresponds to the bitlength of the key K_0. Intuitively, this game challenges the capacity of the adversary to impersonate a user owning a Credential for a given property p_*. The adversary generates a handshake message; he is then required to distinguish the key the challenger computes using Handshake.Match on input the adversary's handshake message and the Matching Reference for p_* from a random value of the same length.

Resistance to Detection Attacks Let us consider an adversary $\mathcal{A}$ whose goal is to verify presence of a property of his choice without owning the corresponding Matching Reference. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$; $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$ for which no call to $\mathcal{O}_{\mathsf{Grant}}$ has been made (i.e. $\mathcal{A}$ does not have a Matching Reference for p_*). $\mathcal{A}$ also chooses a property $p_\circ$. Then, challenger and adversary engage in the execution of Handshake as shown in Figure 4.3.

The adversary is the challenged to output a guess for b. n corresponds to the bitlength of the key K_0. Intuitively, this game challenges the capacity of the adversary to impersonate a user owning a Matching Reference for a given property p_*. The adversary receives a handshake message; then, he is required to distinguish the key linked to a successful detection of $p_\circ$ by the challenger and a

$$\begin{array}{lcll}
c & & : & \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_c) \rightarrow (n_c, \mathsf{state}_{c,\mathsf{upd}}) \\
c & \longrightarrow \mathcal{A} & : & n_c \\
\mathcal{A} & \longrightarrow c & : & n_{\mathcal{A}} \\
c & & : & \mathsf{Handshake.RandomizeCredentials}(cred_{p_*}, \mathsf{state}_c, n_{\mathcal{A}}) \\
 & & & \rightarrow (SH_c, \mathsf{state}_{c,\mathsf{upd}}) \\
c & \longrightarrow \mathcal{A} & : & SH_c \\
\mathcal{A} & & : & \mathsf{Handshake.RandomizeCredentials}(cred_{p_\circ}, \mathsf{state}_{\mathcal{A}}, n_c) \\
 & & & \rightarrow (SH_{\mathcal{A}}, \mathsf{state}_{\mathcal{A},\mathsf{upd}}) \\
\mathcal{A} & \longrightarrow c & : & SH_{\mathcal{A}} \\
c & & : & \mathsf{Handshake.Match}(SH_{\mathcal{A}}, (match_{c,p_\circ}, X_c), \mathsf{state}_c) \rightarrow (K_0) \\
c & & : & K_1 \overset{R}{\leftarrow} \{0,1\}^n \\
c & & : & b \overset{R}{\leftarrow} \{0,1\} \\
c & \longrightarrow \mathcal{A} & : & K_b
\end{array}$$

Figure 4.3: Challenge for an adversary attempting to break detection resistance.

successful detection of p_* by the adversary from a random bitstring of the same length.

Unlinkability of Users Consider an adversary $\mathcal{A}$ whose goal is – given any two protocol instances – to determine whether they were generated by the same user. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$; $\mathcal{A}$ is then challenged as follows: he engages in two instances of Handshake with the challenger and is required to tell whether or not the challenger was impersonating the same user over the two Handshake instances.

Unlinkability of Properties Consider an adversary $\mathcal{A}$ whose goal is – given any two protocol instances – to determine whether they were generated to prove possession of the same property. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$; $\mathcal{A}$ chooses a property p_* such that no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted and is then challenged as follows: he engages in two instances of Handshake with the challenger and has to return *true* if he can decide that during both instances, the challenger has used Credentials for p_*.

Let us analyze more in details the games associated with impersonation and detection resistance. We can see that the adversary is – in both cases – challenged to engage in the Handshake protocol, with one parameter missing: in the first case the Credential and in the second case the Matching Reference for property p_*, that was the object of the challenge. Then in both cases the adversary is shown a key; intuitively, the adversary would be able to compute the key had he had the missing token (Credential or Matching Reference); conversely, without, it is computationally infeasible for him to distinguish the correct key from a random value.

4.3 SecureMatching: the building block

In this Section we introduce SecureMatching, a novel cryptographic scheme that allows a prover to convince a verifier that he owns a given property. In Section 4.4 we then leverage this scheme as a building block to create a Secret Handshake protocol with Dynamic Controlled Matching.

4.3.1 Preliminaries

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 bitlength 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 $H : \mathcal{P} \rightarrow \mathbb{G}_1$. A suitable implementation is the MapToPoint function introduced in [BF03a].

4.3.2 Description of SecureMatching

SecureMatching is a prover-verifier protocol wherein a prover can convince a verifier that he 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 us see the algorithms composing SecureMatching:

- Setup is executed by the certification authority at the bootstrap of the system; 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. The CA sets $\tilde{P} \leftarrow rP$, $S \leftarrow sP$, $T \leftarrow tP$ 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;

- 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 $p \in \mathcal{P}$ whose possession will later be proved 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, the user accepts the Credential and aborts otherwise;

- 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 verification is successful, 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. In addition, 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, the user aborts;

- Matching is executed by a prover A and a a verifier B. 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:

  – Matching.Init B picks $n \overset{R}{\leftarrow} \mathbb{Z}_q^*$, and sends $N_1 = nP$ and $N_2 = n\tilde{P}$ to A;

– Matching.RandomizeCredentials A checks whether $\hat{e}(N_1, \tilde{P}) = \hat{e}(N_2, P)$; if so, A picks $r_1, r_2 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sends to B the tuple $SH_A =< r_1 cred_{p_A},\ r_2 N_2,\ r_1 r_2 S,\ r_1 r_2 T >$;

– Matching.Match B checks whether

$$K = \frac{\hat{e}(r_1 cred_{p_A}, r_2 N_2)^{n^{-1}} \cdot \hat{e}(r_1 r_2 S, X_B)}{\hat{e}(r_1 r_2 T, match_{B,p_B})} \tag{4.1}$$

equals to one; X_B is the secret value associated with B, n is the nonce picked upon execution of Matching.Init. If K is equal to one, B concludes that A possesses property p_B (or similarly that p_A and p_B are the same) and outputs $true$; otherwise B outputs $false$;

4.3.3 Security Analysis

In this Section we analyze the security of SecureMatching. The Security Requirements are similar to the ones listed in Section 4.2.3 with minor differences, as we can see below:

1. *Correctness*: if a prover A engages in SecureMatching with a verifier B, and if A possesses a Credential for B's Matching Reference, the result of the execution of Matching.Match by B will be $true$;

2. *Impersonator Resistance*: given a property p_*; let us assume two users, A and B, engage in SecureMatching; B has a Matching Reference for p_*; then, it is computationally infeasible for A – without the appropriate Credential for p_* – to produce a valid handshake message such that the execution of Matching.Match by B will return $true$;

3. *Detector Resistance*: given a properties p_*; let us assume two users, A and B, engage in SecureMatching; B has a Credential for p_*; then, it is computationally infeasible for A – without the appropriate Matching Reference for p_* – to decide whether B, upon the execution of Matching.RandomizeCredentials, has used the Credential for p_* or another random Credential (for which of course A does not dispose of a Matching Reference);

4. *Unlinkability of Users*: it is computationally unfeasible for a user – engaging in two executions of Matching – to tell whether he was interacting with the same user or two different ones;

5. *Unlinkability of Properties*: it is computationally unfeasible for a user – engaging in two executions of Matching without the appropriate Matching References – to tell whether he was interacting with users having Credentials for the same property or for different ones;

Before the actual analysis, let us spend a few words on Unlinkability of users. Unlinkability of users 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 he 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 protocol message 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.

The adversary is allowed to access a number of oracles managed by the challenger in order to interact with the system; the oracles are:

- $\mathcal{O}_{\mathsf{Setup}}$ 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;

Additionally, we also define the oracle $\mathcal{O}_{\mathsf{H}}$; this oracle can be accessed by the adversary and corresponds – in the real implementation of the scheme – to the hash function H which is modeled as a truly random oracle.

In the rest of this Section we introduce three games, Link, Detect and Impersonate, that capture the essence of the attacks mentioned above, and we show the impossibility of these attacks. 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.* [AKB07], which in turn is similar to the one chosen by Balfanz *et al.* in [BDS+03].

Before proceeding further, we state the well-known BDDH problem:

Definition 11 (Hardness of the *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,*

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

is negligible in the security parameter.

This probability is taken over random choice of $P \in \mathbb{G}_1$, a, b and $c \in \mathbb{Z}_q^*$; x is equal to abc with probability $\frac{1}{2}$ and is otherwise equal to a random value in $\mathbb{Z}_q^*/\{abc\}$ with the same probability. This problem has been extensively used in the literature, for instance in [CPP05]. The security proofs for the scheme follow from the hardness of the BDDH problem in the random oracle model.

4.3.3.1 Unlinkability of Properties

Consider an adversary $\mathcal{A}$ whose goal is – given any two protocol messages – 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.

$\mathcal{A}$ can access $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$. $\mathcal{A}$ 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_*. This implies that Match.Match returns *true* 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 Link.

Lemma 1. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvLink}_{\mathcal{A}} := Pr[\mathcal{A} \ wins \ the \ game \ \mathsf{Link}]$$

then a probabilistic, polynomial time algorithm B can create an environment where it uses $\mathcal{A}$'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 $\mathcal{A}$ to decide if $x = abc$. The algorithm B simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game Link to help compute the solution to the BDDH problem. In particular, B implements the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$ as follows:

$\mathcal{O}_{\mathsf{H}}$: on a query $H(x)$, if x has never been queried before, B picks $h_x \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (x, h_x) in a table. Then B flips a random biased coin $guess(x) \in \{0,1\}$ biased as follows: $guess(x)$ equals 1 with probability δ and is equal to 0 with probability $1-\delta$. B answers as follows: if $guess(x) = 0$, he looks up h_x in the table and answers with $H(x) = h_x P$. Instead, if $guess(x) = 1$, B answers with $H(x) = h_x(aP)$;

$\mathcal{O}_{\mathsf{Setup}}$: B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$, sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. B then publishes the public parameter according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries for $cred_{p_i}$ for arbitrary properties p_i; the challenger answers with $cred_{p_i} = vH(p_i)$; $\mathcal{A}$ can check that $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries B for an arbitrary number of Matching References X_{u_i} and $match_{u_i,p_i}$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. Assume that $guess(p_i) = 0$; B can then answer 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. Then, B looks up in the table for the x_{u_i}, and answers: $X_{u_i} = x_{u_i} s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vbH(p_i) + x_{u_i} bP)$; $\mathcal{A}$ can check that $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$;

Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_*, for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted. Assume that $guess(p_*) = 1$. $\mathcal{A}$ inputs two nonce pairs $N_1 = n_1 P, N_1' = n_1 \tilde{P}$ and $N_2 = n_2 P, N_2' = n_2 \tilde{P}$; B can check that the nonces are generated conforming to the specification of the protocol. B then answers as follows:

$$\begin{cases} SH_1 = < r_1 v h_{p_*}(aP),\ r_2 N_1',\ r_1 r_2 S,\ r_1 r_2 T > \\ SH_2 = < v h_{p_*}(xP),\ r_3 N_2,\ r_3 s(cP),\ r_3 t(cP) > \end{cases}$$

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

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

$$\begin{cases} r_1 r_2 v h_{p_*} ab + r_1 r_2 b x_{u1} = r_1 r_2 b(x_{u1} + v\alpha) \\ r_3 v x h_{p_*} + r_3 c b x_{u2} = r_3 c b(x_{u2} + v\alpha) \end{cases} \tag{4.2}$$

are both verified for any user $u_1, u_2 \in \mathcal{U}$. Indeed, if $\mathcal{A}$ returns *true*, according to the *Correctness* requirement, Matching.Match would return *true* for both SH_1 and SH_2 with any Matching Reference in the set $S_{match,p_*} = \{match_{u_i,p_*} : u_i \in \mathcal{U}\}$ for property p_*. Since this system of equations must be valid for any value of x_{u1} and x_{u2}, we can rewrite 4.2 as

$$\begin{cases} r_1 r_2 v h_{p_*} ab = r_1 r_2 b v\alpha \\ r_3 v x h_{p_*} = r_3 c b v\alpha \end{cases} \tag{4.3}$$

and solve the first equation as $\alpha = a h_{p_*}$. If $\mathcal{A}$ wins the game and decides that the two protocol instances 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.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(x) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $x \neq p_*$, then the execution environment is indistinguishable from the actual game Detect. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(x) = 0 \text{ for all } x \neq p_*] = \delta \cdot (1-\delta)^{\mathcal{Q}_{\mathsf{O}}} \tag{4.4}$$

where $\mathcal{Q}_{\mathsf{O}}$ is the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_{\mathsf{O}}}$ we know that the probability in 4.4 is greater than $\frac{1}{e \cdot \mathcal{Q}_{\mathsf{O}}}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvDetect}_{\mathcal{A}}$ as $\mathsf{AdvDetect}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_{\mathsf{O}} \cdot \mathsf{AdvBDDH}_{\mathcal{B}}$. □

4.3.3.2 Detector Resistance

Consider an adversary $\mathcal{A}$ whose goal is to verify presence of a property of his choice without owning the corresponding Matching Reference. $\mathcal{A}$ can access $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$. $\mathcal{A}$ is then challenged as follows: he chooses a property $p_* \in \mathcal{P}$, such that no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted. Finally the adversary engages in Matching with B, and is challenged to decide whether B, upon the execution of Matching.RandomizeCredentials, has used the Credential for p_* or another random Credential (for which of course $\mathcal{A}$ does not dispose of a Matching Reference). We call this game Detect.

Lemma 2. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

then a probabilistic, polynomial time algorithm B can create an environment where it uses $\mathcal{A}$'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 $\mathcal{A}$ to decide if $x = abc$. The algorithm B simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game Detect to help compute the solution to the BDDH problem. In particular, B implements the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$ as follows:

$\mathcal{O}_{\mathsf{H}}$: on a query $H(x)$, if x has never been queried before, B picks $h_x \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (x, h_x) in a table. Then B flips a random biased coin $guess(x) \in \{0, 1\}$ biased as follows: $guess(x)$ equals 1 with probability δ and is equal to 0 with probability $1 - \delta$. B answers as follows: if $guess(x) = 0$, B looks up h_x in the table and answers with $H(x) = h_x P$. Instead, if $guess(x) = 1$, B answers with $H(x) = h_x(aP)$;

$\mathcal{O}_{\mathsf{Setup}}$: B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. B then publishes the public parameter according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries for $cred_{p_i}$ for arbitrary properties p_i; the challenger answers with $cred_{p_i} = vH(p_i)$; $\mathcal{A}$ can check that $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries B for an arbitrary number of Matching References X_{u_i} and $match_{u_i,p_i}$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. Suppose that $guess(p_i) = 0$. 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. Then, B looks up in the table for the x_{u_i}, and answers: $X_{u_i} = x_{u_i} s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vbH(p_i) + x_{u_i} bP)$; $\mathcal{A}$ can check that $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$;

Challenge $\mathcal{A}$ then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted.

Then $\mathcal{A}$ sends to B a pair of nonces $N_1 = nP, n_2 = n\tilde{P}$; B can check that the nonces were generated according to the specifications of the protocol. Suppose that $guess(p_*) = 1$; B can lookup h_{p_*} and answer by sending the tuple

$$< vh_{p_*}(xP), r_1 N_1, r_1 s(cP), r_1 t(cP) > \tag{4.5}$$

Analysis of $\mathcal{A}$'s answer Let's assume $x = abc$. If the adversary wins this game, it means he was successful in detecting p_*; then – according to the *Correctness* requirement – Matching.Match would return *true* with 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. For every user $u_i \in \mathcal{U}$, we can then write

$$K = \frac{\hat{e}(vh_{p_*}(abcP), r_1nP)^{n^{-1}} \cdot \hat{e}(r_1s(cP), X_{u_i})}{\hat{e}(r_1t(cP), t^{-1}(cred_{p_*} + x_{u_i})(bP))} = 1 \tag{4.6}$$

which is the result of Matching.Match returning *true*. Indeed

$$r_1vh_{p_*}x + r_1bcx_{u_*} - r_1c(vh_{p_*}ab + x_{u_i}b) = 0 \tag{4.7}$$

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

Therefore, if $\mathcal{A}$ wins the game and is able to detect property p_*, B can give the same answer to the BDDH.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(x) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $x \neq p_*$, then the execution environment is indistinguishable from the actual game Detect. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(x) = 0 \text{ for all } x \neq p_*] = \delta \cdot (1-\delta)^{\mathcal{Q}_\mathsf{O}} \tag{4.8}$$

where $\mathcal{Q}_\mathsf{O}$ is the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_\mathsf{O}}$ we know that the probability in 4.8 is greater than $\frac{1}{e \cdot \mathcal{Q}_\mathsf{O}}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvDetect}_{\mathcal{A}}$ as $\mathsf{AdvDetect}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_\mathsf{O} \cdot \mathsf{AdvBDDH}_B$. □

4.3.3.3 Impersonation Resistance

The goal of an adversary $\mathcal{A}$ is to impersonate a user possessing a given property, without actually owning the corresponding Credential. $\mathcal{A}$ can access $\mathcal{O}_\mathsf{H}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$. $\mathcal{A}$ is then challenged as follows: he chooses a property $p_* \in \mathcal{P}$, for which no query to $\mathcal{O}_{\mathsf{Certify}}$ has been made, which will be the object of the challenge. $\mathcal{A}$ then receives a nonce value, and has to produce a valid handshake message, able to convince any user with a valid Matching Reference for property p_*, that he owns the Credential $cred_{p_*}$. We call this game Impersonate.

Notice that this game does not prevent an attacker from stealing legitimate users' Credentials and claiming to possess their properties. As discussed in Chapter 3.1, this property is common to many Secret Handshakes schemes in the literature, for instance [AKB07]. We could require Credentials to be stored on password-protected, tamper resistant hardware; an algorithmic solution however would require an efficient revocation method as will be discussed in the next Chapter.

Lemma 3. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

then a probabilistic, polynomial time algorithm B can create an environment where it uses $\mathcal{A}$'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 $\mathcal{A}$ to decide if $x = abc$. The algorithm B simulates an environment in which $\mathcal{A}$ operates: B will in particular implement the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$.

$\mathcal{O}_{\mathsf{H}}$: on a query $H(x)$, if x has never been queried before, B picks $h_x \xleftarrow{R} \mathbb{Z}_q^*$, storing the pair (x, h_x) in a table. Then B flips a random biased coin $guess(x) \in \{0, 1\}$ biased as follows: $guess(x)$ equals 1 with probability δ and is equal to 0 with probability $1 - \delta$. B answers as follows: if $guess(x) = 0$, B looks up h_x in the table and answers with $H(x) = h_x P$. Instead, if $guess(x) = 1$, B answers with $H(x) = h_x(aP)$;

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

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries for $cred_{p_i}$ for arbitrary properties p_i. Assume that $guess(p_i) = 0$; the challenger can answer by first looking up the value h_{p_i} associated with p_i by the oracle $\mathcal{O}_{\mathsf{H}}$; then B answers with $cred_{p_i} = vbH(p_i)$; $\mathcal{A}$ can check that $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries B for an arbitrary number of Matching References X_{u_i} and $match_{u_i,p_i}$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. B answers as follows: if u_i has never been queried before, B picks $x_{u_i} \xleftarrow{R} \mathbb{Z}_q^*$ and stores the pair (u_i, x_{u_i}) in a table. Then, B looks up in the table for the x_{u_i}, and answers: $X_{u_i} = x_{u_i} r s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1} r(vH(p_i) + x_{u_i} P)$; $\mathcal{A}$ can check that $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$;

Challenge $\mathcal{A}$ then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones for which no query to $\mathcal{O}_{\mathsf{Certify}}$ has been submitted. Assume that $guess(p_*) = 1$. B then sends to $\mathcal{A}$ nonces $cP, r(cP)$ according to the protocol, and challenges $\mathcal{A}$ to produce a Secret Handshake message such that Matching.Match returns *true* with 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.

$\mathcal{A}$ answers the challenge with the tuple $(A, B, C, D) \in \mathbb{G}_1^4$; the adversary wins the game if Matching.Match returns *true* on input the tuple received from the adversary; this implies $\hat{e}(A, B)^{c^{-1}} \cdot \hat{e}(X_{u_*}, C) = \hat{e}(D, match_{u_*,p_*})$.

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

$$\alpha\beta c^{-1} + \gamma s^{-1} r x_{u_*} b = \delta(t^{-1} rvah_{p_*} + t^{-1} r x_{u_*}) \tag{4.9}$$

If $\mathcal{A}$ wins the game, any user u_* must be able to verify property p_*, according to the *Correctness* requirement expressed in Section 4.2.3. B can choose any value for x_{u_*}. Consequently, $\alpha\beta c^{-1}$ and $\delta t^{-1} rvah_{p_*}$ must be independent of x_{u_*}. We can then rewrite Equation 4.9 as

$$\begin{cases} \alpha\beta c^{-1} = \delta t^{-1} rvah_{p_*} \\ \gamma s^{-1} r x_{u_*} b = \delta t^{-1} r x_{u_*} \end{cases} \tag{4.10}$$

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

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(x) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Certify}}$ such that $x \neq p_*$, then the execution environment is indistinguishable from the actual game Impersonate. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(x) = 0 \text{ for all } x \neq p_*] = \delta \cdot (1 - \delta)^{\mathcal{Q}_{\mathsf{O}}} \tag{4.11}$$

where $\mathcal{Q}_{\mathsf{O}}$ is the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{\mathsf{Certify}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_{\mathsf{O}}}$ we know that the probability in 4.11 is greater than $\frac{1}{e \cdot \mathcal{Q}_{\mathsf{O}}}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvImpersonate}_{\mathcal{A}}$ as $\mathsf{AdvImpersonate}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_{\mathsf{O}} \cdot \mathsf{AdvBDDH}_{B}$. □

4.4 From SecureMatching to Secret Handshake

Thanks to the SecureMatching protocol introduced in the previous Section, we can now show how we can build Secret Handshake with Dynamic Controlled Matching; a round of SecureMatching covers half of the properties that Secret Handshakes with Dynamic Controlled Matching need to cover. Indeed, a Secret Handshake with Dynamic Controlled Matching is successful if two users A and B engage in the protocol with A possessing Credentials for B's Matching References and B possessing Credentials for A's Matching References; instead, in SecureMatching, it is sufficient for A to possess a Credential for B's Matching Reference; two symmetric instances of SecureMatching, wherein each user plays in turn the role of prover and verifier would accomplish both verifications.

However, two additional characteristics are required: (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.

4.4.1 The Scheme

In this Section we describe an implementation of Secret Handshake with Dynamic Controlled Matching based on SecureMatching. The algorithms Setup, Certify and Grant are the same as the ones of the SecureMatching protocol described earlier on in this Chapter. Let us now describe the Handshake protocol.

- Handshake is a probabilistic polynomial-time two-party algorithm executed by two users, u_i and u_j; the algorithm is composed of three sub-algorithms:
  - Handshake.Init: the user picks $n \overset{R}{\leftarrow} \mathbb{Z}_q^*$, and produces $N_1 = nP$ and $N_2 = n\tilde{P}$;
  - Handshake.RandomizeCredentials: the user, upon receiving nonces N_1 and N_2, checks whether $\hat{e}(N_1, \tilde{P}) = \hat{e}(N_2, P)$; if so, the user picks $r_1, r_2, r_3 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sends the tuple $SH =< r_1(cred_p + r_3P),\ r_2N_2,\ r_1r_2S,\ r_1r_2T >$;

– Handshake.Match: the user parses SH as (A, B, C, D) and computes

$$K = \left(\frac{\hat{e}(A, B)^{n^{-1}} \cdot \hat{e}(C, X_u)}{\hat{e}(D, match_{u,p})} \right)^{r_1 r_2 r_3} \tag{4.12}$$

The algorithm operates as follows: 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 4.4 – 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 $r_{1B}(cred_{P2} + r_{3B}P)$.

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	$r_{1A}(cred_{P1} + r_{3A}P), r_{2A}(n_B \tilde{P}), r_{1A}r_{2A}S, r_{1A}r_{2A}T$
A computes	$K_1 = \frac{\hat{e}(r_{1B}(cred_{P2} + r_{3B}P), r_{2B}(n_A \tilde{P}))^{n_A^{-1}} \cdot \hat{e}(r_{1B}r_{2B}S, X_B)}{\hat{e}(r_{1B}r_{2B}T, match_{B,P2})}^{r_{1A}r_{2A}r_{3A}}$
B computes	$K_2 = \frac{\hat{e}(r_{1A}(cred_{P1} + r_{3A}P), r_{2A}(n_B \tilde{P}))^{n_B^{-1}} \cdot \hat{e}(r_{1A}r_{2A}S, X_A)}{\hat{e}(r_{1A}r_{2A}T, match_{A,P1})}^{r_{1B}r_{2B}r_{3B}}$
A $\longleftrightarrow$ B	mutual proof of knowledge of K_1 and K_2

Figure 4.4: Using SecureMatching to build a Secret Handshake.

The addition of a random value to the Credential, prevents Alice and Bob from getting a true/false result, as it was the case for Matching.Match. Indeed, the same value that in Matching.Match was equal to one upon success, now equals to $\hat{e}(P, P)^{r_{1A}r_{2A}r_{3A}r}$ at Bob's side; similarly on Alice's side, the value is $\hat{e}(P, P)^{r_{1B}r_{2B}r_{3B}r}$.

However, Alice can raise the computed value to the power of $r_{1A}r_{2A}r_{3A}$; similarly, Bob can raise it to the power of $r_{1B}r_{2B}r_{3B}$, and – in case of successful simultaneous matching – the two values will be equal. This value can be subsequently used to derive a secret key, shared between Alice and Bob only if the matching is successful.

4.4.2 Security Analysis

In this Section we focus on the analysis of the security of Secret Handshakes with Dynamic Controlled Matching. The security requirements and attacker model have already been formally defined in Section 4.2.3. 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 with 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 [BDS$^+$03; AKB07] prove resilience to attacks with weaker adversaries: in particular, in the games presented in [BDS$^+$03; AKB07], the attacker is required to actually produce the correct key instead of only distinguishing it from a random value.

4.4.2.1 Impersonation Resistance

The objective of an adversary $\mathcal{A}$ is to impersonate a user owning a given property, without actually owning the corresponding Credential. At first, $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$.

$\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$, such that no query to $\mathcal{O}_{\mathsf{Certify}}$ has been made. $\mathcal{A}$ 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. $\mathcal{A}$ is then challenged in the following way: he engages in Handshake with the challenger, he receives a bitstring and has to tell whether this bitstring 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 bitstring of the same length. We call this game ImpersonateSH.

Lemma 4. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

then a probabilistic, polynomial time algorithm B can create an environment where it uses $\mathcal{A}$'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 $\mathcal{A}$ to decide if $x = abc$. The algorithm B simulates an environment in which $\mathcal{A}$ operates: B will in particular implement the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$.

$\mathcal{O}_{\mathsf{H}}$: on a query $H(x)$, if x has never been queried before, B picks $h_x \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (x, h_x) in a table. Then B flips a random biased coin $guess(x) \in \{0, 1\}$ biased as follows: $guess(x)$ equals 1 with probability δ and is equal to 0 with probability $1 - \delta$. B answers as follows: if $guess(x) = 0$, B looks up h_x in the table and answers with $H(x) = h_x P$. Instead, if $guess(x) = 1$, B answers with $H(x) = h_x(aP)$;

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

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries for $cred_{p_i}$ for arbitrary properties p_i. Assume that $guess(p_i) = 0$; the challenger answers by first looking up the value h_{p_i} associated with p_i by the oracle $\mathcal{O}_{\mathsf{H}}$; then B answers with $cred_{p_i} = vbH(p_i)$; $\mathcal{A}$ can check that $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries B for an arbitrary number of Matching References X_{u_i} and $match_{u_i,p_i}$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. 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. Then, B looks up in the table for the x_{u_i}, and answers: $X_{u_i} = x_{u_i} r s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1} r(vH(p_i) + x_{u_i} P)$; $\mathcal{A}$ can check that $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$;

Challenge $\mathcal{A}$ then chooses the property $p_* \in \mathcal{P}$ which is the object of the challenge among the ones for which no query to $\mathcal{O}_{\mathsf{Certify}}$ has been made in the previous phase. Finally, $\mathcal{A}$ chooses the property $p_\circ$ that B will use for his Matching Reference.

After this phase, $\mathcal{A}$ and B engage in a Secret Handshake instance; in particular $\mathcal{A}$ sends nonces nP and $n\tilde{P}$; B verifies that the nonces are compliant, sends to $\mathcal{A}$

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 $\mathcal{A}$ to produce SH; $\mathcal{A}$ answers the challenge with $(A, B, C, D) \in \mathbb{G}_1^4$. Suppose that $guess(p_*) = 1$; B then sends $\mathcal{A}$ a key formed as follows:

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

and challenges $\mathcal{A}$ to tell whether it is the correct key or a random bitstring of the same length. Correct key means the key the challenger computes with any Matching Reference $match_{u_*,p_*}$ and X_{u_*} of a user $u_* \in \mathcal{U}$, as specified by the *Correctness* requirement. $\mathcal{A}$ answers the challenge with a bit b; $\mathcal{A}$ wins the game when $b = 0$ if the key is a random bitstring, and $b = 1$ if the key is correct.

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

$$K = \left(\frac{\hat{e}(A, B)}{\hat{e}(C, xP)^{s^{-1}rvh_{p_*}}} \right)^{r_1r_2r_3} = \left(\frac{\hat{e}(A, B)^{c^{-1}} \cdot \hat{e}(C, X_{u_*})}{\hat{e}(D, match_{u_*,p_*})} \right)^{r_1r_2r_3c} \tag{4.13}$$

The right end of the equality represents the fact that if $\mathcal{A}$ was successful in the impersonation and detected the key as correct, then the key must be computable executing Handshake.Match with the Matching Reference for property p_* and the random values r_1, r_2 and r_3c used by the challenger during the invocation of the algorithm Handshake.RandomizeCredentials.

From Equation 4.13 we can rewrite

$$\alpha\beta - \gamma rs^{-1}h_{p_*}vx = \alpha\beta + \gamma x_{u_*}rs^{-1}bc - \delta t^{-1}rc(vh_{p_*}a + x_{u_*}) \tag{4.14}$$

Notice that the challenger can choose any value for x_{u_*}. Consequently, $\gamma rs^{-1}h_{p_*}vx$ and $\delta t^{-1}rvh_{p_*}ac$ must be independent of x_{u_*}. We can then rewrite Equation 4.14 as

$$\begin{cases} \gamma rs^{-1}h_{p_*}vx = \delta t^{-1}rvh_{p_*}ac \\ \gamma x_{u_*}rs^{-1}bc = \delta t^{-1}rcx_{u_*} \end{cases} \tag{4.15}$$

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 $\mathcal{A}$ answers $b = 0$, through the same calculation we conclude that $x \neq abc$, which is the negative answer to the BDDH problem. Therefore if $\mathcal{A}$ wins the game, B can solve the BDDH problem by answering with b.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(x) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Certify}}$ such that $x \neq p_*$, then the execution environment is indistinguishable from the actual game ImpersonateSH. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(x) = 0 \text{ for all } x \neq p_*] = \delta \cdot (1 - \delta)^{\mathcal{Q}_{\mathsf{O}}} \qquad (4.16)$$

where $\mathcal{Q}_{\mathsf{O}}$ is the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{\mathsf{Certify}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_{\mathsf{O}}}$ we know that the probability in 4.16 is greater than $\frac{1}{e \cdot \mathcal{Q}_{\mathsf{O}}}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvImpersonateSH}_{\mathcal{A}}$ as $\mathsf{AdvImpersonateSH}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_{\mathsf{O}} \cdot \mathsf{AdvBDDH}_B$. □

4.4.2.2 Detector Resistance

Consider an adversary $\mathcal{A}$ whose goal is to verify presence of a property of his choice without owning the corresponding Matching Reference. At first, $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$.

$\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$, for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been made. $\mathcal{A}$ 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. $\mathcal{A}$ is then challenged in the following way: he has to engage in Secret Handshake with the challenger and he receives a bitstring, and has to tell whether the bitstring is the key linked to a successful detection of $p_\circ$ by the challenger and to a successful detection of p_* by the adversary, or just a random bitstring of the same length. $\mathcal{A}$ clearly does not posses any of the Matching References in S_{match,p_*}. We call this game DetectSH.

Lemma 5. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

then a probabilistic, polynomial time algorithm B can create an environment where it uses $\mathcal{A}$'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 $\mathcal{A}$ to decide if $x = abc$. The algorithm B simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game Detect to help compute the solution to the BDDH problem. In particular, B will run for $\mathcal{A}$ the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$.

$\mathcal{O}_{\mathsf{H}}$: on a query $H(x)$, if x has never been queried before, B picks $h_x \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (x, h_x) in a table. Then B flips a random biased coin $guess(x) \in \{0, 1\}$ biased as follows: $guess(x)$ equals 1 with probability δ and is equal to 0 with probability $1 - \delta$. B answers as follows: if $guess(x) = 0$, B looks up h_x in the table and answers with $H(x) = h_x P$. Instead, if $guess(x) = 1$, B answers with $H(x) = h_x(aP)$;

$\mathcal{O}_{\mathsf{Setup}}$: B picks $s, t, v \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{P} \leftarrow (bP)$, $S \leftarrow sP$, $T \leftarrow tP$ and $V \leftarrow v(bP)$. B then publishes the public parameter according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries for $cred_{p_i}$ for arbitrary properties p_i; the challenger answers by first looking up the value h_{p_i} associated with p_i by the oracle $\mathcal{O}_{\mathsf{H}}$; then B answers with $cred_{p_i} = vH(p_i)$; $\mathcal{A}$ can check that $\hat{e}(cred_{p_i}, \tilde{P}) = \hat{e}(H(p_i), V)$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries B for an arbitrary number of Matching References X_{u_i} and $match_{u_i,p_i}$ for any given pairs $(u_i, p_i) \in \mathcal{U} \times \mathcal{P}$. Let us assume that $guess(p_i) = 0$. 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. Then, B looks up in the table for the x_{u_i}, and answers: $X_{u_i} = x_{u_i} s^{-1}(bP)$ and $match_{u_i,p_i} = t^{-1}(vbH(p_i) + x_{u_i} bP)$; $\mathcal{A}$ can check that $\hat{e}(T, match_{u_i,p_i}) = \hat{e}(H(p_i), V) \cdot \hat{e}(X_{u_i}, S)$ hold;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{H}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$;

Challenge $\mathcal{A}$ then chooses the property $p_* \in \mathcal{P}$ which is object of the challenge among the ones for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been made. Finally, $\mathcal{A}$ chooses the property $p_\circ$ that B will use for his Credential.

After this phase, $\mathcal{A}$ and B engage in a Secret Handshake instance; in particular, $\mathcal{A}$ sends to B a pair of nonces $N_1 = nP, N_2 = n\tilde{P}$; B can check that they were generated according to the specifications of the protocol. B verifies that the nonces are compliant, sends to $\mathcal{A}$ nonces $n'P, n'\tilde{P}$ according to the protocol.

Suppose that $guess(p_*) = 1$. Notice that, supposing $guess(p_*) = 1$, $H(p_*) = h_{p_*} aP$. B picks $r_1, r_2, r_3 \overset{R}{\leftarrow} \mathbb{Z}_q^*$, sets $r_1' = r_1 bc$, $r_2' = r_2 b^{-1}$ and $r_3' = (bc)^{-1}(vh_{p_*} x + r_3 b - vabch_{p_*})$ and creates a handshake tuple generated adhering to the rules of the algorithm Handshake.RandomizeCredentials, as follows:

$$\begin{cases} r_1'(vH(p_*) + r_3'P) = r_1(vh_{p_*}(xP) + r_3bP) \\ r_2'N_2 = r_2N_1 \\ r_1'r_2'S = r_1r_2s(cP) \\ r_1'r_2'T = r_1r_2t(cP) \end{cases}$$

and sends it to $\mathcal{A}$; B in turn receives $SH = (A, B, C, D) \in \mathbb{G}_1^4$ from $\mathcal{A}$.

At this point B uses the received handshake message and executes Handshake.Match to generate the key to be sent to $\mathcal{A}$. Let us go through the steps of the execution of this sub-algorithm. At first B computes

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

for an unknown value μ. In this step, the challenge has matched property $p_\circ$ from the handshake message received from the adversary. Then the challenger should raise the value K' to the power $r_1'r_2'r_3'$. To do so, the challenger assumes that $x = abc$; the product $r_1'r_2'r_3'$ thus becomes equal to $r_1r_2r_3$. Then the challenger can send to the adversary the following key

$$K = K'^{r_1r_2r_3} = \left(\hat{e}(P,\tilde{P})^{\mu}\right)^{r_1r_2r_3}$$

and challenges $\mathcal{A}$ to tell whether K is the correct key or a random string of the same length. Correct key means the key linked to a successful detection of p_* by the adversary and a successful detection of $p_\circ$ by the challenger.

$\mathcal{A}$ answers the challenge with a bit b; $\mathcal{A}$ wins the game if $b = 0$ if the key is a random bitstring, and $b = 1$ if the key is correct.

Analysis of $\mathcal{A}$'s answer If $\mathcal{A}$ 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. This last consideration stems from the *Correctness* requirement expressed in Section 4.2.3. We can therefore write

$$\left(\frac{\hat{e}(r_1(vh_{p_*}(xP) + r_3bP), r_2nP)^{n^{-1}} \cdot \hat{e}(r_1r_2s(cP), x_{u_*}s^{-1}(bP))}{\hat{e}(r_1r_2t(cP), t^{-1}(vh_{p_*}a + x_{u_*})(bP))}\right)^{\mu} = K = \\ = \left(\hat{e}(P,\tilde{P})^{\mu}\right)^{r_1r_2r_3}$$

From this expression we notice that

$$(r_1r_2vh_{p_*}x + r_1r_2r_3b + r_1r_2x_{u_*}bc - r_1r_2bc(vh_{p_*}a + x_{u_*}))\mu = r_1r_2r_3b\mu \tag{4.17}$$

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

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(x) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $x \neq p_*$, then the execution environment is indistinguishable from the actual game ImpersonateSH. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(x) = 0 \text{ for all } x \neq p_*] = \delta \cdot (1-\delta)^{\mathcal{Q}_{\mathsf{O}}} \tag{4.18}$$

where $\mathcal{Q}_{\mathsf{O}}$ is the number of queries $\mathcal{A}$ makes to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_{\mathsf{O}}}$ we know that the probability in 4.18 is greater than $\frac{1}{e \cdot \mathcal{Q}_{\mathsf{O}}}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvImpersonateSH}_{\mathcal{A}}$ as $\mathsf{AdvImpersonateSH}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_{\mathsf{O}} \cdot \mathsf{AdvBDDH}_B$. □

4.4.3 A Word on Man-In-The-Middle Attacks

In this Section we focus on man-in-the-middle attacks and investigate their relation to the Secret Handshake scheme presented in Section 4.4. 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 assures 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 bitstring.

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 classic Secret Handshakes such as the scheme by Balfanz *et al.* [BDS$^+$03]. 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 untrusted 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 Unlinkability and Anonymity, whereas if the Credentials carry a reference to the identity of their possessor, these requirements 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.* [BDS+03]. 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 that is based on pseudonymity, does not completely solve 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 as illustrated with the following 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.

4.5 Conclusion

In this Chapter we have introduced the concept of Secret Handshake with Dynamic Controlled Matching: after a semantic description of the protocol and of its security requirements, we have proposed at first a prover-verifier protocol using bilinear pairings; then, we have shown how to use this protocol in order to build the first Secret Handshake with Dynamic Controlled Matching.

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.

Chapter 5

Revocation in Secret Handshakes

5.1 Introduction

Supporting revocation in Secret Handshake scenarios is a challenging task. On the one hand, we have seen in Chapter 3 that Anonymity and Unlinkability of users and properties is a desired requirement. On the other hand, most of the approaches to the revocation problem require Credentials to be somehow labeled, so that – either through whitelists or blacklists – revoked Credentials can be told apart from active ones: this openly violates the requirements of Unlinkability.

As described in Chapter 3 in Table 3.1, most of the schemes in the state of the art so far either support revocation for limited versions of Secret Handshake (e.g. without reusable Credentials, without separability) or they support more complete versions of Secret Handshake with no possibility of introducing revocation, at least not without radical changes to the protocol. Notice that also the scheme that we presented in Chapter 4 falls in the latter category: it supports a wide variety of Secret Handshake protocols with reusable Credentials, but revocation support is missing.

In this Chapter we address this challenging problem and show how we can support in a comprehensive way all Secret Handshakes scenarios known in the literature, classic Secret Handshakes, Secret Handshakes with Dynamic Matching or Secret Handshakes with Dynamic Controlled Matching, adding revocation support to each. Our contributions are manifold: at first, in Section 5.2 we define the requirements for a Secret Handshake scheme supporting revocation. Then in Section 5.3 we present the first Secret Handshake scheme with Dynamic Controlled Matching with revocation support and reusable Credentials. The analysis of the security of the scheme, in particular for the resistance to impersonation attacks, requires the support of a rather complex proving strategy, in conjunction with a new hardness assumption, the SM Problem, for which we give evidence of being a hard problem in the generic group model. Finally, in Section 5.4 we introduce the first Secret Handshake scheme with Dynamic Matching with revocation support and reusable Credentials, thus bringing revocation support to the work of Ateniese et al. [AKB07].

5.2 Problem Statement and Motivation

In Chapter 3 we have highlighted a number of features of Secret Handshake schemes. In particular, we have stressed that reusable Credentials is a key feature of a practical scheme; the same holds for revocation. In addition, we have listed Unlinkability of users as an extremely relevant security requirement, whose importance is increased by the sensitive use cases that we have sketched in Chapter 1.

In Table 3.1 unfortunately we can see that only two schemes support all three features, namely [XY04] and [JL07]. The first scheme however only supports a limited version of Anonymity, namely k-Anonymity. The second scheme follows a different approach from white/black list, namely, an epoch-based approach with short-lived Credentials. However both schemes fail to satisfy another important requirement: separability. As we have seen in Chapter 3, separability is the key enabler for more flexible versions of Secret Handshake, namely Secret Handshake with Dynamic Matching and Secret Handshake with Dynamic Controlled Matching.

It is therefore evident that a scheme supporting separability, reusable Credentials, revocation and Unlinkability is missing. This is going to be the contribution of this Chapter.

A possible alternative approach to the one that we follow in the remainder of this Chapter would be the use of Zero-Knowledge Proofs of Knowledge (ZKPK) [GMW86; GMR89; GMW91] together with Accumulators, as suggested in [CL02]. This approach allows a prover to prove to a verifier possession of valid (i.e. not revoked) Credentials in an anonymous and unlinkable way. However this accomplishes just half of the requirements of a Secret Handshake, namely one the two parties proving to the other that it possesses a Credential: turning this into a two-party protocol satisfying fairness according to Definition 8 is not a straightforward task. These shortcomings have already been acknowledge by Balfanz and colleagues [BDS^{+}03], who stated that accumulators are ill-suited for Secret Handshake scenarios. For this reason we do not follow this approach.

5.2.1 Syntactic Definition

In this Section we formally define a Secret Handshake scheme supporting revocation of Credentials. The scheme is constructed using two symmetric instances of a prover-verifier protocol. The difference with the scheme of Chapter 4 is that in this case the two schemes will effectively be separated, leading to the computation of two separate keys. It is therefore essential that, at the end of the scheme, the two users can prove to one another knowledge of both keys simultaneously: this will ensure fairness according to Definition 8.

The algorithms composing the scheme are:

- $\mathsf{Setup}(k) \rightarrow (\mathsf{param}, \mathsf{secret})$ is a probabilistic polynomial-time algorithm executed by the certification authority taking a security parameter k as input and producing public parameters param and private parameters secret;

- $\mathsf{Certify}(u, p, \mathsf{secret}) \rightarrow (cred_{u,p}, x_{u,p})$ is a probabilistic polynomial-time algorithm executed by the certification authority upon a user's request. The certification entity verifies that the supplicant user $u \in \mathcal{U}$ possesses the property $p \in \mathcal{P}$ whose possession will later be proved during the protocol execution; after a successful verification, the certification entity issues to u the appropriate Credential formed by the pair $(cred_{u,p}, x_{u,p})$. The user can verify the correctness of the Credential. If the verification succeeds, the user accepts the Credential and aborts otherwise;

- $\mathsf{Grant}(u, p, \mathsf{secret}) \rightarrow (match_p)$ is a probabilistic polynomial-time algorithm 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 verification is successful, the certification entity issues the appropriate Matching Reference $match_p$; the user can verify the correctness of the Matching Reference. If the verification succeeds, the user accepts the Matching Reference and aborts otherwise;

- $\mathsf{Revoke}(u, p, \mathsf{secret}, \mathsf{rl}) \rightarrow (\mathsf{rl}_{\mathsf{upd}})$ is a probabilistic polynomial-time algorithm executed by the certification entity when the Credential for property p owned by user u needs to be revoked. The certification entity executes the algorithm by updating a public revocation list rl, appending revocation information for the newly revoked Credential;

- $\mathsf{Handshake}$ is a probabilistic polynomial-time two-party algorithm executed by two users, u_i and u_j; the algorithm is composed of three sub-algorithms:

  – $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}) \rightarrow (n, \mathsf{state}_{\mathsf{upd}})$ outputs a nonce n and updates the internal state state;

  – $\mathsf{Handshake.RandomizeCredentials}(\mathsf{param}, (cred_{u,p_1}, x_{u,p_1}), \mathsf{state}, n) \rightarrow (SH,$ $\mathsf{state}_{\mathsf{upd}}$, $K_1)$ takes as input the system parameters, the user Credential, the nonce n received from the other party and produces the Secret Handshake message SH and the key K_1 linked to the proof of possession of property p_1; it also updates the internal state state;

  – $\mathsf{Handshake.CheckRevoked}(\mathsf{param}, SH, \mathsf{rl}, \ldots) \rightarrow (b)$ takes as input the system parameters, the Secret Handshake message received from the other party, an updated version of the revocation list rl and additional parameters; the algorithm outputs a boolean value set to true if the Credential used to generate SH has been revoked;

  – $\mathsf{Handshake.Match}(\mathsf{param}, SH, match_{p_2}, \mathsf{state}) \rightarrow (K_2)$ takes as input the system parameters, SH, the Credentials received from the other user, the local state, the Matching Reference $match_{u,p}$; the user first executes sub-algorithm $\mathsf{Handshake.CheckRevoked}$ to check whether the other user's Credential has been revoked, in which case the algorithm aborts; if $\mathsf{Handshake.CheckRevoked}$ gives a negative answer (i.e. if the Credential has not been revoked), the algorithm outputs a key K_2, linked to the verification of possession of property p_2;

Assume two users u_i and u_j engage in the execution of $\mathsf{Handshake}$; u_i's input is a Credential for property p_2, $(cred_{u_i,p_2}, x_{u_i,p_2})$ and a Matching Reference for property p_3, $match_{p_3}$; u_j's input is a Credential for property p_1, $(cred_{u_j,p_1}, x_{u_j,p_1})$

and a Matching Reference for property p_4, $match_{p_4}$. The algorithm operates as described in Figure 5.1.

u_i			:	$\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_i) \rightarrow (n_i, \mathsf{state}_{i,\mathsf{upd}})$
u_i	$\longrightarrow$	u_j	:	n_i
u_j			:	$\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_j) \rightarrow (n_j, \mathsf{state}_{j,\mathsf{upd}})$
u_j	$\longrightarrow$	u_i	:	n_j
u_j			:	$\mathsf{Handshake.RandomizeCredentials}((cred_{u_j,p_1}, x_{u_j,p_1}), \mathsf{state}_j, n_i)$ $\rightarrow (SH_j, \mathsf{state}_{j,\mathsf{upd}}, K_{j,1})$
u_j	$\longrightarrow$	u_i	:	SH_j
u_i			:	$\mathsf{Handshake.RandomizeCredentials}((cred_{u_i,p_2}, x_{u_i,p_2}), \mathsf{state}_i, n_j)$ $\rightarrow (SH_i, \mathsf{state}_{i,\mathsf{upd}}, K_{i,1})$
u_i	$\longrightarrow$	u_j	:	SH_i
u_i			:	$\mathsf{Handshake.Match}(SH_j, match_{p_3}, \mathsf{state}_i) \rightarrow (K_{i,2})$
u_j			:	$\mathsf{Handshake.Match}(SH_i, match_{p_4}, \mathsf{state}_j) \rightarrow (K_{j,2})$

Figure 5.1: The Handshake algorithm executed by two users u_i and u_j.

The two pairs of keys $K_{i,1}$, $K_{j,2}$, and $K_{i,2}$, $K_{j,1}$ are equal under the following conditions: $p_1 = p_3$ and $p_2 = p_4$. This implies that u_i's Credential is for the same property as u_j's Matching Reference and similarly, that u_j's Credential is for the same property as u_i's Matching Reference. Otherwise at least one of the two keys is a random value with overwhelming probability.

Notice that Handshake.CheckRevoked has been purposely defined with a different set of arguments; indeed, depending on the particular instance of Secret Handshake (classic, with Dynamic Matching, with Dynamic Controlled Matching), a variable list of arguments is required, as we shall see in the rest of this Chapter.

5.2.2 Security Requirements

In this Section we analyze the security requirements of Secret Handshakes, taking revocation into consideration.

In the previous Section we have seen that two separate keys are computed; with a reference to Figure 5.1, key $K_{i,1}$ is the key that u_i is able to compute if successful in the proof of possession of property p_2; key $K_{j,2}$ is the key that u_j is able to compute if successful in the verification of property p_4. If both are successful in the proof and verification (respectively), then $K_{i,1} = K_{j,2}$. The same applies for $K_{i,2}$ and $K_{j,1}$. Thus, the two pairs of keys are effectively separate, in that each pair depends on the proof/verification of one property. Therefore in the security analysis we will acknowledge this separation by focusing only on one key at a time in the games for detection and impersonation resistance.

The handshake is sealed by a simultaneous proof of knowledge of both keys. It is a necessary prerequisite for the security of the scheme that the proof of knowledge does not – in any way – leak the actual value of the keys and that it proves knowledge of both keys simultaneously.

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

1. *Correctness*: if two users A and B engage in Handshake, and if A possesses valid Credentials for B's Matching References and B possesses valid Credentials for A's Matching References, they both output the same pair of keys;

2. *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;

3. *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;

4. *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;

5. *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 for Impersonator Resistance we consider a weaker requirement than the one considered in Section 4.2.3: indeed the adversary is required to output the correct key instead of distinguishing between the correct key and a random value. The same holds for the security analyses of Secret Handshake schemes in the literature, such as [BDS+03; AKB07].

We consider a similar type of adversarial model as in Section 4.2.3. A substantial difference with respect to that model however, is brought forward by the fact that this scheme supports revocation. Indeed, in the present model the adversary can always obtain Credentials for any properties at his will, with no restriction; however the Credentials for properties being object of impersonation attacks are revoked before the challenge.

The adversary is allowed to access a number of oracles managed by the challenger in order to interact with the system; the oracles are:

- $\mathcal{O}_{\mathsf{Setup}}$ 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 $\mathsf{Certify}$;
- $\mathcal{O}_{\mathsf{Grant}}$ is invoked when the adversary wants to receive a Matching Reference for a given property through the execution of Grant;
- $\mathcal{O}_{\mathsf{Revoke}}$ is invoked when the adversary wants to receive a Revocation Handle for a given Credential through the execution of Revoke;

Let us now see how the security requirements listed above can be modeled with the oracles that we have introduced in this Section.

Resistance to Impersonation Attacks The goal of an adversary $\mathcal{A}$ is to impersonate a user possessing a given property, without actually owning the corresponding Credential. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$; $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$; all Credentials for p_* $\mathcal{A}$ received so far are revoked. Then, challenger and adversary engage in the execution of only one side of the $\mathsf{Handshake}$ protocol, as shown in Figure 5.2. The adversary is the challenged to output the key K;

$$\begin{array}{cccll}
c & & & : & \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_c) \rightarrow (n_c, \mathsf{state}_{c,\mathsf{upd}}) \\
c & \longrightarrow & \mathcal{A} & : & n_c \\
\mathcal{A} & \longrightarrow & c & : & SH_{\mathcal{A}} \\
\mathcal{A} & \longrightarrow & c & : & K
\end{array}$$

Figure 5.2: Challenge for an adversary attempting to break impersonation resistance.

Resistance to Detection Attacks Let us consider an adversary $\mathcal{A}$ whose goal is to verify presence of a property of his choice without owning the corresponding Matching Reference. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$; $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$ for which no call to $\mathcal{O}_{\mathsf{Grant}}$ has been made (i.e. $\mathcal{A}$ does not have a Matching Reference for p_*). Then, challenger and adversary engage in the execution of Handshake as shown in Figure 5.3.

$$\begin{array}{cccll}
\mathcal{A} & \longrightarrow & c & : & n_{\mathcal{A}} \\
c & & & : & \mathsf{Handshake.RandomizeCredentials}((cred_{c,p_*}, x_{c,p_*}), \mathsf{state}_c, n_{\mathcal{A}}) \\
 & & & & \rightarrow (SH_c, \mathsf{state}_{c,\mathsf{upd}}, K_{c,1}) \\
c & \longrightarrow & \mathcal{A} & : & SH_c \\
c & & & : & K_0 \leftarrow K_{c,1} \\
c & & & : & K_1 \overset{R}{\leftarrow} \{0,1\}^n \\
c & & & : & b \overset{R}{\leftarrow} \{0,1\} \\
c & \longrightarrow & \mathcal{A} & : & K_b
\end{array}$$

Figure 5.3: Challenge for an adversary attempting to break detection resistance.

The adversary is the challenged to output a guess for b. n corresponds to the bitlength of the key $K_{c,1}$. Intuitively, this game challenges the capacity of the adversary to impersonate a user owning a Matching Reference for a given property p_*. The adversary receives a handshake message and is then required to distinguish the key linked to a successful detection of p_* from a random bitstring of the same length.

Unlinkability of Users Consider an adversary $\mathcal{A}$ whose goal is – given any two protocol instances – to determine whether they were generated by the same user. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$; $\mathcal{A}$ is then challenged as follows: he engages in two instances of Handshake with the challenger and is required to tell whether or not the challenger was impersonating the same user over the two Handshake instances.

Unlinkability of Properties Consider an adversary $\mathcal{A}$ whose goal is – given any two protocol instances – to determine whether they were generated to prove possession of the same property. $\mathcal{A}$ has access to oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$,$\mathcal{O}_{\mathsf{Revoke}}$; $\mathcal{A}$ chooses a property p_* such that no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted and is then challenged as follows: he engages in two instances of Handshake with the challenger and has to return true if he can decide that during both instances, the challenger has used Credentials for p_*.

5.3 Secret Handshake with Dynamic Controlled Matching and Revocation Support

In this Section we describe a Secret Handshake scheme supporting Dynamic Controlled Matching, reusable Credentials and revocation. The scheme fulfills the security requirements outlined in the previous Section, as we shall see later in this Chapter.

5.3.1 An overview of the solution

In this Section we give the reader an insight on the choices behind the actual design of the scheme and their rationale.

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 $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 [AKB07] for more details).

Our starting objective is to design a scheme that helps a prover convince a verifier that he owns the Credential for a property; however, the verification will be successful only for entitled verifiers. In addition, we also want to support revocation of Credentials.

To this end, we need some means of secretly "labeling" each Credential, so that we can later on reveal the label and use it as a handle to refuse handshake instances embedding it.

Let us assume that g and $\tilde{g}$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. Also, $t \in \mathbb{Z}_q^*$ is a master secret. Next, we describe how we represent strings into group elements. Following [BB04; Wat05], let $\tilde{g} \overset{R}{\leftarrow} \mathbb{G}_2$; let us also choose $n+1$ random values $\{y_i\}_{i=0}^n \overset{R}{\leftarrow} \mathbb{Z}_q^*$; we assign $\tilde{g}_0 = \tilde{g}^{y_0}, \tilde{g}_1 = \tilde{g}^{y_1}, \ldots, \tilde{g}_n = \tilde{g}^{y_n}$. 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 $H(v) = \tilde{g}_0 \prod_{i \in V(v)} \tilde{g}_i = \tilde{g}^{h(v)} \in \mathbb{G}_2$.

Then, given a property p, Matching References can be formed as $H(p)^t = \tilde{g}^{h(p)t} \in \mathbb{G}_2$ and given to verifiers; in order to successfully authenticate as a possessor of property p, a prover must then prove knowledge of $g^{h(p)t} \in \mathbb{G}_1$. However, instead of simply giving that value to the prover, we pick a random value $x \in \mathbb{Z}_q^*$, different for every Credential, and give x and $g^{(x+h(p)t)}$ to the prover. $g^{(x+h(p)t)}$ is the Credential and x is the aforementioned tag, called *Identification Handle* in the rest of this Chapter, used to identify Credentials.

Then, a prover can be authenticated by a verifier as follows: the verifier sends a challenge $\tilde{g}^m$ and receives $\langle g^{r(x+h(p)t)}, g^r \rangle$ from the prover, where r is a random number, used by the prover to salt the handshake message. The prover can compute $K = (\hat{e}(g, \tilde{g})^m)^{rx}$ and the verifier can compute $K' = \left(\hat{e}\left(g^{r(x+h(p)t)}, \tilde{g}\right) / \hat{e}\left(g^r, \tilde{g}^{h(p)t}\right)\right)^m$; if the authentication is successful, K and K' are the same.

If the Credential is to be revoked at some point, all we need to do is reveal $\tilde{g}^x$, called *Revocation Handle* in the rest of the Chapter. This way, the verifier can verify if the Credential used by the prover has been revoked, by checking if $\hat{e}\left(g^{r(x+h(p)t)}, \tilde{g}\right) = \hat{e}\left(g^r, \tilde{g}^{h(p)t} \cdot \tilde{g}^x\right)$ holds.

Two challenges arise: first, it should be impossible to use the value x in order to trace Credentials before they have been revoked; and second, a user should be forced to send Credentials unmodified. The solution presented above respects the privacy of users: prior to the revocation of a given Credential, an attacker cannot use the Identification Handle to link two different instances of the handshake to the same user: it is easy to show that linking the same x through subsequent instances of the protocol, is equivalent to solving DDH in $\mathbb{G}_1$. We remind the reader that, under the SXDH assumption, the DDH problem is still hard in the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, despite the existence of a computable pairing operation.

However this solution still does not prevent an attacker from modifying Credentials in order to circumvent revocation. In order to prevent this attack, we also introduce another public parameter $W = g^w$, where $w \overset{R}{\leftarrow} \mathbb{Z}_q^*$ is kept secret. Each Credential is raised to the power of the product zw, z being a random number different for every Credential, to yield a randomized version of the Credentials such as $g^{zw(x+h(p)t)}$; in addition, the prover also receives $\tilde{g}^{(zw)^{-1}}$ and $\tilde{g}^{z^{-1}}$. The verifier then computes $K = \left(\hat{e}\left(g^{rzw(x+h(p)t)}, \tilde{g}^{(zw)^{-1}}\right) / \hat{e}\left(g^r, \tilde{g}^{h(p)t}\right)\right)^m$. In addition we require the verifier to also check that $\hat{e}\left(g, \tilde{g}^{z^{-1}}\right) = \hat{e}\left(W, \tilde{g}^{(zw)^{-1}}\right)$.

The protocol introduced in the next Section is not very different from the simple one that we proposed here. The only modifications are an additional random number used to also salt the terms $\tilde{g}^{z^{-1}}$ and $\tilde{g}^{(zw)^{-1}}$, which would otherwise not be randomized and open up to tracing attacks. Finally, the master secret t is substituted by a function $f(p)$ to the same end.

5.3.2 Description of the Scheme

In this Section we introduce the Secret Handshake scheme. The active parties are essentially users and a trusted entity that we will call certification authority (CA). Users receive from the CA 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.
The scheme is composed of the following algorithms:

- Setup according to the security parameter k, the CA chooses g and $\tilde{g}$, randomly selected generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The CA also picks $w \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $W \leftarrow g^w$. Finally the CA picks $\{y_i\}_{i=0}^n \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and assigns $\tilde{g}_0 \leftarrow \tilde{g}^{y_0}, \tilde{g}_1 \leftarrow \tilde{g}^{y_1}, \ldots, \tilde{g}_n \leftarrow \tilde{g}^{y_n}$. The CA chooses as well a function $f : \mathcal{P} \rightarrow \mathbb{Z}_q^*$, where $\mathcal{P}$ is the set of all properties in the system. f is implemented maintaining a list of pairs $(p \in \mathcal{P}, f(p) \in \mathbb{Z}_q^*)$, 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 $f(p) = r$. The system's public parameters are $\{q, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, W, \tilde{g}_0, \ldots, \tilde{g}_n, \hat{e}\}$. The values $w, y_0, \ldots, y_n$ and the function $f(p)$ are instead kept secret by the CA;

- Certify upon user request, the CA verifies that the supplicant user $u \in \mathcal{U}$ possesses the property $p \in \mathcal{P}$ whose possession will later be proved during the execution of the protocol; 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 $x_{u,p} \overset{R}{\leftarrow} \mathbb{Z}_q^*$, 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 $C_{u,p,1} = g^{zw(x_{u,p}+f(p)h(p))}$, $C_{u,p,2} = \tilde{g}^{(zw)^{-1}}$ 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 $g^{f(p)}$. The user verifies that $\hat{e}(C_{u,p,1}, C_{u,p,2})$ $= \hat{e}(g^{x_{u,p}}, \tilde{g}) \cdot \hat{e}(g^{f(p)}, H(p))$;

- Grant upon a user's request, the CA verifies that – according to the policies of the system – user u 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 $match_p = \tilde{g}^{h(p)f(p)}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f(p)}$. The user verifies that $\hat{e}(g, match_p) = \hat{e}(g^{f(p)}, H(p))$;

- 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 $m \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and produces $\tilde{g}^m$;
  - Handshake.RandomizeCredentials the user picks $r, s \overset{R}{\leftarrow} \mathbb{Z}_q^*$; 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 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

$$\hat{e}\left((C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}\right) = \hat{e}\left(g^r, match_p \cdot rev\right) \tag{5.1}$$

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 $\langle g^r, (C_{u,p,1})^{rs}, \ (C_{u,p,2})^{s^{-1}}, (C_{u,p,3})^{s^{-1}}\rangle$. The user checks whether

$$\hat{e}\left(g, (C_{u,p,3})^{s^{-1}}\right) = \hat{e}\left(W, (C_{u,p,2})^{s^{-1}}\right) \tag{5.2}$$

and computes

$$K = \left(\frac{\hat{e}\left((C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}\right)}{\hat{e}\left(g^r, match_p\right)}\right)^m \tag{5.3}$$

$match_p$ 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 5.4 shows how the handshake is carried out.

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.

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 5.2 holds, otherwise abort
Alice :	check that Equation 5.1 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 5.2 holds, otherwise abort
Bob :	check that Equation 5.1 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

Figure 5.4: Secret Handshake with Dynamic Controlled Matching.

Notice that if a user does not have the correct Matching Reference for the received Credential, Handshake.CheckRevoked would fail altogether; however in this case Handshake.Match would also fail and the receiving user would discard the handshake instance anyway. It is clear that after revocation, Credentials can be traced *only* by users that possess the Matching Reference for the property object of that Credential; these users were already potentially able to match the given Credential. For other users, past and future transcripts of handshake instances produced from that Credential are still anonymous and unlinkable.

5.3.3 Security Analysis

This Section analyzes the protocol presented in the previous Section with respect to the requirements outlined in Section 5.2.2. Notice that the proofs do not rely on random oracles, albeit the function f 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 $g^{f(p)}$ 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.

Before proceeding further, we state a well-known hard problem:

Definition 12 (Hardness of the *Decisional Diffie-Hellman* Problem)**.** *We say that the Decisional Diffie-Hellman Problem (DDH) is hard if, for all probabilistic, polynomial-time algorithms* $\mathcal{B}$*,*

$$\mathsf{AdvDDH}_B := Pr[\mathcal{B}(g, g^a, g^b, g^x) = \top \textit{ if } x = ab] - \tfrac{1}{2}$$

is negligible in the security parameter. We assume a random choice of $g \in \mathbb{G}_2$*,* a*,* $b \in \mathbb{Z}_q^*$*;* x *is equal to* ab *with probability* $\frac{1}{2}$ *and is otherwise equal to a random value in* $\mathbb{Z}_q^*/\{ab\}$ *with the same probability.*

We also introduce a new intractability assumption.

Definition 13. *[Hardness of the SM Problem] Let* $w, y, m \in \mathbb{Z}_q^*$*, let* g *be a generator of* $\mathbb{G}_1$ *and* $\tilde{g}$ *be a generator of* $\mathbb{G}_2$*. Let oracle* $O_{w,y}(\cdot)$ *take input* $x \in \mathbb{Z}_q^*$ *and produce output* $g^{zw(x+y)}$*,* $\tilde{g}^{z^{-1}}$ *and* $\tilde{g}^{(zw)^{-1}}$ *where* z *is randomly drawn from* $\mathbb{Z}_q^*$ *upon each oracle query. We say that the SM Problem is hard if, for all probabilistic, polynomial-time algorithms* $\mathcal{A}$*,*

$$\mathsf{AdvSM}_{\mathcal{A}} := Pr[\mathcal{A}(g, g^w, \tilde{g}, \tilde{g}^{w^{-1}}, \tilde{g}^y, \tilde{g}^m, O_{w,y}) = a, a^{sw(x_*+y)}, \tilde{g}^{(sw)^{-1}}, \tilde{g}^{(s)^{-1}}, \hat{e}\,(a, \tilde{g})^{mx_*}]$$

such that $x_* \notin \mathcal{O}$*, is negligible in the security parameter;* $a \in \mathbb{G}_1$*.* $\mathcal{O}$ *is the set of queries* $\mathcal{A}$ *makes to oracle* $O_{w,y}$*. This probability is taken over random choice of* $g \in \mathbb{G}_1$*,* $\tilde{g} \in \mathbb{G}_2$*, and* $w, y, m \in \mathbb{Z}_q^*$*.*

Intuitively, the assumption tells that it is unfeasible to compute a tuple $\left\langle g^{sw(x_*+y)}, \tilde{g}^{(sw)^{-1}}, \tilde{g}^{s^{-1}} \right\rangle$ for a new value x_* and prove knowledge of it, yet having an oracle that can do so for any query. In demonstrating the complexity assumption, we follow the approach presented by Victor Shoup in [Sho97] and extensively used by the research community [ACHdM05; BB08; LRSW99]. As an example, the well known SDH assumption was thus proved by Boneh and Boyen in [BB08].

In what follows we will provide evidence as to the hardness of the problem introduced in Definition 13, by proving a lower bound on the computational complexity under the generic group model. The generic group model is a theoretical framework for the analysis of the success of algorithms in groups where the representation of the elements reveals no information to the attacker. The most popular is the one presented by Victor Shoup [Sho97]. In this model, the attacker is not given direct access to group elements, but rather to the images of group elements under a random one-to-one mapping. The only operations the attacker can perform are therefore equality testing by a bitwise comparison on the images. Group operations can be computed by the attacker through a series of oracles. It is clear that in this situation, the attacker can gain no advantage in solving a computational problem from the representation of the group element.

Internally, the simulator represents the elements of $\mathbb{G}_1$ as their discrete logarithms relative to a chosen generator. To represent the images of the elements of $\mathbb{G}_1$ for the attacker, we use a random one-to-one mapping $\xi_1 : \mathbb{Z}_q^* \to \{0,1\}^{\lceil log_2 q \rceil}$, where q is the group order. For instance, the group element g^a is represented internally as a, whereas the attacker is given the external string representation $\xi_1(a) \in \{0,1\}^{\lceil log_2 q \rceil}$. We similarly define a second mapping $\xi_2 : \mathbb{Z}_q^* \to \{0,1\}^{\lceil log_2 q \rceil}$ to represent $\mathbb{G}_2$, and a third mapping $\xi_T : \to \{0,1\}^{\lceil log_2 q \rceil}$ to represent $\mathbb{G}_T$. The adversary communicates with the oracles using the string representation of the group elements exclusively. The adversary is also given $q = |\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T|$.

The following theorem establishes the unconditional hardness of the SM problem in the generic bilinear group model. Our proof uses a technique similar to the one adopted by Ateniese *et al.* in [ACHdM05].

Theorem 1. *Suppose $\mathcal{A}$ is an algorithm that is able to solve the SM problem in generic bilinear groups of order q, making at most q_G oracle queries for the group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, the oracle $O_{w,y}(\cdot)$ and the bilinear pairing $\hat{e}$, all counted together. Suppose also that the integers $w, y, m \in \mathbb{Z}_q^*$ and the encoding functions ξ_1, ξ_2, ξ_T are chosen at random. Then, the probability ϵ that $\mathcal{A}$ on input $(q, \xi_1(1), \xi_1(w), \xi_2(1), \xi_2(w^{-1}), \xi_2(y), \xi_2(m))$ produces in output $(\xi_1(r), \xi_1(rsw(x_*+y)), \xi_2((sw)^{-1}), \xi_2((s)^{-1}), \xi_T(rx_*m))$ with x_* not previously queried to $O_{w,y}$, is bounded by*

$$\epsilon = O(q_G^2/q)$$

Proof. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$.

$\mathcal{B}$ maintains three lists of pairs $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 1, \ldots, \tau_1\}$, $L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 1, \ldots, \tau_2\}$ and $L_T = \{(F_{T,i}, \xi_{T,i}) : i = 1, \ldots, \tau_T\}$, such that, at step τ in the game, $\tau_1 + \tau_2 + \tau_T = \tau + 6$. The entries $F_{1,i}$, $F_{2,i}$ and $F_{T,i}$ are polynomials with coefficients in $\mathbb{Z}_q^*$. The entries $\xi_{1,i}$, $\xi_{2,i}$, $\xi_{T,i}$ will be all the strings given out to the adversary.

The lists are initialized at step $\tau = 0$ by setting $\tau_1 = 2$, $\tau_2 = 4$, $\tau_T = 0$ and assigning $F_{1,1} = 1$, $F_{1,2} = W$, $F_{2,1} = 1$, $F_{2,2} = W^{-1}$ $F_{2,3} = Y$ and $F_{2,4} = M$ where W, Y and M are indeterminants. The corresponding $\xi_{1,\cdot}$ and $\xi_{2,\cdot}$ are set to random distinct strings. In what follows we describe how $\mathcal{B}$ answers $\mathcal{A}$'s query:

Group operations : $\mathcal{A}$ may request a group operation in $\mathbb{G}_1$ as a multiplication or as a division. Before answering a $\mathbb{G}_1$ query, the simulator $\mathcal{B}$ starts by incrementing the τ_1 counter by one. $\mathcal{A}$ gives $\mathcal{B}$ two operands $\xi_{1,i}$, $\xi_{1,j}$ with $1 \le i, j < \tau_1$, and a multiply/divide selection bit. To respond, $\mathcal{B}$ creates a polynomial $F_{1,\tau_1} \leftarrow F_{1,i} \pm F_{1,j}$; the sign depends on the multiply/divide bit (plus in case of multiply, minus in case of divide). If the result is identical to an earlier polynomial $F_{1,l}$ for some $l < \tau_1$, the simulator $\mathcal{B}$ duplicates its string representation $\xi_{1,\tau_1} \leftarrow \xi_{1,l}$; otherwise, it lets ξ_{1,τ_1} be a fresh random string in $\{0,1\}^{\lceil log_2 q \rceil}$, distinct from $\xi_{1,1}, \ldots, \xi_{1,\tau_1 - 1}$. The simulator appends the pair $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to the list L_1 and gives the string ξ_{1,τ_1} back to $\mathcal{A}$. Group operation queries in $\mathbb{G}_2$ and $\mathbb{G}_T$ are answered in a similar way, based on the lists L_2 and L_T respectively.

Pairing : a pairing query consists of two operands $\xi_{1,i}$ and $\xi_{2,j}$ with $1 \le i \le \tau_1$ and $1 \le j \le \tau_2$ for the current values of τ_1 and τ_2. Upon receipt of such a query from $\mathcal{A}$, the counter τ_T is incremented. The simulator then computes the product of polynomials $F_{T,\tau_T} \leftarrow F_{1,i} \cdot F_{2,j}$. If the same polynomial was already present in L_T, i.e., if $F_{T,\tau_T} = F_{T,l}$ for some $l < \tau_T$, then $\mathcal{B}$ simply clones the associated string $\xi_{T,\tau_T} \leftarrow \xi_{T,l}$, otherwise it sets ξ_{T,τ_T} to a new random string in $\{0,1\}^{\lceil log_2 q \rceil}$, distinct from $\xi_{T,1}, \ldots, \xi_{1,\tau_T - 1}$. The simulator then adds the pair $(F_{T,\tau_T}, \xi_{T,\tau_T})$ to the list L_T, and gives the string ξ_{T,τ_T} to $\mathcal{A}$.

Oracle O : let τ_O be a counter initialized to 0 and $\mathcal{O}$ an empty set. At the beginning of any oracle query, $\mathcal{A}$ inputs $x \in \mathbb{Z}_q^*$; to start, $\mathcal{B}$ adds x to the set $\mathcal{O}$ and increments the counter τ_1 and τ_O by one, and the counter τ_2 by two, choosing a *new* indeterminant Z_{τ_O}; it then sets $F_{1,\tau_1} \leftarrow Z_{\tau_O} W(x + Y)$; it also sets $F_{2,\tau_2 - 1} \leftarrow Z_{\tau_O}^{-1}$ and $F_{2,\tau_O} \leftarrow (Z_{\tau_O} W)^{-1}$. If the same polynomials were already present in L_1 or L_2, i.e., if $F_{1,\tau_1} = F_{1,l}$ for some $l < \tau_1$, or, for $j \in \{0,1\}$, $F_{2,\tau_2 - j} = F_{2,l'}$ for some $l' < \tau_2$, then $\mathcal{B}$ simply clones the associated string $\xi_{1,\tau_1} \leftarrow \xi_{T,l}$, $\xi_{2,\tau_2 - j} \leftarrow$

$\xi_{2,l'}$; otherwise it sets the strings ξ_{1,τ_1} and ξ_{2,τ_2-j} to distinct random values in $\{0,1\}^{\lceil log_2 q \rceil}$, different from the other strings already contained in the lists. The simulator then adds the pairs $(F_{1,\tau_1}, \xi_{1,\tau_1})$ to the list L_1 and $(F_{2,\tau_2-j}, \xi_{2,\tau_2-j})$ to the list L_2, giving the strings ξ_{1,τ_1} and ξ_{2,τ_2-j} to $\mathcal{A}$.

We assume that the SXDH assumption holds, therefore we do not create any isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ or vice versa.

When $\mathcal{A}$ terminates, it returns the tuple $\langle \xi_{1,\alpha}, \xi_{1,\beta}, \xi_{2,\gamma}, \xi_{2,\delta}, \xi_{T,k} \rangle$ where $1 \le \alpha, \beta, \le \tau_1$, $1 \le \gamma, \delta \le \tau_2$ and $1 \le k \le \tau_T$. Let $F_{1,\alpha}$, $F_{1,\beta}$, $F_{2,\gamma}$, $F_{2,\delta}$ and $F_{T,k}$ be the corresponding polynomials in the lists L_1, L_2 and L_T, and g^α, g^β, $\tilde{g}^\gamma$, $\tilde{g}^\delta$, $\hat{e}(g,\tilde{g})^k$ the corresponding elements in $\mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_T$.

In order to exhibit the correctness of $\mathcal{A}$'s answer, $\mathcal{B}$ should check that the system of equation

$$\begin{cases} \left(\dfrac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, \tilde{g}^y)}\right)^m = \hat{e}(g,\tilde{g})^k & (5.4) \\ \dfrac{\hat{e}(g^w, \tilde{g}^\gamma)}{\hat{e}(g, \tilde{g}^\delta)} = 1 & (5.5) \end{cases}$$

is verified. Let us set $\alpha = r$, $k = r x_* m$ and $\delta = s^{-1}$, for some integers $r, x_*, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. If the above system is verified, we can rewrite $g^\alpha = g^r$, $g^\beta = g^{rsw(x_*+y)}$, $\tilde{g}^\gamma = \tilde{g}^{(sw)^{-1}}$, $\tilde{g}^\delta = \tilde{g}^{(s)^{-1}}$, $\hat{e}(g,\tilde{g})^k = \hat{e}(g,\tilde{g})^{r x_* m}$; if $x_* \notin \mathcal{O}$ the attacker has produced a valid answer, according to Definition 13.

In order to verify the system above within the simulation framework, $\mathcal{B}$ computes

$$\begin{cases} F_{T,*} = (F_{1,\beta} \cdot F_{2,\gamma} - F_{1,\alpha} \cdot Y) M - F_{T,K} & (5.6) \\ F_{T,\circ} = F_{2,\gamma} \cdot W - F_{2,\delta} & (5.7) \end{cases}$$

To proceed with our demonstration, first of all we show that it is not possible that $F_{T,*} = F_{T,\circ} = 0$ for every value of W, Y, M and Z_i, $1 \le i \le \tau_O$. This result implies that the success of $\mathcal{A}$ in the game must depend on the particular values assigned to W, Y, M and Z_i.

Let us first observe that the polynomials $F_{1,\alpha}$, $F_{1,\beta}$ are by construction formed as

$$F_{1,\alpha} = \alpha_0 + \alpha_1 W + \sum_{i=1}^{\tau_O} (\alpha_{2,i} Z_i W (x_i + Y))$$

$$F_{1,\beta} = \beta_0 + \beta_1 W + \sum_{i=1}^{\tau_O} (\beta_{2,i} Z_i W (x_i + Y))$$

where x_i is the element of $\mathbb{Z}_q^*$ queried upon the i-th query to the oracle O. The polynomials $F_{2,\gamma}$ and $F_{2,\delta}$ instead are formed as

$$F_{2,\gamma} = \gamma_0 + \gamma_1 W^{-1} + \gamma_2 Y + \gamma_3 M + \sum_{i=1}^{\tau_O} (\gamma_{4,i} Z_i^{-1} + \gamma_{5,i} (Z_i W)^{-1})$$
$$F_{2,\delta} = \delta_0 + \delta_1 W^{-1} + \delta_2 Y + \delta_3 M + \sum_{i=1}^{\tau_O} (\delta_{4,i} Z_i^{-1} + \delta_{5,i} (Z_i W)^{-1})$$

Plugging these equations back in Equation 5.7, gives us

$$\delta_0 + \delta_1 W^{-1} + \delta_2 Y + \delta_3 M + \sum_{i=1}^{\tau_O} (\delta_{4,i} Z_i^{-1} + \delta_{5,i} (Z_i W)^{-1}) =$$
$$\gamma_0 W + \gamma_1 + \gamma_2 YW + \gamma_3 MW + \sum_{i=1}^{\tau_O} (\gamma_{4,i} Z_i^{-1} W + \gamma_{5,i} (Z_i)^{-1}) \tag{5.8}$$

If the attacker wins the game, Equation 5.8 must be symbolically equal to zero; simplifying all the unique terms, we are left with

$$\delta_0 + \sum_{i=1}^{\tau_O} (\delta_{4,i} Z_i^{-1}) = \gamma_1 + \sum_{i=1}^{\tau_O} (\gamma_{5,i} (Z_i)^{-1}) \tag{5.9}$$

from which we conclude that $F_{2,\gamma} = \gamma_1 W^{-1} + \sum_{i=1}^{\tau_O} (\gamma_{5,i} (Z_i W)^{-1})$.

Let us now consider Equation 5.6, which can be rewritten as

$$\Bigg(\gamma_1 \beta_0 W^{-1} + \gamma_1 \beta_1 + \sum_{i=1}^{\tau_O} (\gamma_1 \beta_{2,i} Z_i (x_i + Y)) +$$
$$+ \sum_{i=1}^{\tau_O} \left(\beta_0 \gamma_{5,i} (Z_i W)^{-1} + \beta_1 \gamma_{5,i} Z_i^{-1} + \sum_{j=1}^{\tau_O} \left(\beta_{2,j} \gamma_{5,i} (Z_i)^{-1} Z_j (x_j + Y) \right) \right) -$$
$$+ \left(\alpha_0 Y + \alpha_1 WY + \sum_{i=1}^{\tau_O} (\alpha_{2,i} Z_i WY (x_i + Y)) \right) \Bigg) M = F_{T,K} \tag{5.10}$$

If the attacker wins the game, Equation 5.10 must be symbolically equal to zero.

First of all, the right hand of the equation is an element of the group $\mathbb{G}_T$; the adversary can receive elements in $\mathbb{G}_T$ only through invocations of the Pairing oracle. We recall that this oracle returns a product of an element in $\mathbb{G}_1$ multiplied by an element in $\mathbb{G}_2$. Second, we notice that each term of the left hand of the equation contains M.

Therefore, from $F_{T,K}$ we delete all the terms that do not contain M. Then, we simplify M on both sides, ending up with the following equation

$$\begin{aligned}
&\left(\gamma_1\beta_0 W^{-1} + \gamma_1\beta_1 + \sum_{i=1}^{\tau_O}(\gamma_1\beta_{2,i}Z_i(x_i+Y)) + \right.\\
&+\sum_{i=1}^{\tau_O}\left(\beta_0\gamma_{5,i}(Z_iW)^{-1} + \beta_1\gamma_{5,i}Z_i^{-1} + \sum_{j=1}^{\tau_O}\left(\beta_{2,j}\gamma_{5,i}(Z_i)^{-1}Z_j(x_j+Y)\right)\right) - \\
&\left. + \left(\alpha_0 Y + \alpha_1 WY + \sum_{i=1}^{\tau_O}(\alpha_{2,i}Z_iWY(x_i+Y))\right)\right) = \qquad (5.11)\\
&= k_0 + k_1 W + \sum_{i=1}^{\tau_O}(k_{2,i}Z_iW(x_i+Y)) \qquad (5.12)
\end{aligned}$$

Further more, we simplify all the unique terms, as follows: $\gamma_1\beta_0 = 0$ since it is the only term containing W^{-1}; $\beta_0\gamma_{5,i} = 0$ since it is the only term containing $(Z_iW)^{-1}$; $\beta_1\gamma_{5,i}$ since it is the only term containing Z_i^{-1}; $\alpha_1 = \alpha_{2,i} = 0$ since they are the only terms containing WY and Z_iWY^2 respectively; $k_1 = k_{2,i} = 0$ since the are the only coefficients of W and of Z_iW respectively; finally $\gamma_1\beta_{2,i} = 0$ since it is the only term containing Z_iY. Notice also that all the terms $\beta_{2,j}\gamma_{5,i}$ must be equal to zero for all $i \neq j$ since they would contain terms in $(Z_i)^{-1}Z_j$.

We must also simplify the terms in $\gamma_1\beta_1$: indeed we have previously set the coefficient $\beta_1\gamma_{5,i} = 0$, but we cannot set $\gamma_{5,i} = 0$ since this would unduly cancel out terms in $\beta_{2,i}\gamma_{5,i}(Z_i)^{-1}Z_i(x_i+Y) = \beta_{2,i}\gamma_{5,i}(x_j+Y)$ which instead can be balanced out with linear combinations of terms in k_0 and $\alpha_0 Y$. Therefore the term $\beta_1 = 0$.

By simplifying these terms, we end up with

$$\beta_{2,i}\gamma_{5,i}(x_i+Y) - \alpha_0 Y = k_0 \qquad (5.13)$$

Now, $\alpha_0 = \beta_{2,i}\gamma_{5,i}$ since they are the only coefficients of Y. Then $k_0 = \beta_{2,i}\gamma_{5,i}x_i$. However this is not a valid solution, x_i is the i-th value queried to oracle O, and thus belongs to $\mathcal{O}$. We therefore conclude that it is impossible for the attacker to win the game for every value of W, Y, M and Z_i; instead this depends on a lucky instantiation of such variables.

The simulator $\mathcal{B}$ therefore chooses random values $\bar{w}, \bar{y}, \bar{m}, \bar{z}_1, \ldots, \bar{z}_{\tau_O}$ for each of the variables W, Y, M and Z_i. Let us analyze the probability that the attacker has won the game given the chosen assignment of the variables: this happens if (i) no two non-identical polynomials in the lists L_1, L_2 and L_T assume the same value and (ii) if the assignment satisfies $F_{T,*} = F_{T,\circ} = 0$. If (i) is true, $\mathcal{B}$'s simulation was flawed because two group elements – that were equal – have been presented as distinct to the attacker.

Summing up, the probability of success of the attacker is bounded by the probability that any of the following equations holds:

$$F_{1,i}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) - F_{1,j}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) = 0, \quad i,j \text{ s.t. } F_{1,i} \neq F_{1,j} \tag{5.14}$$
$$F_{2,i}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) - F_{2,j}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) = 0, \quad i,j \text{ s.t. } F_{2,i} \neq F_{2,j} \tag{5.15}$$
$$F_{T,i}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) - F_{T,j}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) = 0, \quad i,j \text{ s.t. } F_{T,i} \neq F_{T,j} \tag{5.16}$$
$$F_{T,*}(\bar{w},\bar{y},\bar{m},\bar{z}_1,\dots,\bar{z}_{\tau_O}) = 0 \tag{5.17}$$
$$F_{T,\circ}(\bar{w},\bar{y},\bar{m},\dots,\bar{z}_{\tau_O}) = 0 \tag{5.18}$$

For fixed i,j each non-trivial polynomial 5.14, 5.15, 5.16 has degree at most 1 and it vanishes with probability $\leq 1/q$. Polynomials 5.17 and 5.18 have too degree at most 1 and vanish with probability $\leq 1/q$. We sum over all the (i,j) to bound the overall success probability ϵ of the attacker $\mathcal{A}$ as $\epsilon \leq \binom{\tau_1}{2}\frac{1}{q} + \binom{\tau_2}{2}\frac{1}{q} + \binom{\tau_T}{2}\frac{1}{q} + \frac{2}{q}$. Since $\tau_1 + \tau_2 + \tau_T \leq q_G + 5$, we end up with

$$\epsilon \leq \frac{(q_G+5)^2}{2q} + \frac{2}{q} = O(q_G^2/q)$$

□

5.3.3.1 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 $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. $\mathcal{A}$ 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.

Lemma 6. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvTraceProperty}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{TraceProperty}] - \tfrac{1}{2}$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game TraceProperty to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge; $f(p)$ is implemented as follows: on a query for $f(p)$, if p has never been queried before, $\mathcal{B}$ picks a random value $r_p \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p, r_p) in a table. Then $\mathcal{B}$ flips a random biased coin $guess(p) \in \{0, 1\}$ biased as follows: $guess(p)$ equals 1 with probability δ and is equal to 0 with probability $1-\delta$. $\mathcal{B}$ answers as follows: if $guess(p) = 0$, he looks up r_p in the table and answers with r_p. Instead, if $guess(p) = 1$, $\mathcal{B}$ answers with $f(p) = br_p$; finally, $\mathcal{B}$ picks and publishes the other public parameters according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: the adversary submits a query to receive a Matching Reference for property p_i; assuming that $guess(p_i) = 0$, $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_* for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted. Let us assume $guess(p_*) = 1$. $\mathcal{B}$ engages in two instances of Handshake with $\mathcal{A}$; in particular, $\mathcal{B}$ receives two nonces $\tilde{g}^{m_1}$ and $\tilde{g}^{m_2}$; $\mathcal{B}$ then picks $x_1, x_2, s_1, s_2, r \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and $p_* \overset{R}{\leftarrow} \mathcal{P}$; $\mathcal{B}$ generates two handshake tuples as follows:

$$\left\langle g^r, g^{rs_1 w(x_1 + br_{p_*} h(p_*))}, \tilde{g}^{(s_1 w)^{-1}}, \tilde{g}^{(s_1)^{-1}} \right\rangle$$
$$\left\langle g^a, g^{as_2 w x_2} g^{\sigma s_2 w r_{p_*} h(p_*)}, \tilde{g}^{(s_2 w)^{-1}}, \tilde{g}^{(s_2)^{-1}} \right\rangle$$

and also gives $\mathcal{A}$ the two keys that are generated by Handshake.RandomizeCredentials, $(\hat{e}(g, \tilde{g}^{m_1}))^{rx_1}$ and $(\hat{e}(g^a, \tilde{g}^{m_2}))^{x_2}$.

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

$$\begin{cases} (r(x_1 + br_{p_*} h(p_*)) - r\alpha) m_1 = r m_1 x_1 \\ (a x_2 + \sigma r_{p_*} h(p_*) - a\alpha) m_2 = a m_2 x_2 \end{cases} \tag{5.19}$$

If $\mathcal{A}$'s answer is positive, it means that the system of equations is verified. Then we can solve the first equation as $\alpha = br_{p_*}h(p_*)$, and plugging in the second equation $\mathcal{B}$ can verify that $\sigma = ab$, which is the positive answer to the DDH problem. If not, $\mathcal{B}$ can give the negative answer to DDH.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(p) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $p \neq p_*$, then the execution environment is indistinguishable from the actual game TraceProperty. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(p) = 0 \text{ for all } p \neq p_*] = \delta \cdot (1 - \delta)^{\mathcal{Q}_p} \tag{5.20}$$

where $\mathcal{Q}_p$ is the number of different properties $\mathcal{A}$ queries to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_p}$ we know that the probability in 5.20 is greater than $\frac{1}{e \cdot \mathcal{Q}_p}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvTraceProperty}_{\mathcal{A}}$ as $\mathsf{AdvTraceProperty}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_p \cdot \mathsf{AdvDDH}_{B}$. □

5.3.3.2 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 $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. 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:

- one where a user uses Credentials that have not yet been revoked, for which the adversary has a corresponding Matching Reference;
- a second situation in which 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.

Lemma 7. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvTraceUser1}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{TraceUser1}] - \frac{1}{2}$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game TraceCredential to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge, picks and publishes the public parameters according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_*; it is worth noting that $\mathcal{A}$ does not have any restriction on the choice of p_*. $\mathcal{B}$ then engages in two instances of Handshake with $\mathcal{A}$; in particular, $\mathcal{B}$ receives two nonces $\tilde{g}^{m_1}$ and $\tilde{g}^{m_2}$; $\mathcal{B}$ then picks $r, s_1, s_2 \xleftarrow{R} \mathbb{Z}_q^*$ and prepares two handshake tuples as follows:

$$\left\langle g^r, g^{rs_1w(b+h(p_*)f(p_*))}, \tilde{g}^{(s_1w)^{-1}}, \tilde{g}^{(s_1)^{-1}} \right\rangle$$

$$\left\langle g^a, g^{s_2w\sigma} g^{as_2wh(p_*)f(p_*)}, \tilde{g}^{(s_2w)^{-1}}, \tilde{g}^{(s_2)^{-1}} \right\rangle$$

and also sends $\mathcal{A}$ the two keys that are generated by Handshake.RandomizeCredentials, $\left(\hat{e}\left(g^{b}, \tilde{g}^{m_1}\right)\right)^{r}$ and $\hat{e}\left(g^{\sigma}, \tilde{g}^{m_2}\right)$.

Analysis of $\mathcal{A}$'s answer It is straightforward to verify that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can give the same answer to solve the DDH problem. Indeed, if $\mathcal{A}$ wins the game, he is able to tell if both handshake messages contain the same Identification Handle x_*. Let us assume this is the case. Then, the same Revocation Handle $rev_* = \tilde{g}^{x_*}$ can be used to revoke both Credentials. Then, performing a check as described in Equation 5.1, the following system

$$\begin{cases} r(b + h(p_*)f(p_*)) - rh(p_*)f(p_*) = rx_* \\ \sigma + ah(p_*)f(p_*) - ah(p_*)f(p_*) = ax_* \end{cases} \tag{5.21}$$

should hold.

Then we can solve the first equation as $x_* = b$, and plugging in the second equation $\mathcal{B}$ can verify that $\sigma = ab$, which is the positive answer to the DDH problem. If not, $\mathcal{B}$ can give the negative answer to DDH. $\square$

Now we turn our attention to TraceUser2. This game features an attacker trying to trace users under the following circumstances: the user has Revocation Handles for the Credentials used by the challenger, but does not have the corresponding Matching References. As a consequence of this last factor, the challenger will not give the keys generated by Handshake.RandomizeCredentials as it was the case instead in TraceUser1. The reasons for this are twofold: first of all, as we shall see in Lemma 9, without Matching Reference an attacker is not able to distinguish the correct key from a random value. Second, if the attacker was equipped with the handshake tuple SH, the Revocation Handle and the correct key, tracing would be feasible: indeed we can rewrite Equation 5.1, the equality whose satisfaction is verified upon execution of Handshake.CheckRevoked, as

$$\begin{aligned} \hat{e}\left(\left(C_{u,p,1}\right)^{rs}, \left(C_{u,p,2}\right)^{s^{-1}}\right) &= \hat{e}\left(g^{r}, match_p\right) \cdot \hat{e}\left(g^{r}, rev\right) \\ \frac{\hat{e}\left(\left(C_{u,p,1}\right)^{rs}, \left(C_{u,p,2}\right)^{s^{-1}}\right)}{\hat{e}\left(g^{r}, match_p\right)} &= \hat{e}\left(g^{r}, rev\right) \\ \left(\frac{\hat{e}\left(\left(C_{u,p,1}\right)^{rs}, \left(C_{u,p,2}\right)^{s^{-1}}\right)}{\hat{e}\left(g^{r}, match_p\right)}\right)^{m} &= \left(\hat{e}\left(g^{r}, rev\right)\right)^{m} \\ K &= \left(\hat{e}\left(g^{r}, rev\right)\right)^{m} \end{aligned}$$

where m is the nonce chosen by the adversary. The last equality shows that, in presence of the correct key, an adversary can check for which Revocation Handle the equality holds, thus being able to perform tracing. So it is clear that – in presence of the correct key and of the Revocation Handle, tracing is possible by definition.

Lemma 8. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvTraceUser2}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{TraceUser2}] - \tfrac{1}{2}$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game $\mathsf{TraceCredential}$ to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge; $f(p)$ is implemented as follows: on a query for $f(p)$, if p has never been queried before, $\mathcal{B}$ picks a random value $r_p \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p, r_p) in a table. Then $\mathcal{B}$ flips a random biased coin $guess(p) \in \{0, 1\}$ biased as follows: $guess(p)$ equals 1 with probability δ and is equal to 0 with probability $1-\delta$. $\mathcal{B}$ answers as follows: if $guess(p) = 0$, $\mathcal{B}$ looks up r_p in the table and answers with r_p. Instead, if $guess(p) = 1$, $\mathcal{B}$ answers with $f(p) = br_p$; finally, $\mathcal{B}$ picks and publishes the other public parameters according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: the adversary submits a query to receive a Matching Reference for property p_i; assuming that $guess(p_i) = 0$, $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_* for which no query to $\mathcal{O}_{\mathsf{Grant}}$ has been submitted; let us assume $guess(p_*) = 1$. $\mathcal{B}$ chooses a user u_*, picks x_{u_*}

and gives $\mathcal{B}$ the Revocation Handle $\tilde{g}^{x_{u*}}$; $\mathcal{B}$ then engages in two instances of $\mathsf{Handshake}$ with $\mathcal{A}$ by picking $r, s_1, s_2 \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and preparing two handshake tuples as follows:

$$\left\langle g^r, g^{rs_1w(x_{u*}+br_{p*}h(p_*))}, \tilde{g}^{(s_1w)^{-1}}, \tilde{g}^{(s_1)^{-1}} \right\rangle$$
$$\left\langle g^a, g^{as_2wx_{u*}}g^{\sigma s_2wr_{p*}h(p_*)}, \tilde{g}^{(s_2w)^{-1}}, \tilde{g}^{(s_2)^{-1}} \right\rangle$$

Analysis of $\mathcal{A}$'s answer It is straightforward to verify that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can give the same answer to solve the DDH problem. Indeed, if $\mathcal{A}$ wins the game, he is able to tell if both handshake messages contain the same Identification Handle x_*. Let us assume this is the case. Then, the same Revocation Handle $rev_* = \tilde{g}^{x_*}$ can be used to revoke both Credentials. Then, performing a check as described in Equation 5.1, the following system

$$\begin{cases} r(x_{u*} + br_{p*}h(p_*)) - rbr_{p*}h(p_*) = rx_* \\ as_2x_{u*} + \sigma r_{p*}h(p_*) - abr_{p*}h(p_*) = ax_* \end{cases} \tag{5.22}$$

should hold.

The system is verified only if $\sigma = ab$, which is the positive answer to the DDH problem. If not, $\mathcal{B}$ can give the negative answer to DDH.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(p) = 0$ for all other queries to oracle $\mathcal{O}_{\mathsf{Grant}}$ such that $p \neq p_*$, then the execution environment is indistinguishable from the actual game $\mathsf{TraceUser2}$. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(p) = 0 \text{ for all } p \neq p_*] = \delta \cdot (1-\delta)^{\mathcal{Q}_p} \tag{5.23}$$

where $\mathcal{Q}_p$ is the number of different properties $\mathcal{A}$ queries to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_p}$ we know that the probability in 5.23 is greater than $\frac{1}{e \cdot \mathcal{Q}_p}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvTraceUser2}_{\mathcal{A}}$ as $\mathsf{AdvTraceUser2}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_p \cdot \mathsf{AdvDDH}_B$. □

5.3.3.3 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, $\mathcal{A}$ can access $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. 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 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.

Lemma 9. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game Detect to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge; $f(p)$ is implemented as follows: on a query for $f(p)$, if p has never been queried before, $\mathcal{B}$ picks a random value $r_p \overset{R}{\leftarrow} \mathbb{Z}_q^*$, storing the pair (p, r_p) in a table. Then $\mathcal{B}$ flips a random biased coin $guess(p) \in \{0,1\}$ biased as follows: $guess(p)$ equals 1 with probability δ and is equal to 0 with probability $1-\delta$. $\mathcal{B}$ answers as follows: if $guess(p) = 0$, $\mathcal{B}$ looks up r_p in the table and answers with $f(p) = r_p$. Instead, if $guess(p) = 1$, $\mathcal{B}$ answers with $f(p) = br_p$; finally, $\mathcal{B}$ picks and publishes the public parameters according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries the challenger to obtain a Matching Reference for property p_i; assuming that $guess(p_i) = 0$, $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_*, such that no query to the oracle $\mathcal{O}_{\mathsf{Grant}}$ has been submitted. Let us assume that $guess(p_*) = 1$: as a consequence, $f(p_*) = br_{p_*}$. $\mathcal{B}$ then engages in an instances of Handshake with

$\mathcal{A}$; in particular, $\mathcal{B}$ receives a nonce $\tilde{g}^m$; $\mathcal{B}$ then picks $x, s \xleftarrow{R} \mathbb{Z}_q^*$ sets $r' = a$ and $x' = -br_{p_*}h(p_*) + x + \sigma a^{-1} r_{p_*} h(p_*)$ and constructs a handshake message as follows:

$$\begin{cases} g^{r'} = g^a \\ g^{r'sw(x'+f(p_*)h(p_*))} = g^{r'sw(x'+br_{p_*}h(p_*))} = g^{aswx} g^{\sigma swr_{p_*}h(p_*))} \\ \tilde{g}^{(sw)^{-1}} \\ \tilde{g}^{s^{-1}} \end{cases}$$

Finally, $\mathcal{A}$ receives the key that Handshake.RandomizeCredentials computes: in order to compute such key, $\mathcal{B}$ assumes that $\sigma = ab$, and therefore computes $\hat{e}\left(g^a, \tilde{g}^m\right)^{x'} = \hat{e}\left(g^a, \tilde{g}^m\right)^{x}$. $\mathcal{A}$ answers the challenge with a bit b; $\mathcal{A}$ wins the game if $b = 0$ if the key is a random bitstring, and $b = 1$ if the key is correct.

Analysis of $\mathcal{A}$'s answer It is straightforward to verify that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can give the same answer to solve the DDH problem. Indeed, if $\mathcal{A}$ wins the game and answers $b = 1$, it means that the key $\mathcal{B}$ generated was correct. Then, the same key must be computable using Equation 5.3 with the Matching Reference for property p_*, $\tilde{g}^{f(p_*)h(p_*)} = \tilde{g}^{br_{p_*}h(p_*)}$.

Then we can write

$$(ax + \sigma r_{p_*}h(p_*) - abr_{p_*}h(p_*))m = max$$

However this equation is satisfied only if $\sigma = ab$, which is the positive answer to the DDH problem. If not, $\mathcal{B}$ can give the negative answer to DDH.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(p) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $p \neq p_*$, then the execution environment is indistinguishable from the actual game Detect. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(p) = 0 \text{ for all } p \neq p_*] = \delta \cdot (1-\delta)^{\mathcal{Q}_p} \tag{5.24}$$

where $\mathcal{Q}_p$ is the number of different properties $\mathcal{A}$ queries to $\mathcal{O}_{\mathsf{Grant}}$ oracle. By setting $\delta \approx \frac{1}{\mathcal{Q}_p}$ we know that the probability in 5.24 is greater than $\frac{1}{e \cdot \mathcal{Q}_p}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvDetect}_{\mathcal{A}}$ as $\mathsf{AdvDetect}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_p \cdot \mathsf{AdvDDH}_B$. □

5.3.3.4 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.

Then the adversary is challenged to engage in the Impersonate game against the attacker. The game is defined as follows: before the beginning of the game, the challenger flips a coin and based on its outcome, plays either the Impersonate1 or the Impersonate2 game. If the adversary wins the game, the challenger gains an advantage on one of the two hard problems: which problem depends on the adversary's behaviour. As the challenger must commit to one game in advance the advantage that the challenger receives is decreased by a factor $\frac{1}{2}$.

Now, let us begin with the description of Impersonate. Let $\mathcal{A}$'s goal be the impersonation of a user owning a Credential for a given property. $\mathcal{A}$ can access $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. $\mathcal{A}$ 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. As mentioned before, we consider a weaker version of Impersonation Resistance than the one adopted in Section 4.4.2.1, where the adversary was not required to compute the key but to distinguish it from a random value.

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 Impersonate.

Now let us show how we construct the two sub-games. 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 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_*.

This means that the challenger can check that

$$\left(\frac{\hat{e}(g^{\beta}, \tilde{g}^{\gamma})}{\hat{e}(g^{\alpha}, match_{p_*})}\right)^m = \left(\frac{\hat{e}(g^{\beta}, \tilde{g}^{\gamma})}{\hat{e}(g^{\alpha}, \tilde{g}^{f(p_*)h(p_*)})}\right)^m = e^k \tag{5.25}$$

and that

$$\hat{e}\left(g, \tilde{g}^{\delta}\right) = \hat{e}\left(g^{w}, \tilde{g}^{\gamma}\right) \tag{5.26}$$

Let us set $\alpha = r$, $k = rx_{u_*,p_*}m$ and $\delta = s^{-1}$, for some integers $r, x_{u_*,p_*}, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, from Equation 5.26 we derive that $\gamma = (ws)^{-1}$ and from Equation 5.25 that $\beta = rsw(x_{u_*,p_*} + f(p_*)h(p_*))$.

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 5.1 of the Handshake.CheckRevoked sub-algorithm. Let us set $X_{u,p} = x_{u,p} + f(p)h(p)$. When the attacker is challenged, we have seen that he produces the value $g^{rsw(x_{u_*,p_*}+f(p_*)h(p_*))} = g^{rsX_{u_*,p_*}}$. If we define the set

$$Q_{\mathcal{A}} = \{X_{u,p} \in \mathbb{Z}_q^* : \mathcal{A} \textit{ has received } g^{zwX_{u,p}}, \tilde{g}^{(zw)^{-1}}, \tilde{g}^{z^{-1}} \textit{ from a query to } \mathsf{Certify}\}$$

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 $X_{u_*,p_*} \in Q_{\mathcal{A}}$.

Lemma 10. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvImpersonate1}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{Impersonate1}]$$

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$'s advantage to solve a given instance of the SM Problem.

Proof. The challenger $\mathcal{B}$ is defined as follows. $\mathcal{B}$ receives an instance $\Big\langle g, g^w, \tilde{g}, \tilde{g}^{w^{-1}}\tilde{g}^{y}$, $\tilde{g}^m$, $O_{w,y}\rangle$ of the SM problem and wishes to use $\mathcal{A}$ to produce the tuple $\langle g^r$, $g^{rsw(x_*+y)}$, $\tilde{g}^{(zw)^{-1}}, \tilde{g}^{s^{-1}}, \hat{e}\left(g, \tilde{g}\right)^{rx_*m}\rangle$, such that x_* has not been queried to O. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ sets public parameters $g, \tilde{g}$ as the ones received from the challenge. $\mathcal{B}$ sets $W \leftarrow g^w$, $\tilde{g}_0 = \tilde{g}^y$. It then picks $\{y_i\}_{i=1}^n \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{g}_1 = \tilde{g}^{y_1}, \ldots, \tilde{g}_n = \tilde{g}^{y_n}$. $\mathcal{B}$ then publishes the public parameter according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Credentials for an arbitrary number of user u_i and property p_i; $\mathcal{B}$ answers by picking a random Identification Handle $x_{u_i,p_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and by giving it to $\mathcal{A}$; $\mathcal{B}$ then queries the oracle $O_{w,y}$ providing $v = \sum_{i \in V(p_i)} y_i + \frac{x_{u_i,p_i}}{f(p_i)}$ as input, and adding the value v to the set $\mathcal{O}$ of queries to oracle O. The output of the oracle is $(g^{zw(\frac{x_{u_i,p_i}}{f(p_i)}+y+\sum_{i\in V(p_i)} y_i)}, \tilde{g}^{(zw)^{-1}}, \tilde{g}^{z^{-1}})$. $\mathcal{B}$ then assigns $C_{u_i,p_i,1} \leftarrow \left(g^{zw(\frac{x_{u_i,p_i}}{f(p_i)}+y+\sum_{i\in V(p_i)} y_i)}\right)^{f(p_i)} = g^{zw(x_{u_i,p_i}+f(p_i)h(p_i))}$, $C_{u_i,p_i,2} \leftarrow \tilde{g}^{(zw)^{-1}}$, $C_{u_i,p_i,3} \leftarrow \tilde{g}^{z^{-1}}$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Matching References for an arbitrary number of properties p_i; $\mathcal{B}$ answers with $match_{p_i} = \tilde{g}^{f(p_i)h(p_i)}$, and also gives to $\mathcal{A}$ the value $g^{f(p_i)}$;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{A}$ queries $\mathcal{B}$ for an arbitrary number of Revocation Handles for user u_i and property p_i; $\mathcal{B}$ answers with $rev_{u_i,p_i} = \tilde{g}^{x_{u_i,p_i}}$;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge $\mathcal{A}$ then declares that this phase of the game is over. $\mathcal{B}$ therefore revokes each of the Credentials $\mathcal{A}$ requested in the previous phase. $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$. $\mathcal{B}$ challenges $\mathcal{A}$ by sending $\tilde{g}^m$ and $\mathcal{A}$ answers the challenge with the tuple $\langle g^\alpha, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta, e^k \rangle$.

Analysis of $\mathcal{A}$'s response If $\mathcal{A}$ wins the game, $\mathcal{B}$ can check that

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, match_{p_*})}\right)^m = \left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}\left(g^\alpha, \tilde{g}^{f(p_*)(y+\sum_{i\in V(p_*)} y_i)}\right)}\right)^m = e^k \tag{5.27}$$

and that

$$\hat{e}\left(g, \tilde{g}^\delta\right) = \hat{e}\left(g^w, \tilde{g}^\gamma\right) \tag{5.28}$$

as mandated by the Handshake.Match sub-algorithm detailed in Section 5.3.2.

Let us set $\alpha = r$, $k = rx_*m$ and $\delta = s^{-1}$, for some integers $r, x_*, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, from Equation 5.28 we derive that $\gamma = (ws)^{-1}$ and from Equation 5.27 that $\beta = rswf(p_*)(\frac{x_*}{f(p_*)} + y + \sum_{i\in V(p_*)} y_i)$. Notice that by the definition of the game, the attacker has not received a Credential containing the term $g^{zwX_{u_*,p_*}} = g^{zw(x_*+f(p_*)h(p_*))}$ from a query to the $\mathcal{O}_{\mathsf{Certify}}$ oracle.

This implies in turn that the value $v_* = \frac{x_*}{f(p_*)} + \sum_{i \in V(p_*)} y_i$ has never been queried by the challenger to the oracle $O_{w,y}$ in the execution of a $\mathcal{O}_{\mathsf{Certify}}$ query: as a consequence, v_* does not belong to the set $\mathcal{O}$. Therefore we conclude that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can provide

$$\left\langle (g^{\alpha})^{f(p_*)}, g^{\beta}, \tilde{g}^{\gamma}, \tilde{g}^{\delta}, \hat{e}\left(g, \tilde{g}\right)^{k} \cdot \hat{e}\left(g^{\alpha}, \tilde{g}^{m}\right)^{f(p_*) \sum_{i \in V(p_*)} y_i} \right\rangle$$

as an answer to the SM problem. □

We now turn our attention to the Impersonate2 game, focusing on an adversary reusing an already received Credential and yet able to circumvent revocation. Given that the revocation check involves the Matching Reference for a property, the only way for the attacker is to use a Credential received for a property – say $p_{\circ}$ – in the attempt to impersonate a different property – say p_*.

Lemma 11. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvImpersonate2}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{Impersonate2}] - \frac{1}{2}$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\left\langle \tilde{g}, \tilde{g}^{a}, \tilde{g}^{b}, \tilde{g}^{\sigma} \right\rangle$ of the DDH problem and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ sets the public parameter $\tilde{g}$ as the ones received from the DDH challenge; it then picks $\{y_i\}_{i=0}^{n} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and sets $\tilde{g}_0 = (\tilde{g}^{a})^{y_0}$, $\tilde{g}_1 = (\tilde{g}^{a})^{y_1}, \ldots \tilde{g}_n = (\tilde{g}^{a})^{y_n}$; notice that with such a setting of parameters, we can write $h'(p) = ah(p)$ where $h(p) = y_0 + \sum_{i \in V(p)} y_i$; the other parameters are then picked and published following the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Credentials for an arbitrary number of pairs $(u, p) \in \mathcal{U} \times \mathcal{P}$; $\mathcal{B}$ answers by first of all picking $x_{u,p} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and by giving it to $\mathcal{A}$. Then, the challenger gives $cred_{u,p} = \langle C_{u,p,1}, C_{u,p,2}, C_{u,p,3} \rangle$ to $\mathcal{A}$, where $C_{u,p,1} = g^{(zw)}$, $C_{u,p,2} = \tilde{g}^{(zw)^{-1}(x_{u,p}+f(p)h'(p))} = \tilde{g}^{(zw)^{-1}(x_{u,p}+f(p)ah(p))}$ and $C_{u,p,3} = \tilde{g}^{z^{-1}(x_{u,p}+f(p)h'(p))} = \tilde{g}^{z^{-1}(x_{u,p}+f(p)ah(p))}$; $\mathcal{A}$ also receives $g^{f(p)}$. This representation of Credentials is indistinguishable from the ones mandated by the protocol: indeed notice that we can set $z = z'(x_{u,p} + f(p)h'(p))$ and we can rewrite $C_{u,p,1} = g^{z'w(x_{u,p}+f(p)h'(p))}$, $C_{u,p,2} = \tilde{g}^{(z'w)^{-1}}$ and $C_{u,p,3} = \tilde{g}^{z'^{-1}}$ which is exactly the way Credentials are formulated according to the algorithm Certify described in Section 5.3.2. $\mathcal{B}$ adds to a list V the tuple $(\tilde{g}^{x_{u,p}+h'(p)f(p)}, u, p, x_{u,p})$ for each query of $\mathcal{A}$ and keeps it for later use.

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Matching References for an arbitrary number of properties p; $\mathcal{B}$ answers with $match_p = \tilde{g}^{f(p)h'(p)} = \tilde{g}^{f(p)ah(p)}$; $\mathcal{A}$ also receives $g^{f(p)}$;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{A}$ queries $\mathcal{B}$ for an arbitrary number of Revocation Handles for user u_i and property p_i; $\mathcal{B}$ answers with $rev_{u_i,p_i} = \tilde{g}^{x_{u_i,p_i}}$;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge $\mathcal{A}$ then declares that this phase of the game is over. $\mathcal{B}$ therefore revokes each Credential requested by $\mathcal{A}$ in the previous phase. $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$. $\mathcal{B}$ challenges $\mathcal{A}$ by sending $\tilde{g}^b$ and $\mathcal{A}$ answers the challenge with the tuple $\langle g^\alpha, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta, \hat{e}(g, \tilde{g})^k \rangle$.

Analysis of $\mathcal{A}$'s response If $\mathcal{A}$ wins the game, $\mathcal{B}$ can check that

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, match_{p_*})}\right)^b = \tag{5.29}$$

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, \tilde{g}^{f(p_*)h'(p_*)})}\right)^b = \tag{5.30}$$

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, \tilde{g}^{af(p_*)h(p_*)})}\right)^b = \hat{e}(g, \tilde{g})^k \tag{5.31}$$

and that

$$\hat{e}\left(g, \tilde{g}^\delta\right) = \hat{e}\left(g^w, \tilde{g}^\gamma\right) \tag{5.32}$$

as mandated by the Handshake.Match sub-algorithm detailed in Section 5.3.2.

Let us set $\alpha = r$, $k = rx_*b$ and $\delta = s^{-1}$, for some integers $r, x_*, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, from Equation 5.32 we derive that $\gamma = (ws)^{-1}$; if we set $v_* = x_* + f(p_*)h'(p_*)$, from Equation 5.31 we can write $\beta = rswv_*$.

We know by definition that the attacker has already received $C_{u_\circ,p_\circ,1} = g^{zwv_*} = g^{zw(x_{u_\circ,p_\circ}+f(p_\circ)h'(p_\circ))}$ during the previous query phase. Consequently, the Revocation Handle $rev_{u_\circ,p_\circ} = \tilde{g}^{x_{u_\circ,p_\circ}}$ has also been published. $\mathcal{B}$ can easily recover $u_\circ$ and $p_\circ$, since he can check which $\tilde{g}^{x_{u_\circ,p_\circ}+h'(p_\circ)f(p_\circ)}$ in the list V satisfies the following equality $\hat{e}(g^\beta,\tilde{g}^\gamma)=\hat{e}(g^\alpha,\tilde{g}^{x_{u_\circ,p_\circ}+h'(p_\circ)f(p_\circ)})$; $\mathcal{B}$ can then look up the respective $x_{u_\circ,p_\circ}$.

If $p_\circ = p_*$, then $\mathcal{A}$ has lost the game, since a successful answer of the attacker cannot be revoked by any of the issued Revocation Handles, whereas this Credential can be revoked with $rev_{u_\circ,p_\circ}=\tilde{g}^{x_{u_\circ,p_\circ}}$. Then it must be that $p_\circ \neq p_*$; in this case $x_* = x_{u_\circ,p_\circ}+f(p_\circ)h'(p_\circ)-f(p_*)h'(p_*) = x_{u_\circ,p_\circ}+a(f(p_\circ)h(p_\circ)-f(p_*)h(p_*))$.

It then follows that

$$\begin{aligned}\hat{e}\left(g,\tilde{g}\right)^k &= \hat{e}\left(g,\tilde{g}\right)^{rb(x_{u_\circ,p_\circ}+a(f(p_\circ)h(p_\circ)-f(p_*)h(p_*)))} = \\ &= \hat{e}\left(g,\tilde{g}\right)^{rbx_{u_\circ,p_\circ}}\cdot\hat{e}\left(g,\tilde{g}\right)^{rab(f(p_\circ)h(p_\circ)-f(p_*)h(p_*))}\end{aligned}$$

The challenger is therefore able to compute

$$\hat{e}\left(g,\tilde{g}\right)^{rab} = \left(\frac{\hat{e}\left(g,\tilde{g}\right)^k}{\hat{e}\left(g^\alpha,\tilde{g}^b\right)^{x_{u_\circ,p_\circ}}}\right)^{(f(p_\circ)h(p_\circ)-f(p_*)h(p_*))^{-1}}$$

and can then solve the given DDH instance by checking whether $\hat{e}\left(g,\tilde{g}\right)^{rab}=\hat{e}\left(g^\alpha,\tilde{g}^\sigma\right)$. □

Now we address the security of impersonation resistance as a whole, by presenting a Lemma that unifies Impersonate1 and Impersonate2.

Lemma 12. *If an adversary $\mathcal{A}$ has a non-null advantage*

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

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$ to gain either an advantage $\frac{\mathsf{AdvImpersonate}_{\mathcal{A}}}{2}$ on the Decisional Diffie-Hellman problem (DDH) or an advantage $\frac{\mathsf{AdvImpersonate}_{\mathcal{A}}}{2}$ on the SM Problem.

Proof. At the beginning of the game, the challenger flips a fair coin $b \in \{1,2\}$ and starts to play Impersonateb with the adversary. The adversary does not know which game the challenger plays.

If the adversary wins, the challenger succeeds in obtaining a reduction for at least one of the two games: which game depends on the behaviour of the attacker; however the challenger cannot predict this as it must commit to one game in advance, before knowing the behaviour of the adversary.

Let us assume that the attacker wins with some probability p; then, the advantage p will be decreased by a factor of $\frac{1}{2}$ to turn it into an actual advantage against the game that has actually been chosen.

Lemmas 10 and 11 tell us that this advantage can be turned into an advantage over the DDH or SM problems. □

5.4 Secret Handshake with Dynamic Matching and Revocation Support

In this Section we show how the scheme introduced in Section 5.3 can be changed to support Dynamic Controlled Matching, reusable Credentials and revocation. We remind the reader that the difference between Dynamic Matching and Dynamic Controlled Matching is that in the former, users can compute Matching References at their will, whereas in the latter, Matching References are only computable by the certification entity.

The scheme that we introduce in this Section effectively adds revocation support to the scheme presented by Ateniese and colleagues in [AKB07].

This scheme is essentially equal to the scheme introduced in Section 5.3, with one substantial difference: $f(p) = 1$ for each property $p \in \mathcal{P}$. We recall that $f(p)$ was previously used by the certification authority to prevent users from forging Matching References. Now this is not needed anymore since users must be able to create Matching References at their will. With this scheme, two users with valid Credentials can interact expressing *wishes* on the property certified by the other user's Credential; wishes are represented by self-generated Matching References. Both users at the end of the protocol share a common key pair if they both own Credentials for the property expected by the other user.

Let us review the algorithms, highlighting the changes with respect to the scheme in Section 5.3:

- Setup f is implemented by setting $f(p) = 1$ for all $p \in \mathcal{P}$; the rest does not change;
- Certify no changes; in particular $C_{u,p,1}$ is still formed as $g^{zw(x_{u,p}+f(p)h(p))}$; however in this case, $g^{zw(x_{u,p}+f(p)h(p))} = g^{zw(x_{u,p}+h(p))}$;
- Grant any user wishing to compute a Matching Reference for property $p \in \mathcal{P}$ can compute $match_p = \tilde{g}^{h(p)f(p)} = \tilde{g}^{h(p)} = H(p) = \tilde{g}_0 \prod_{i \in V(p)} \tilde{g}_i$ thanks to the knowledge of public parameters $\tilde{g}_0 = \tilde{g}^{y_0}, \tilde{g}_1 = \tilde{g}^{y_1}, \ldots, \tilde{g}_n = \tilde{g}^{y_n}$;
- 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* $rev_{u,p} = \tilde{g}^{x_{u,p}+h(p)}$ to a publicly available revocation list L_{rev};

- Handshake is a probabilistic polynomial-time two-party algorithm executed by two users; the algorithm is composed of three sub-algorithms:
  - Handshake.Init no change;
  - Handshake.RandomizeCredentials no change;
  - Handshake.CheckRevoked the user parses 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

    $$\hat{e}\left((C_{u,p,1})^{rs}, (C_{u,p,2})^{s^{-1}}\right) = \hat{e}\left(g^r, rev\right) \tag{5.33}$$

    is verified with any of the Revocation Handles rev in the list L_{rev}. If the check is successful, B discards the current handshake instance;
  - Handshake.Match no change;

As we can see, Revocation Handles are formed differently, as $rev_{u,p} = \tilde{g}^{x_{u,p}+h(p)}$. Consequently, also the Handshake.CheckRevoked algorithm changes. In particular, the Matching Reference is no longer required to perform the check; in this case, since every user has the right to match any property, a revoked Credential loses its Unlinkability to every other user.

5.4.1 Security Analysis

In this Section we will analyse the security of the Secret Handshake scheme introduced in the previous one. Due to the extreme similarity with the scheme presented in Section 5.3, we will not repeat the previous analysis, but we will only highlight the more significant differences.

5.4.1.1 Detection Resistance and Unlinkability of Properties

Detection resistance and Unlinkability of properties are two strictly related requirements. Before their investigation, let us first of all make some general considerations about the nature of the protocol. At first we shall state two predicates:

$\mathfrak{P}1$:= "*User A has a Credential for user B's Matching Reference's property*"
$\mathfrak{P}2$:= "*User B has a Credential for user A's Matching Reference's property*"

The conditions under which users A and B know of a successful Secret Handshake are $\mathfrak{P}1 \bigcup \mathfrak{P}2$. If A has a legitimate Credential for the wish of B (the Matching Reference generated by B), and guesses correctly the property object of B's Credential, then A has legitimately detected B's property. If A is successful twice, we might say that he has been able to trace the property of user B over two different protocol instances.

Both situations are acceptable and are indeed a feature of Dynamic Matching. However it should be clear that this does not in any way imply that a user, by simply performing Secret Handshake repeatedly, trying all possible Matching References, will eventually discover another user's Credential. Indeed the discovery of the remote user's property (proposition $\mathfrak{P}2$) is subject to the possession of an appropriate Credential for the remote user's self-generated Matching Reference (proposition $\mathfrak{P}1$).

For these reasons, detection resistance as introduced in Section 5.3.3.3 is no longer a requirement.

As for Unlinkability of properties, the requirements still stands, but the approach used to prove it in Section 5.3.3.1 needs to be changed. Indeed, in the TraceProperty game, the adversary receives the keys that the challenger computes after the execution of the Handshake.RandomizeCredentials algorithm. The adversary is then challenged to tell whether or not upon both executions of Secret Handshake, the challenger has used Credentials for p_*.

The game has to be modified since in the protocol introduced in this Section, the adversary disposes – by definition – of Matching References for each property. In presence of a handshake message $SH = \left\langle g^r, (C_{u,p,1})^{rs},\ (C_{u,p,2})^{s^{-1}}, (C_{u,p,3})^{s^{-1}} \right\rangle$ and of the correct key $\hat{e}\left(g, \tilde{g}\right)^k$, the adversary can mount a dictionary attack and check for which $p \in \mathcal{P}$, the following equation

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

is satisfied (m is the nonce generated by the attacker for that session of the protocol); this way, having the correct key, the adversary can easily find out the property contained in each Secret Handshake instance and compare them, thus systematically winning the game.

Therefore the adversary cannot be provided with the key. We state once more that this is not limiting in any way the capabilities of the attacker. Indeed, every user has always a chance to compute a key and check whether it is the same as the one computed by the remote user: this happens however under the provision that the user has a correct Credential for the remote user's self-generated Matching Reference. If this side of the handshake succeeds, then the user has a chance of inferring something about his success in the detection (and consequently, if carried out over multiple instances, in the linking) of the property object of the remote user's Credential. In case the mutual proof of knowledge of the keys is not successful however, the user does not know whether this is due to the key computed by Handshake.RandomizeCredentials or by Handshake.Match. Therefore he cannot infer anything about the other user's property.

To show this, we create a game called TraceProperty′, differing from TraceProperty because the adversary is not allowed to receive the correct key at the end. In addition, the adversary also does not receive the Revocation Handle for the Credential, since otherwise he could infer something by running Handshake.CheckRevoked. Finally, the adversary cannot submit any choice for the property it will be attempting to trace. To create the reduction we then use the same strategy adopted by Ateniese and colleagues in [AKB07] for the proof of Lemma 1.

Lemma 13. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{TraceProperty'}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{TraceProperty'}] - \frac{1}{2}$$

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$'s advantage to solve a given instance of the Decisional Diffie-Hellman Problem (DDH) in $\mathbb{G}_1$.

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game TraceProperty to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge, picks and publishes the public parameters according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge At the end of this phase $\mathcal{B}$ engages in two instances of $\mathsf{Handshake}$ with $\mathcal{A}$; in particular, $\mathcal{B}$ picks $x_1, x_2, s_1, s_2, r \overset{R}{\leftarrow} \mathbb{Z}_q^*$; $\mathcal{B}$ generates two handshake tuples as follows:

$$\left\langle g^r, g^{rs_1w(x_1+b)}, \tilde{g}^{(s_1w)^{-1}}, \tilde{g}^{s_1^{-1}} \right\rangle$$
$$\left\langle g^a, g^{as_2wx_2}g^{\sigma s_2 w}, \tilde{g}^{(s_2w)^{-1}}, \tilde{g}^{s_2^{-1}} \right\rangle$$

Analysis of $\mathcal{A}$'s answer It is straightforward to verify that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can give the same answer to solve the DDH problem. □

In addition, we can also show that – upon a failed Secret Handshake execution, which implies in absence of the correct key, handshake tuples are *property-oblivious.*

Definition 14 (Property-Obliviousness)**.** *A Secret Handshake is property-oblivious if – at the end of an unsuccessful Secret Handshake– a user does have any information on the property of the other user.*

We now claim that our scheme guarantees property-obliviousness, under the provision that the adversary does not get any information about the correct key.

Lemma 14. *The scheme presented in Section 5.4 guarantees property-obliviousness under the assumption that the mutual proof of knowledge of the keys does not leak any information about them.*

Lemma 14 tells us that – if the handshake fails – all Credentials are equally likely to have been used and therefore indistinguishable between one another.

For the proof of this Lemma, we draw inspiration from the work of Nasserian and Tsudik [NT06] which in turn builds on the paper by Li, Du and Boneh [LDB05]. Before proving the Lemma, we introduce the following terminology. Two distribution families $D_1(t)$ and $D_2(t)$ are statistically indistinguishable if

$$\sum_y \left| Pr_{x \in D_1(k)}[x = y] - Pr_{x \in D_2(k)}[x = y] \right| \textit{ is negligible in } k$$

where k is the security parameter of our scheme.

Proof. In order to demonstrate this, let us at first list the information that an adversary, trying to infer something about the property of the remote party, is entitled to get[1], besides the public parameters of the system:

$$\left\langle g^r, g^{rsw(x+h(p))}, \tilde{g}^{(sw)^{-1}}, \tilde{g}^{s^{-1}}, \tilde{g}^{x+h(p)} \right\rangle$$

Let us now, fixed $g \in \mathbb{G}_1$, $\tilde{g} \in \mathbb{G}_2$, r and $s \in \mathbb{Z}_q^*$ and $p \in \mathcal{P}$ build the two distribution families

$$D_1(k) = \left\{ g^{rs(x+h(p))}, \tilde{g}^{x+h(p)} | x \in \{1 \dots 2^k q\} \right\}$$

and

$$D_2(k) = \left\{ g^{rs(x')}, \tilde{g}^{x'} | x' \in \{1 \dots 2^k q\} \right\}$$

Since these values are chosen at random and the two distribution families range over the same values, the two distributions are statistically indistinguishable.

In more detail, since q is the order of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, $D_1(k)$ and $D_2(k)$ (for a fixed k) each have q points. The probability difference on any point is at most $\frac{1}{2^k q}$, therefore, the total difference is at most $\frac{q}{2^k q} = \frac{1}{2^k}$. Since this quantity is negligible in k, the two distribution sets are statistically indistinguishable. □

5.4.1.2 Unlinkability of Users

In Section 5.3.3.2 we have presented two different scenarios where we want Unlinkability of users to hold: one where a user uses Credentials that have not yet been revoked and a second situation in which the user uses Credentials that have already been revoked; however, due to the changes brought forward in this protocol, we have seen that once Matching Reference are public, revoked Credential become linkable. Therefore the second scenario is no longer relevant.

As for the first one, in Section 5.3.3.2 we have presented a game called TraceUser1, showing that if an adversary that can break Unlinkability of users for non-revoked Credentials exists, we can use it to build an algorithm that breaks the DDH problem in $\mathbb{G}_1$. To prove the same requirement for the protocol object of this analysis the same game can be used.

Lemma 15. *If an adversary $\mathcal{A}$ has a non-null advantage*

[1]Notice that Revocation Handles are included; by including them we are also implying that the knowledge of Revocation Handles does not in any way help the detection or tracing of properties.

$$\mathsf{AdvTraceUser1}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{TraceUser1}] - \frac{1}{2}$$

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

A proof of this Lemma is straightforward adaptation of the proof of Lemma 7. We therefore omit it here.

5.4.1.3 Impersonation Resistance

In Section 5.3.3.4 we have introduced a game called Impersonate, where the adversary is challenged to pick a property $p_* \in \mathcal{P}$, engage in a Secret Handshake instance with the challenger, generate a handshake tuple and produce the key that theHandshake.RandomizeCredentials algorithm would generate on input a Credential for p_*. In the query phase, the adversary is entitled to ask for arbitrary Credentials. However, at the end of the query phase, the challenger revokes all Credentials handed out to the adversary, so the adversary has to be able to produce a handshake tuple for which Handshake.CheckRevoked does not return true.

Furthermore, we have presented two sub-games challenging different capabilities of the adversary: in particular Impersonate1 challenges the ability of the adversary to forge Credentials while Impersonate2 challenges the ability of the adversary to reuse revoked Credentials yet bypassing revocation.

The second game does not hold anymore for the protocol presented in this Section, since Matching References are no longer required to perform Handshake.CheckRevoked: an adversary reusing a revoked Credential would systematically lose the Impersonate game.

The first game instead holds unmodified.

Lemma 16. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvImpersonate1}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{Impersonate1}]$$

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$'s advantage to solve a given instance of the SM Problem.

A proof of this Lemma is straightforward adaptation of the proof of Lemma 10. We therefore omit it here.

5.5 Conclusion

In this Chapter we have addressed the problem of revocation of Credentials in Secret Handshake. We have presented two schemes: a first one with the same functional requirements as the one introduced in Chapter 4, namely Secret Handshake with Dynamic Controlled Matching with support for reusable Credentials; in addition, our protocol brings revocation support based on revocation lists where the certification entity can publish Revocation Handles. However the introduction of revocation does not hinder other security requirements, namely Unlinkability of users. Indeed, users are unlinkable even after revocation; a revoked user loses its Unlinkability after the publication of the corresponding Revocation Handle, *only* to users that own Credentials for the particular property object of the revoked Credential.

Then we have presented a second scheme, similar to the first one but supporting Dynamic Matching instead of Dynamic Controlled Matching; this scheme effectively enhances the work of Ateniese and colleagues [AKB07] with revocation support.

The two schemes represent a contribution to the state of the art, as they are the first ones supporting revocation, reusable Credentials and Secret Handshakes more flexible than classic Secret Handshake.

In the study of the security of the protocols, we have discovered an interesting new complexity assumption, the SM Problem, which we have proved secure in the generic group model. We have proved the security of the schemes presented in the Chapter by reduction to the new assumption as well as to other well-known hard problems, without relying on the random oracle model. As we shall see in the next Chapter, the SM Problem is of independent interest, and can be used in the security analysis of other schemes.

Chapter 6

Towards Decentralized Secret Handshakes

6.1 Introduction

In Chapters 4 and 5, we have presented a number of Secret Handshake schemes that improve the state of the art by introducing Dynamic Controlled Matching and an approach to support revocation in most Secret Handshake schemes.

One common trait of these schemes, and of all the schemes in the state of the art reviewed in Chapter 3, is that they all rely on a centralized entity that we called certification authority, that is in charge of generating public parameters and of handing over either just Credentials or both Credentials and Matching References.

In this Chapter we investigate further scenarios by relaxing this property. We present two different solutions: the first one is a modification of the scheme presented in Chapter 5, whereby 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 second scheme is instead a completely decentralized one, where a set of users can organize themselves autonomously in order to issue Credentials and be able to mutually authenticate. The first scheme maintains the characteristics of Dynamic Controlled Matching, whereas the second one is a classic Secret Handshake scheme, and hence non-separable.

We will investigate thoroughly the security of the first scheme; as for the second one instead, we will only informally motivate its security, since its purpose is to serve as building block for the use-case of Chapter 7. Nonetheless, the scheme is built on well-established cryptographic primitives, whose security has extensively been studied by the research community.

6.2 Secret Handshakes with multiple CA support

In this Section we present a variation of the scheme that we introduced in Section 5.3; in the variation, there is no single central CA, but a coalition of CAs. Multiple CAs may be a requirement when the properties at stake are – for example – membership to different secret agencies that do not want to delegate the execution of Certify, Grant and Revoke for security reasons, but still want to be able to conduct Secret Handshakes involving properties under the control of the various CAs.

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.

6.2.1 Description of the Scheme

Let us review the algorithms, highlighting the changes with respect to the scheme in Section 5.3:

- 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 $\{y_i\}_{i=0}^{n} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ 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 $g_0 \prod_{i \in V(p)} g_i = g^{h(p)}$.

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

The system's parameters are $\{q, \mathbb{G}_1, \mathbb{G}_2, g, \tilde{g}, W, g_0, \dots, g_n\}$; the values $y_0, \dots, y_n$ 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 $f_{CA}(p)$;

- 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 $x_{u,p} \overset{R}{\leftarrow} \mathbb{Z}_q^*$, 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 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 $\hat{e}(g, match_p) = \hat{e}(g^{f_{CA}(p)}, \tilde{g}^{t_{CA} h(p)})$;

The algorithms that have not been listed stay the same.

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 can 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.

6.2.2 Security Analysis

In this Section we address the security analysis of the scheme. Throughout our analysis, we shall consider two separate types of adversary: a first type is 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 $f_{CA}(p)$.

A second type of adversary is 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 $f_{CA}(p)$.

6.2.2.1 Security against adversary type I

In Section 5.3.3 we have studied the security of the scheme upon which the present scheme is built, against this type of adversary. As a consequence, we will not present the full analysis again but only explain how it can be adapted to fit the scheme introduced in this Section.

Here below, we list each of the games used in the proofs of security of the scheme in Section 5.3 and highlight how they can be adapted to the scheme presented in this Section:

- TraceProperty: this game addresses an attacker willing to trace the same properties over multiple handshake instances; the reduction provided in the game leverages on the fact that the CA can choose arbitrary values for $f(p)$; tracking the same value over multiple handshake instances is equivalent to solving the DDH problem. Given that also in the present scheme each CAs can choose arbitrary values for $f_{CA}(p)$, the same game and the same reduction can be used to prove the fulfillment of the Unlinkability of properties requirement;

- TraceUser1: this game challenges the adversary's capability to trace a non-revoked user, having the appropriate Matching Reference for the user's Credential; the reduction leverages on the fact that the only value that can be traced upon multiple executions of the protocol to the same user is $x_{u,p}$; this value can be chosen arbitrarily by the CA and tracing it upon multiple executions of Secret Handshake is equivalent to solving the DDH problem. The same game and the same reduction can be used to prove the fulfillment of this requirement;

- TraceUser2: this game challenges the adversary's ability to trace a revoked user without the appropriate Matching Reference for the user's (revoked) Credential; indeed the check that deanonymizes a revoked user can only be executed if a user holds a Matching Reference for the property object of the revoked Credential; the reduction once more leverages on the fact that the CA can choose arbitrary values for $f(p)$, feature that has not changed in this scheme; if an attacker able to win this game exists, we can use its advantage to break the DDH problem. Given that also in the present scheme each CAs can choose arbitrary values for $f_{CA}(p)$, the same game and the same reduction can be used to prove the fulfillment of the Unlinkability of properties requirement;

- Detect: this game addresses 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; the detection is successful if the adversary is able to distinguish the correct key from a random string; the reduction once more leverages on the fact that the CA is free to pick arbitrary values for $f(p)$; if the attacker is successful in distinguishing the key, the challenger can solve the DDH problem. Given that also in the present scheme each CAs can choose arbitrary values for $f_{CA}(p)$, the same game and the same reduction can be used to prove the fulfillment of the detector resistance requirement;

- Impersonate1: this game addresses the resistance of the scheme to an attacker that tries to conduct a successful Secret Handshake for a given property, having access to an arbitrary number of Credentials that are however revoked before the challenge phase: in addition, it is assumed that the attacker will forge a brand new Credential in his impersonation attempt. In Section 5.3.3 we have introduced a new hard problem, the SM problem, and given evidence of its intractability; we have then been able to demonstrate that the challenger can set up an environment where the advantage that an attacker has against the Impersonate1 game can be used to solve arbitrary instances of the SM problem. The changes that have been brought forward in this scheme change the nature of the reduction: indeed the values $\tilde{g}_0, \ldots, \tilde{g}_n$ have now become $g_0 = g^{y_0}, \ldots, g_n = g^{y_n}$, so the value $\tilde{g}^y$ cannot be assigned to $\tilde{g}_0$; however the challenger is still free to simulate the values T_{CA} as $(\tilde{g}^y)^{t_{CA}}$; the challenger can then pick random values for $f_{CA}(p)$ and $y_0, \ldots, y_n$; then the challenger can answer all the queries of the adversary, in particular, the challenger can answer queries as if coming from different CAs. At the end of the challenge phase the challenger revokes all Credentials issued; the adversary is then challenged to pick a CA and a property and produce a Credential for that property; that Credential will be formed as $\left\langle g^r, g^{rswx}, \tilde{g}^{(sw)^{-1}}, \tilde{g}^{s^{-1}} \right\rangle$; the constraint is that the challenger has not issued the adversary the tuple $\left\langle g^{zwx}, \tilde{g}^{(zw)^{-1}}, \tilde{g}^{z^{-1}} \right\rangle$ as a result of a Certify query; the challenger can therefore play the same game with the adversary to prove the scheme's resistance to the same type of attack;

- Impersonate2: this game focuses on an adversary trying to conduct a successful Secret Handshake for a given property; the adversary can receive an arbitrary number of Credentials that are revoked before the challenge phase: the additional assumption is that during the challenge phase, the adversary will reuse one of the Credentials received during the query phase, and is nonetheless able to circumvent the revocation checks. The reduction leverages on the fact that if the adversary reuses one of the Credentials received during the challenge phase and is yet able to circumvent revocation, it must be that it is using it for a different property; if so, then the key the adversary generates must contain the term $t_{CA_i} f(p_\circ) h(p_\circ) -$

$t_{CA_j} f(p_*)h(p_*)$, where $p_\circ$ is the original property for the Credential and p_* is the property the adversary has picked for the challenge; CA_i and CA_j are any two (not necessarily distinct) CAs. Then, the challenger can set all the elements T_{CA} as functions of $\tilde{g}^a$ (notice that in Section 5.3.3 this was done by setting $\tilde{g}_0, \ldots, \tilde{g}_i$ as function of $\tilde{g}^a$; this is however no longer possible given that these elements have been replaced by $g_0, \ldots, g_n$) and send the nonce $\tilde{g}^b$ during the challenge phase and use the adversary to solve the DDH in $\mathbb{G}_2$. The same game can be played against an adversary of this scheme and the same reduction can be used;

- Impersonate: this game is required to unite the two different games Impersonate2 and Impersonate2 and prove that the scheme is resistant to impersonation attacks regardless of the strategy used by the adversary; in order to use this approach, it needs to be shown that the criteria as the one used to create the two separate games creates a clear cut between the two, such that no additional strategy exists. We can use the same criteria used in Section 5.3.3.4: Impersonate1 requires the adversary to produce a handshake tuple containing a Credential that has never been issued before by any CA; Impersonate2 instead requires the adversary to reuse a Credential that has at some point been issued by one CA. In particular on the last point, the Credential must have been issued for some property $p_\circ$ by some CA $CA_\circ$ but can be reused either for a different property p_* for any CA (even $CA_\circ$) or for a different CA CA_* for any property (even $p_\circ$). The used criterion creates two games whose union is the original game; as a conclusion, the same game can be adapted to the same end to prove that the scheme presented in this Section fulfills the requirement of resistance to impersonation attacks;

6.2.2.2 Security against adversary type II

It remains to be shown that colluding CAs cannot forge Credentials and Matching References for a target CA_*. 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, the challenger will run the oracle $\mathcal{O}_{\mathsf{CASetup}}$ for the attacker and provide the output to the adversary: 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 $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. 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.

Lemma 17. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvCADetect}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{CADetect}]$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\langle g, g^a, g^b, g^\sigma \rangle$ of the DDH problem in $\mathbb{G}_1$ and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates, using $\mathcal{A}$'s advantage in the game Detect to help compute the solution to the DDH problem. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{CASetup}}$: $\mathcal{B}$ uses g as the one received from the DDH challenge; the other parameters are generated according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Setup}}$: $f_{\mathcal{B}}(p)$ is implemented as follows: on a query for $f_{\mathcal{B}}(p)$, if p has never been queried before, $\mathcal{B}$ picks a random value $r_p \xleftarrow{R} \mathbb{Z}_q^*$, storing the pair (p, r_p) in a table. Then $\mathcal{B}$ flips a random biased coin $guess(p) \in \{0,1\}$ biased as follows: $guess(p)$ equals 1 with probability δ and is equal to 0 with probability $1-\delta$. $\mathcal{B}$ answers as follows: if $guess(p) = 0$, $\mathcal{B}$ looks up r_p in the table and answers with $f_{\mathcal{B}}(p) = r_p$. Instead, if $guess(p) = 1$, $\mathcal{B}$ answers with $f_{\mathcal{B}}(p) = br_p$; the other parameters are generated according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{B}$ answers according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries $\mathcal{B}$ to receive a Matching Reference for property p_i; assuming that $guess(p_i) = 0$, $\mathcal{B}$ can answer according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{B}$ answers according to the rules of the protocol;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;
Challenge At the end of this phase, $\mathcal{A}$ chooses a property p_*, such that no query to the oracle $\mathcal{O}_{\mathsf{Grant}}$ has been submitted. Let us assume that $guess(p_*) = 1$: as a consequence, $f(p_*) = br_{p_*}$. $\mathcal{B}$ then engages in an instances of Handshake with $\mathcal{A}$; in particular, $\mathcal{B}$ receives a nonce $\tilde{g}^m$; $\mathcal{B}$ then picks $x, s \xleftarrow{R} \mathbb{Z}_q^*$ sets $r' = a$ and $x' = x + \sigma a^{-1} r_{p_*} h(p_*) t_{\mathcal{B}} - br_{p_*} h(p_*) t_{\mathcal{B}}$ and constructs a handshake message as follows:

$$\begin{cases} g^{r'} = g^a \\ g^{r'sw(x'+f(p_*)h(p_*)t_{\mathcal{B}})} = g^{r'sw(x'+br_{p_*}h(p_*)t_{\mathcal{B}})} = g^{aswx} g^{sw\sigma r_{p_*} h(p_*) t_{\mathcal{B}}} \\ \tilde{g}^{(sw)^{-1}} \\ \tilde{g}^{s^{-1}} \end{cases}$$

$t_{\mathcal{B}}$ is the value used by $\mathcal{B}$ to generate $T_{\mathcal{B}}$. $\mathcal{A}$ receives the key generated by the algorithm Handshake.RandomizeCredentials: in order to compute such key, $\mathcal{B}$ assumes that $\sigma = ab$, and therefore computes $\hat{e}\left(g^a, \tilde{g}^m\right)^{x'} = \hat{e}\left(g^a, \tilde{g}^m\right)^{x}$. $\mathcal{A}$ answers the challenge with a bit b; $\mathcal{A}$ wins the game if $b = 0$ if the key is a random bitstring, and $b = 1$ if the key is correct.
Analysis of $\mathcal{A}$'s answer It is straightforward to verify that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can give the same answer to solve the DDH problem. Indeed, if $\mathcal{A}$ wins the game and answers $b = 1$, it means that the key $\mathcal{B}$ generated was correct. Then, the same key must be computable using Equation 5.3 with the Matching Reference for property p_*, $\tilde{g}^{f(p_*)h(p_*)t_{\mathcal{B}}}$.

Then we can write

$$(ax + \sigma r_{p_*} h(p_*) t_{\mathcal{B}} - abr_{p_*} h(p_*) t_{\mathcal{B}})m = max$$

However this equation is satisfied only if $\sigma = ab$, which is the positive answer to the DDH problem. If not, $\mathcal{B}$ can give the negative answer to DDH.

A detailed analysis shows that if $guess(p_*) = 1$ and $guess(p) = 0$ for all other queries to $\mathcal{O}_{\mathsf{Grant}}$ such that $p \neq p_*$, then the execution environment is indistinguishable from the actual game Detect. This happens with probability

$$Pr[guess(p_*) = 1 \text{ and } guess(p) = 0 \text{ for all } p \neq p_*] = \delta \cdot (1 - \delta)^{\mathcal{Q}_p} \tag{6.1}$$

where $\mathcal{Q}_p$ is the number of different properties $\mathcal{A}$ queries to the oracle $\mathcal{O}_{\mathsf{Grant}}$. By setting $\delta \approx \frac{1}{\mathcal{Q}_p}$ we know that the probability in 6.1 is greater than $\frac{1}{e \cdot \mathcal{Q}_p}$. So in conclusion, we can bound the probability of success of the adversary $\mathsf{AdvCADetect}_{\mathcal{A}}$ as $\mathsf{AdvCADetect}_{\mathcal{A}} \leq e \cdot \mathcal{Q}_p \cdot \mathsf{AdvDDH}_{B}$. □

CA Impersonation Resistance To address the analysis of this requirement, we follow the same strategy adopted in Section 5.3.3.4; 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 $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. 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.

To create the first sub-game, CAImpersonate1, we also assume that the attacker will forge a brand new Credential in the challenge phase.

Lemma 18. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvCAImpersonate1}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{CAImpersonate1}]$$

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$'s advantage to solve a given instance of the SM Problem.

Proof. We define the challenger $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\left\langle g, g^w, \tilde{g}, \tilde{g}^{w^{-1}}, \tilde{g}^y,\right.$ $\left.\tilde{g}^m,\ O_{w,y}\right\rangle$ of the SM problem and wishes to use $\mathcal{A}$ to produce the tuple $\langle g^r,\ g^{rsw(x_*+y)},$ $\tilde{g}^{(sw)^{-1}},\ \tilde{g}^{s^{-1}},\ \hat{e}\,(g,\tilde{g})^{rx_*m}\rangle$, such that x_* has not been queried to O. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{CASetup}}$: $\mathcal{B}$ sets public parameters $g, \tilde{g}, g^w, \tilde{g}^{w^{-1}}$ as the ones received from the challenge;

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ sets $T_{\mathcal{B}}$ as $\tilde{g}^y$; the other parameters are generated according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Credentials for an arbitrary number of user u_i and property p_i; $\mathcal{B}$ answers by picking a random Identification Handle $x_{u_i,p_i} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and by giving it to $\mathcal{A}$; $\mathcal{B}$ then queries the oracle $O_{w,y}$ providing $v = \frac{x_{u_i,p_i}}{f_{\mathcal{B}}(p_i)h(p_i)}$ as input, and adding the value v to the set $\mathcal{O}$ of queries to oracle O. The output of the oracle is $\left\langle g^{zw\left(\frac{x_{u_i,p_i}}{f_{\mathcal{B}}(p_i)h(p_i)}+y\right)}, \tilde{g}^{(zw)^{-1}}, \tilde{g}^{z^{-1}}\right\rangle$. $\mathcal{B}$ then assigns $C_{u_i,p_i,1} \leftarrow$ $\left(g^{zw\left(\frac{x_{u_i,p_i}}{f_{\mathcal{B}}(p_i)h(p_i)}+y\right)}\right)^{f_{\mathcal{B}}(p_i)h(p_i)} = g^{zw(x_{u_i,p_i}+yf_{\mathcal{B}}(p_i)h(p_i))}$, $C_{u_i,p_i,2} \leftarrow \tilde{g}^{(zw)^{-1}}$, $C_{u_i,p_i,3}$ $\leftarrow \tilde{g}^{z^{-1}}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{\mathcal{B}}(p_i)}$ and $\tilde{g}^{yh(p_i)}$;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Matching References for an arbitrary number of properties p_i; $\mathcal{B}$ answers with $match_{p_i} = \tilde{g}^{yf_{\mathcal{B}}(p_i)h(p_i)}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{\mathcal{B}}(p_i)}$ and $\tilde{g}^{yh(p_i)}$;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{A}$ queries $\mathcal{B}$ for an arbitrary number of Revocation Handles for user u_i and property p_i; $\mathcal{B}$ answers with $rev_{u_i,p_i} = \tilde{g}^{x_{u_i,p_i}}$;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge $\mathcal{A}$ then declares that this phase of the game is over. $\mathcal{B}$ therefore revokes each of the Credentials $\mathcal{A}$ requested in the previous phase. $\mathcal{A}$ then chooses a property

$p_* \in \mathcal{P}$. $\mathcal{B}$ challenges $\mathcal{A}$ by sending $\tilde{g}^m$ and $\mathcal{A}$ answers the challenge with the tuple $\langle g^\alpha, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta, e^k \rangle$.

Analysis of $\mathcal{A}$'s response If $\mathcal{A}$ wins the game, $\mathcal{B}$ can check that

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, match_{p_*})}\right)^m = \left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}\left(g^\alpha, \tilde{g}^{y f_{\mathcal{B}}(p_*) h(p_*)}\right)}\right)^m = e^k \tag{6.2}$$

and that

$$\hat{e}\left(g, \tilde{g}^\delta\right) = \hat{e}\left(g^w, \tilde{g}^\gamma\right) \tag{6.3}$$

as mandated by the Handshake.Match sub-algorithm detailed in Section 5.3.2.

Let us set $\alpha = r$, $k = r x_* m$ and $\delta = s^{-1}$, for some integers $r, x_*, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, from Equation 6.3 we derive that $\gamma = (sw)^{-1}$ and from Equation 6.2 that $\beta = rsw(x_* + y f_{\mathcal{B}}(p_*) h(p_*))$. Notice that by the definition of the game, the attacker has not received a Credential containing the term $g^{zwX_{u_*,p_*}} = g^{zw(x_* + y f_{\mathcal{B}}(p_*) h(p_*))}$ from the challenger as answer to a query to the $\mathcal{O}_{\mathsf{Certify}}$ oracle.

This implies in turn that the value $v_* = \frac{x_*}{f_{\mathcal{B}}(p_*) h(p_*)}$ has never been queried by the challenger to the oracle $O_{w,y}$ in the execution of a $\mathcal{O}_{\mathsf{Certify}}$ query: as a consequence, v_* does not belong to the set $\mathcal{O}$. Therefore we conclude that, if $\mathcal{A}$ wins the game, $\mathcal{B}$ can provide

$$\left\langle (g^\alpha)^{f_{\mathcal{B}}(p_*) h(p_*)}, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta, \hat{e}\left(g, \tilde{g}\right)^k \right\rangle$$

as an answer to the SM problem. □

We now turn our attention to the CAImpersonate2 game, focusing on a malicious CA attempting to succeed in the impersonation of a user owning a credential for a given property p_*, under the control of the CA managed by the challenger. We also introduce an additional constraint on the impersonation strategy: we assume that the adversary will be reusing an already revoked Credential received from another CA and yet able to circumvent revocation. Given that the revocation check involves the Matching Reference for a property, the only way for the attacker to succeed is to use a Credential received for a property – say $p_\circ$ – in the attempt to impersonate a different property – say p_*.

The malicious CA $\mathcal{A}$ can access $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$. 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.

Lemma 19. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvCAImpersonate2}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{CAImpersonate2}] - \tfrac{1}{2}$$

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

Proof. We define $\mathcal{B}$ as follows. $\mathcal{B}$ is given an instance $\left\langle \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^\sigma \right\rangle$ of the DDH problem and wishes to use $\mathcal{A}$ to decide if $\sigma = ab$. The algorithm $\mathcal{B}$ simulates an environment in which $\mathcal{A}$ operates. In particular, $\mathcal{B}$ implements the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$ as follows:

$\mathcal{O}_{\mathsf{CASetup}}$: $\mathcal{B}$ sets the public parameter $\tilde{g}$ as the ones received from the DDH challenge; the other parameters are generated according to the rules of the protocol;

$\mathcal{O}_{\mathsf{Setup}}$: $\mathcal{B}$ sets $T_{\mathcal{B}} = \tilde{g}^a$; the other parameters are then picked and published following the rules of the protocol;

$\mathcal{O}_{\mathsf{Certify}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Credentials for an arbitrary number of pairs $(u, p) \in \mathcal{U} \times \mathcal{P}$; $\mathcal{B}$ answers with $cred_{u,p} = \langle C_{u,p,1}, C_{u,p,2}, C_{u,p,3} \rangle$ where $C_{u,p,1} = g^{zw}$, $C_{u,p,2} = \tilde{g}^{(zw)^{-1}(x_{u,p} + af_{\mathcal{B}}(p)h(p))}$ and $C_{u,p,3} = \tilde{g}^{z^{-1}(x_{u,p} + af_{\mathcal{B}}(p)h(p))}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{\mathcal{B}}(p)}$ and $\tilde{g}^{ah(p)}$; this representation of Credentials is indistinguishable from the ones mandated by the protocol: indeed notice that we can set $z = z'(x_{u,p} + af_{\mathcal{B}}(p)h(p))$ and we can rewrite $C_{u,p,1} = g^{z'w(x_{u,p} + af_{\mathcal{B}}(p)h(p))}$, $C_{u,p,2} = \tilde{g}^{(z'w)^{-1}}$ and $C_{u,p,3} = \tilde{g}^{z'^{-1}}$ which is exactly the way Credentials are formulated according to the algorithm $\mathsf{Certify}$ described in Section 5.3.2. $\mathcal{B}$ adds to a list V the tuple $(\tilde{g}^{x_{u,p} + ah(p)f_{\mathcal{B}}(p)}, u, p, x_{u,p})$ for each query of $\mathcal{A}$ and keeps it for later use;

$\mathcal{O}_{\mathsf{Grant}}$: $\mathcal{A}$ queries $\mathcal{B}$ for Matching References for an arbitrary number of properties p; $\mathcal{B}$ answers with $match_p = \tilde{g}^{af_{\mathcal{B}}(p)h(p)}$; to allow the user to verify the goodness of the Credential, the CA gives to the user $g^{f_{\mathcal{B}}(p)}$ and $\tilde{g}^{ah(p)}$;

$\mathcal{O}_{\mathsf{Revoke}}$: $\mathcal{A}$ queries $\mathcal{B}$ for an arbitrary number of Revocation Handles for user u_i and property p_i; $\mathcal{B}$ answers with $rev_{u_i,p_i} = \tilde{g}^{x_{u_i,p_i}}$;

Setup and Queries The adversary $\mathcal{A}$ can interact with the oracles $\mathcal{O}_{\mathsf{CASetup}}$, $\mathcal{O}_{\mathsf{Setup}}$, $\mathcal{O}_{\mathsf{Certify}}$, $\mathcal{O}_{\mathsf{Grant}}$, $\mathcal{O}_{\mathsf{Revoke}}$;

Challenge $\mathcal{A}$ then declares that this phase of the game is over. $\mathcal{B}$ therefore revokes each Credential requested by $\mathcal{A}$ in the previous phase. $\mathcal{A}$ then chooses a property $p_* \in \mathcal{P}$. $\mathcal{B}$ challenges $\mathcal{A}$ by sending $\tilde{g}^b$ and $\mathcal{A}$ answers the challenge with the tuple $\langle g^\alpha, g^\beta, \tilde{g}^\gamma, \tilde{g}^\delta, \hat{e}\,(g, \tilde{g})^k\rangle$.
Analysis of $\mathcal{A}$'s response If $\mathcal{A}$ wins the game, $\mathcal{B}$ can check that

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, match_{p_*})}\right)^b = \tag{6.4}$$

$$\left(\frac{\hat{e}(g^\beta, \tilde{g}^\gamma)}{\hat{e}(g^\alpha, \tilde{g}^{af_{\mathcal{B}}(p_*)h(p_*)})}\right)^b = \hat{e}\,(g, \tilde{g})^k \tag{6.5}$$

and that

$$\hat{e}\left(g, \tilde{g}^\delta\right) = \hat{e}\,(g^w, \tilde{g}^\gamma) \tag{6.6}$$

as mandated by the Handshake.Match sub-algorithm detailed in Section 5.3.2.

Let us set $\alpha = r$, $k = rx_*b$ and $\delta = s^{-1}$, for some integers $r, x_*, s \in \mathbb{Z}_q^*$ unknown to $\mathcal{B}$. Then, from Equation 6.6 we derive that $\gamma = (ws)^{-1}$; if we set $v_* = x_* + af_{\mathcal{B}}(p_*)h(p_*)$, from Equation 6.5 we can write $\beta = rsv_*$.

We know by definition that the attacker has already received $C_{u_\circ,p_\circ,1} = g^{zwv_*} = g^{zw(x_{u_\circ,p_\circ}+af_{\mathcal{B}}(p_\circ)h(p_\circ))}$ during the previous query phase. Consequently, the Revocation Handle $rev_{u_\circ,p_\circ} = \tilde{g}^{x_{u_\circ,p_\circ}}$ has also been published. $\mathcal{B}$ can easily recover $u_\circ$ and $p_\circ$, since he can check which element $\tilde{g}^{x_{u_\circ,p_\circ}+ah(p_\circ)f_{\mathcal{B}}(p_\circ)}$ within the list V satisfies the following equality $\hat{e}(g^\beta, \tilde{g}^\gamma) = \hat{e}(g^\alpha, \tilde{g}^{x_{u_\circ,p_\circ}+ah(p_\circ)f_{\mathcal{B}}(p_\circ)})$, and then look up the respective $x_{u_\circ,p_\circ}$.

If $p_\circ = p_*$, then $\mathcal{A}$ has lost the game, since a successful answer of the attacker cannot be revoked by any of the issued Revocation Handles, whereas this Credential can be revoked with $rev_{u_\circ,p_\circ}$. Then it must be that $p_\circ \neq p_*$; in this case $x_* = x_{u_\circ,p_\circ} + af_{\mathcal{B}}(p_\circ)h(p_\circ) - af_{\mathcal{B}}(p_*)h(p_*)$.

It then follows that

$$\hat{e}\,(g, \tilde{g})^k = \hat{e}\,(g, \tilde{g})^{rb(x_{u_\circ,p_\circ}+af_{\mathcal{B}}(p_\circ)h(p_\circ)-af_{\mathcal{B}}(p_*)h(p_*))}$$

The challenger is therefore able to compute

$$\hat{e}\,(g, \tilde{g})^{rab} = \left(\frac{\hat{e}\,(g, \tilde{g})^k}{\hat{e}\,(g^r, \tilde{g}^b)^{x_{u_\circ,p_\circ}}}\right)^{(f_{\mathcal{B}}(p_\circ)h(p_\circ)-f_{\mathcal{B}}(p_*)h(p_*))^{-1}}$$

and can then solve the given DDH instance by checking whether $\hat{e}\,(g, \tilde{g})^{rab} = \hat{e}\,(g^\alpha, \tilde{g}^\sigma)$.

□

To conclude the analysis of impersonation resistance by adversary type II, we introduce a final lemma that merges the two games $\mathsf{CAImpersonate1}$ and $\mathsf{CAImpersonate2}$ engaging the attacker in the $\mathsf{CAImpersonate}$ game where the challenger gets an advantage in solving a hard problem independently of the strategy of the adversary.

Lemma 20. *If an adversary $\mathcal{A}$ has a non-null advantage*

$$\mathsf{AdvCAImpersonate}_{\mathcal{A}} := Pr[\mathcal{A} \textit{ wins the game } \mathsf{CAImpersonate}]$$

then a probabilistic, polynomial time algorithm $\mathcal{B}$ can create an environment where it uses $\mathcal{A}$ to gain either an advantage $\frac{\mathsf{AdvCAImpersonate}_{\mathcal{A}}}{2}$ on the Decisional Diffie-Hellman problem (DDH) or an advantage $\frac{\mathsf{AdvCAImpersonate}_{\mathcal{A}}}{2}$ on the SM Problem.

A proof of this Lemma is a straightforward adaptation of the proof of Lemma 12, which we therefore omit here.

6.3 Secret Handshakes with Ad-Hoc Certification

In this Section we present a Secret Handshake scheme where Credentials and Matching References are not generated centrally by one (or a coalition of) CAs, but where instead the task of generating Credentials and Matching References is shared among users in a completely decentralized way. The resulting scheme is non-separable, and does not support reusable Credentials. As a result of non-separability, the scheme only supports matching within the same group. The main idea behind the scheme is that, if a new user is endorsed by t existing group members out of the total n group members (t and n being parameters of the system), he is then able to obtain a Credential certifying that he is a group member too.

The scheme is based on secret sharing and threshold cryptography, concepts first introduced by Shamir [Sha79] and independently by Blakley [Bla79]. Threshold cryptography is based on the main idea of sharing "*something*" (such as for instance a secret value, or something more complex such as the power of generating a signature) among n users, in a way that only t of them, cooperating, can obtain it, but no subset of $t-1$ can.

Using this scheme, as we shall see in the rest of this Section, an initial set of users (*group managers*) can spontaneously form a group; a consensus of at least t managers has the capability to appoint new managers and to appoint simple group members. Group members can simply authenticate as such, but cannot issue new Credentials.

In Section 6.3.5 we will sketch the security analysis of the scheme, without going into the details. The reason for this choice is that this scheme only intends to be a preliminary effort towards Secret Handshakes with ad-hoc certification and to serve as a building block to support the use-case scenario presented in Chapter 7. Nonetheless, the scheme is built on solid primitives such as threshold cryptography and OSBEs, whose security has already been extensively studied in the literature.

6.3.1 Preliminaries

Secret sharing allows a set of n parties to possess shares of a secret value, such that any t shares can be used to reconstruct the secret, yet any $t-1$ shares provide no information about it. Secret sharing was first proposed by Shamir [Sha79] and independently by Blakley [Bla79]. This initial idea has been extended in a number of works [DJ97; Fel87; HJKY95; Ped91b; Ped91a]. For the objectives of our scheme, we will use two different algorithms: Share [Ped91b; Ped91a], used by n users to share a random secret without a dealer, so that t are in principles able to reconstruct it; and Redistribute [DJ97], used by t shareholders to compute n' new, unrelated shares of the same secret, so that t' new shareholders can reconstruct the secret. In both algorithms, the secret is actually never reconstructed, and – since there is no dealer – none of the shareholders knows it. We call $\Gamma_{\mathcal{P}}^{(n,t)}$ the access structure wherein a secret is shared among a population of users in the set $\mathcal{P}$, with $|\mathcal{P}| = n$ so that any subset $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$, with $|\mathcal{B}| = t$ can reconstruct that secret.

Share : this algorithm is executed by each P_i belonging to an authorised set $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$ with cardinality t. Each P_i picks a random $r_i \overset{R}{\leftarrow} \mathbb{Z}_q$, forms a random polynomial $f_i(u) = r_i + a_{i,1}u + \ldots + a_{i,t-1}u^{t-1}$ and sends $f_i(j) \mod q$ to each $P_j \in \mathcal{P}/P_i$; the (unknown) shared secret is $R = \sum_{j \in \mathcal{P}} r_j$. Every $P_i \in \mathcal{P}$ computes its share of the secret $R_i = \sum_{j \in \mathcal{B}} f_j(i)$; additionally, each P_i broadcasts $g^{r_i} \mod p$; this way, everybody can compute $g^R \mod p$;

Redistribute : this algorithm is executed by each P_i belonging to an authorized set $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$ with cardinality t. The objective is to generate new shares for the new access structure $\Gamma_{\mathcal{P}'}^{(n',t')}$. Each P_i computes a random polynomial formed as

$f_i(u) = R_i + a_{i,1}u + \ldots + a_{i,t'-1}u^{t'-1}$, where R_i is the local share of the secret possessed by P_i; each P_i sends $f_i(j) \mod q$ to each $P_j \in \mathcal{P}'/P_i$; then, each P_i can locally generate its new share R'_i by Lagrange interpolation;

Using these two algorithms, we can generate threshold signatures. At first we describe a variant [PK96] of the famous DSS signature scheme [BP]. Let the secret signing key be $x \in \mathbb{Z}_q$ and the public key be $g^x \mod p$. The scheme has two algorithms:

Sign : given the message m and a random number $e \overset{R}{\leftarrow} \mathbb{Z}_q$, compute the signature (w, v) such that $w = (g^e \mod p) \mod q$ and $v = wx + h(m)e \mod q$;

Verify : (w, v) is a valid signature on the message m if the following equality $w = \left(g^{vh(m)^{-1}} (g^x)^{-wh(m)^{-1}} \mod p\right) \mod q$ holds;

A threshold signature scheme [Boy86; Des87; DF89] leverages on the aforementioned secret sharing techniques in order to share the secret key x among n parties, thus distributing the signing capabilities over n parties, so that any subset of t can jointly compute a signature, whereas no $t-1$ subset can. In [PK96], Park and Kurosawa propose a threshold version of the aforementioned DSS signature variant. The variant assumes that the Share algorithm has been executed to create an access structure $\Gamma_{\mathcal{P}}^{(n,t)}$ and that any principal in $\mathcal{P}$ has a share x_i of the (unknown) private key x. In addition, the public key g^x is publicly known. The Verify algorithm stays the same, whereas the Sign algorithm is modified as follows:

Sign : this algorithm is executed by each P_i belonging to an authorized set $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$ with cardinality t; each P_i has a local share x_i of the secret signing key x. All the P_i engage in the Share algorithm, generating the value $g^e \mod q$ and a local share e_i of the (unknown) random value e; then, each P_i sends the value $v_i = g^e x_i + h(m)e_i \mod q$ to the requester of the signature, along with the value $g^e \mod q$; the first part of the signature is $w = g^e \mod q$; given the set of shares $\{v_i, i \in \mathcal{B}\}$, the second part of the signature v can be computed through Lagrange interpolation;

As we can see, the Share algorithm is executed twice, once prior to signing to generate the public key and shares of the private key, and a second time, to generate the first part of the signature and shares of its discrete log.

Now we introduce the final building block of our scheme, Oblivious Signature-Based Envelopes (OSBEs). OSBEs have been introduced by Li, Du and Boneh in [LDB05]. An OSBE scheme allows two parties to share a key if a predefined party among the two possesses a signature on an agreed-upon message. Nasserian and Tsudik have presented in [NT06] – among others – an OSBE scheme based on the DSS variant mentioned in the previous Section.

At first a message m is chosen; the OSBE round happens between a party P1 who might have a signature (w, v) on m and P2. The OSBE round proceeds as follows:

OSBERound : P1 sends w to P2; P2 generates $r \overset{R}{\leftarrow} \mathbb{Z}_q$, sends g^r to P1 and computes $K_2 = \left((g^x)^w w^{h(m)}\right)^r$; P1 computes $K_1 = (g^r)^v$; $K_1 = K_2$, i.e. P1 and P2 will share a key if P1's signature on m was correct;

6.3.2 Syntactic Definition

In this Section we first give an informal definition of the operations of the scheme, and then define its syntax.

A secret group starts with the invocation of Setup by the initial set of n group managers. At the end of the algorithm, the group managers have picked a group message that will be used later on, they have agreed on a known public key and they have distributed secret shares of an (unknown) private key, representing the group managership token. Notice that the actual private key is unknown; it could be known if at least t group managers colluded and reciprocally revealed their shares, however the threshold t can be set in order to discourage such attempts. In situations where hierarchy of group managers is important, each group managers can be supplied a different number of shares according to their importance.

The algorithm UpdateManagers can be invoked by at least t group managers to redistribute *different* shares of the *same* private key to a *different* set of group managers, fixing a new threshold t'. The reasons for invoking the UpdateManagers are mainly twofold: (i) if the group grows, new group managers need to be appointed and therefore receive shares of the private key; (ii) when a group of (less then t) managers needs to be revoked[1], t other group managers can generate a different set of shares, not related with

[1]Notice that revoking group managership to t group managers or more is not feasible since in principles they can reconstruct the group secret key; this situation can be thwarted by appropriately setting t.

the old set, and distribute the new shares to all managers but the ones whose manager rights need to be revoked, thus effectively preventing the latter from further executing group manager tasks. Regular invocations of UpdateManagers can be scheduled so that new shares are proactively being distributed and the population of group managers can be both purged of elements that are no longer trustworthy and enriched with new trusted ones.

t group managers can also issue group membership Credentials, that users can use as Credentials in the Secret Handshake. To this end, t group managers jointly engage in the execution of GrantMembership, issuing to the new member a signature on the group message that verifies under the group public key, representing the group membership Credentials. Once group members receive their membership tokens (i.e. Credential), they can perform a Secret Handshake, using the Handshake algorithm.

Notice that the policy of a group could be that each new group member automatically becomes a group manager as well; this could be enforced by accompanying each invocation of GrantMembership with an invocation of UpdateManagers, equipping the new user with both group membership and managership Credentials.

Let us now present the scheme through a syntactic definition of its algorithms:

- $\mathsf{Setup}(k) \rightarrow (PK_{\mathcal{G}}, SK_{\mathcal{G}}, \Gamma_{\mathcal{P}}^{(n,t)}, M_{\mathcal{G}})$ is a probabilistic polynomial-time algorithm executed by a group of n managers, taking a security parameter k as input; the managers agree on a message $M_{\mathcal{G}}$, produce a public key $PK_{\mathcal{G}}$ and an access structure $\Gamma_{\mathcal{P}}^{(n,t)}$ to the (unknown) secret key $SK_{\mathcal{G}}$;

- $\mathsf{UpdateManagers}(\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}) \rightarrow (\Gamma_{\mathcal{P}'}^{(n',t')})$ is a probabilistic polynomial-time algorithm executed by authorized set of managers $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$ with cardinality t; the algorithm generates a new access structure $\Gamma_{\mathcal{P}'}^{(n',t')}$;

- $\mathsf{GrantMembership}(\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}) \rightarrow (\sigma)$ is a probabilistic polynomial-time algorithm executed by authorized set of managers $\mathcal{B} \in \Gamma_{\mathcal{P}}^{(n,t)}$ with cardinality t; the algorithm generates a signature σ on the message $M_{\mathcal{G}}$, under the public key $PK_{\mathcal{G}}$;

- Handshake is a probabilistic polynomial-time two-party algorithm executed by two users, u_i and u_j; the algorithm is composed of two sub-algorithms:

- Handshake.RandomizeCredentials(σ) $\rightarrow (SH, \mathsf{state})$ takes as input the signature σ and produces the Secret Handshake message SH; it also creates the internal state state;
- Handshake.Match($\sigma, PK_{\mathcal{G}}, SH, M_{\mathcal{G}}, \mathsf{state}$) $\rightarrow (K_1, K_2)$ takes as input the signature σ, the public key, SH, the Secret Handshake message received from the other user, the local state, the group message $M_{\mathcal{G}}$; the algorithm outputs a pair of keys K_1 and K_2;

The algorithm operates as described in Figure 6.1

u_j			:	Handshake.RandomizeCredentials(σ_j) $\rightarrow (SH_j, \mathsf{state}_j)$
u_i			:	Handshake.RandomizeCredentials(σ_i) $\rightarrow (SH_i, \mathsf{state}_i)$
u_i	$\longrightarrow$	u_j	:	SH_i
u_j	$\longrightarrow$	u_i	:	SH_j
u_i			:	Handshake.Match($\sigma_i, PK_{\mathcal{G}}, SH_j, M_{\mathcal{G}}, \mathsf{state}_i$) $\rightarrow (K_{1,i}, K_{2,i})$
u_j			:	Handshake.Match($\sigma_j, PK_{\mathcal{G}}, SH_i, M_{\mathcal{G}}, \mathsf{state}_j$) $\rightarrow (K_{1,j}, K_{2,j})$

Figure 6.1: The Handshake algorithm executed by two users u_i and u_j.

$K_{1,j} = K_{2,i}$ and $K_{1,i} = K_{2,j}$ are equal, provided that both u_i and u_j have signatures on $M_{\mathcal{G}}$.

6.3.3 The Scheme

In this Section we present a practical scheme based on the cryptographic building blocks introduced in Section 6.3.1. The scheme is composed by the following algorithms:

- Setup this algorithm is executed by a set of n group initiators; each group initiator jointly engages in the Share algorithm, forming the access structure $\Gamma_{\mathcal{P}}^{(n,t)}$, computing the group public key $PK_{\mathcal{G}} = g^x$ and shares of the private key $SK_{\mathcal{G}} = x$ that we represent as $SK_{\mathcal{G}}^i = x_i$; the public parameters are the group public key $PK_{\mathcal{G}} = g^x$ and a message $M_{\mathcal{G}}$; $SK_{\mathcal{G}}^i$ represents the group managership Credential;

- UpdateManagers this algorithm is executed by a set of t group managers; each group manager jointly engages in the Redistribute algorithm, forming a new access structure $\Gamma_{\mathcal{P}'}^{(n',t')}$ computing new shares of the private key $SK_{\mathcal{G}}^i = x_i$ and giving them to the new set of group managers;

- GrantMembership this algorithm is executed by a set of t group managers out of the n total; each group manager checks the requesting users' conformity to the group join policy; after a successful check, each of the t group managers engages in the Sign algorithm, forming a new signature on message $M_{\mathcal{G}}$, that verifies correctly under the group public key $PK_{\mathcal{G}}$; the signature is then issued to the supplicant user; the signature represents the group membership Credential;

- Handshake this algorithm is executed by two group members that want to mutually authenticate as members of the group; the algorithm is composed of two sub-algorithms:

  - Handshake.RandomizeCredentials the user has a signature (w, v) on $M_{\mathcal{G}}$; the user picks $r \overset{R}{\leftarrow} \mathbb{Z}_q$ and sends the pair (w, g^r);
  - Handshake.Match the user receives $(w', g^{r'})$; the user has a signature (w, v) on $M_{\mathcal{G}}$; the user computes $K_1 = \left((g^x)^{w'} w'^{h(M_{\mathcal{G}})}\right)^r$ and $K_2 = (g^{r'})^v$ where g^x is the group public key $PK_{\mathcal{G}}$, $M_{\mathcal{G}}$ is the message of the group and r is the random value picked upon the execution of Handshake.RandomizeCredentials;

Let us assume that two users u_i and u_j hold signature $\sigma_i = (w_i, v_i)$ and $\sigma_j = (w_j, v_j)$ respectively. The two users can engage in a Secret Handshake scheme as shown in Figure 6.2.

$$\begin{array}{ll}
u_i \longrightarrow u_j & w_i, g^{r_i} \\
u_j \longrightarrow u_i & w_j, g^{r_j} \\
u_i & \text{computes } K_{1,i} = \left((g^x)^{w_j}\, w_j^{h(M_{\mathcal{G}})}\right)^{r_i} \\
 & \text{and } K_{2,i} = (g^{r_j})^{v_i} \\
u_j & \text{computes } K_{1,j} = \left((g^x)^{w_i}\, w_i^{h(M_{\mathcal{G}})}\right)^{r_j} \\
 & \text{and } K_{2,j} = (g^{r_i})^{v_j}
\end{array}$$

Figure 6.2: The Handshake algorithm executed by two users u_i and u_j.

$K_{1,j} = K_{2,i}$ and $K_{1,i} = K_{2,j}$ are equal, provided that both u_i and u_j have signatures on $M_{\mathcal{G}}$.

6.3.4 A word on Revocation

The existence of Credentials comes along with the requirement for revocation. The revocation of group managership Credentials should be treated differently from the revocation of simple group membership ones.

The fact that single group manager Credentials fall into the hand of an attacker is not so dangerous, thanks to the use of threshold cryptography: an attacker would indeed need to get hold of at least t group managership Credentials, and since t is a parameter of the system, this threshold can be set to meet the desired degree of security. The most suitable solution to counter the theft of group manager Credentials is therefore a proactive strategy: managership Credentials can be periodically updated by invoking UpdateManagers so as to make sure that the probability that an attacker gets hold of at least t valid managership Credentials is arbitrarily low.

As for group membership Credentials, revocation is required when either a group member got his membership token stolen or when he no longer qualifies for membership. The loss of a group membership Credential is somewhat thornier then the loss of group managership Credentials, since group membership Credentials can be used directly to authenticate to another group member.

There are two possible ways in which revocation of group membership Credentials can be achieved: one, suggested by Boneh and Franklin in [BF03a], consists in periodically updating $M_{\mathcal{G}}$. For example, concatenating the current year to $M_{\mathcal{G}}$ = "`group1`" would make sure that only group members owning a signature on the message "`group1 || 2010`" are able to successfully perform a Secret Handshake. At the expiration date, a group member simply needs to apply for GrantMembership and, if it is still eligible, get a new signature for the next time period, in this case, for the next year.

The second approach for revocation is a reactive one, similar to the one proposed in [BDS+03]. This approach is based on the fact that upon execution of the Secret Handshake, users exchange part of their group membership Credential, in particular, the first component w of their signature (w, v). w is formed by the t group managers that engage in the GrantMembership, who know this value. Consequently, should this Credential be revoked, the t group managers can just forge a new signature on the message "w `is a revoked group membership token`" and broadcast it to all group members.

6.3.5 Sketch of the Security Analysis

The scheme presented in the previous Section leverages on well-established primitives, such as secret sharing, threshold signatures and Oblivious Signature-Based envelopes. The security of these primitives has extensively been studied by the research community. We therefore do not present formal proofs of security for the presented scheme; instead, based on the security of the building blocks, we present arguments supporting the security of the overall scheme.

Let us focus our analysis on three main typologies of adversary: passive adversaries, whose objective is to link users and group membership by observing multiple Secret Handshake executions; active outsiders, who attempt to engage in Secret Handshake without appropriate Credentials; and active insiders, whose aim is to engage in authentication with Credentials bypassing revocation.

Let us start with the requirements of Unlinkability of users and properties (in this case, group membership); Unlinkability of group membership is guaranteed assuming that the proof of knowledge of the computed key pair does not reveal anything about their value in case of failure. The security derives from the fact that the only group-wide value x, present in public key, private key and its shares, is not exchanged during the Secret Handshake execution.

As for Unlinkability of users, our scheme suffers from the same shortcomings of Balfanz's scheme [BDS^{+}03], in that the scheme supports this security property only if users are provided with a number of one-time-use signature pairs (w, v), and none of them are ever reused. In addition, the t group managers that participated in the generation of the signature can deanonymize the bearer of the signature since they have generated its randomizer.

The goal of an active outsider is to authenticate as a group member without owning a membership token. Such an adversary has a number of options to achieve the goal: tricking group managers into issuing them a group membership token is not feasible, since it would require tricking more then t managers, whereas t can be made big enough to counter such attempts; collecting t shares of the private key is not feasible both for the fact that t can be set arbitrarily big and due to the fact that shares can be proactively revoked; finally, group membership tokens are DSS signatures, whose security has been studied extensively in the literature [BP]; this signature scheme guarantees resistance to existential forgery, therefore an active outsider has no possibility of forging membership tokens. Without a valid Credential, users cannot engage in successful authentication; this follows from the property of semantic security against the receiver, which is guaranteed by OSBEs [NT06].

Active insiders have similar goals as active outsiders, with the difference that they dispose of a number of valid Credentials; this fact can be modeled with an oracle that generates a number of signatures on $M_{\mathcal{G}}$, with the attacker then trying to generate a new signature with a different randomizer (thus circumventing revocation). However, the same considerations mentioned for active outsiders apply here: resistance to existential forgery includes this oracle, and therefore its existence does not impact the security of the scheme; the same applies for the semantic security against the receiver of OSBEs.

6.4 Conclusion

The focus of this Chapter has been the study of Secret Handshake schemes that do not rely on a centralized certification entity; to this end, we have presented a 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. We have studied the security of the scheme, by leveraging on the security of the scheme introduced in Section 5.3, and by introducing new security proof to show that colluding CAs cannot forge Credentials nor Matching References for another CA.

In the second part of the Chapter we have presented a scheme that only supports classic Secret Handshakes, wherein, however, Credentials can be formed in an ad-hoc fashion by group members themselves. The scheme builds on top of some well-established cryptographic primitives such as secret sharing, threshold signatures and oblivious signature-based envelopes.

Part II

Use Cases

Chapter 7

Secret Interest Groups in Social Networks

7.1 Introduction

In this Chapter we present the first framework that allows the creation of Secret Interest Groups (SIGs) in Online Social Networks (OSNs); SIGs are self-managed groups formed outside of the social network, around secret, sensitive or private topics. Members exchange Credentials that can be used inside the social network to authenticate upon friendship requests or to secure user-generated content. To this end we leverage the cryptographic algorithms presented in Section 6.3 to build a practical scheme, and we describe a java implementation of the framework for Facebook.

In the sequel of this Chapter we first study the operational and security requirements of a generic SIG framework; then, we describe the implementation of the SIG framework suited to the Facebook OSN platform. To the best of our knowledge this work represents the first effort towards a solution for this interesting problem.

7.2 Problem Statement and Motivation

Ever received a friendship request on Facebook reading "Hi, I'm John Smith, add me as a friend, we were classmates at university“, remaining clueless about who this person is? Or ever been told that there is a profile under your name on a given social network, except you have never created such profile? If you are avid social network users, the answer to the first question is most likely yes, and a good percentage will answer yes to the second question too.

Indeed Online Social Networks (OSN) are becoming one of the most prominent communication technologies. Platforms as Facebook now count millions of users that are sharing information every day. Since the content is in many cases hosted on the OSN provider, OSN users can be profiled and offered detailed advertisement to support the OSN provider's revenues from advertisement.

A problem which is particularly felt among social network users is identity theft and identity spoofing [CT09; Pir08]. The root of the problem is that in many OSNs there is little or no verification that a person that joins the social network is really who he or she claims to be. This shortcoming needs to be combined to a second one: social network users base the decision on whether to accept a friendship request on name, pictures and fragments of text, information that is often easily retrievable elsewhere on the Internet. It is therefore relatively simple [BSBK09] for an attacker to set-up a profile on an OSN, either impersonating a legitimate party or based on an imaginary identity, and then to convince other users to accept friendship request, in order to have access to their private information that is shared with friends.

To improve the security of this process, users could be asked to provide Credentials along with the friendship request. However to be meaningful, Credentials cannot be self-generated, but need to be generated and maintained after verification by a third party: this task is cumbersome and expensive and on the one hand, it is unrealistic to expect a central entity (the social network provider for example) to attend to it for free, and on the other hand, users are not likely to pay for such service.

A viable solution consists on users creating ad-hoc, trusted groups *outside of the social network*, issuing group membership Credential and presenting such Credentials upon friendship invitations within the social network; the same happens with friendship, which is formed outside of the social network and then leverages on the social network to foster communications between friends.

A natural evolution of the aforementioned trusted friends groups are Secret Interest Groups (SIG), user-created groups with particular attention to confidential or simply privacy-sensitive topics. Indeed users of online social networks are often also exchanging personal and sensitive material; moreover, OSNs are more and more the theater of political, religious debate, often used as means to exchange confidential material that cannot go through official channels as it has been the case for instance in Iran during the recent post-electoral turmoils [TW09].

Our goal in this Chapter is therefore the creation of a framework that supports the formation and evolution of Secret Interest Groups. With our framework, users are able to handle the joining and leaving of members, to revoke or grant administration privilege to members, and to grant Credentials that members can use to secure their relationship with other members in a social network. The design of the framework raises two challenges: the framework should (i) suit the requirements of ad-hoc groups, namely, it should not require a centralized entity but operate in a distributed fashion and (ii) it should be possible to implement it in a real OSN, with all the constraints thereof, for instance, the scarcity of direct connections between users, communication patterns centralized at the OSN servers and the fact that users are not necessarily simultaneously online.

The SIG framework that we have *designed and implemented*, supports the aforementioned technical requirements along with the high security requirements akin to secret groups. These groups are secret in the sense that membership to a SIG is a sensitive information that users are reluctant to expose publicly. A SIG can be for instance centered around religious, political, sexual interests, such that users are very interested to interact with other users that share the same interest, but extremely reluctant to admit publicly that they belong to such an interest group. This notwithstanding, a SIG can be centered around less secret topics, that still represent privacy-sensitive topics requiring a certain degree of security or privacy. The proposed framework therefore addresses the more complex case of secret groups, but can be applied to simple private user groups too.

7.3 Design of the SIG Framework

The SIG framework can be split into two main parts: *OSN external* and *OSN internal*. As the names suggest, the main difference is that some of the algorithms of the SIG framework will operate outside of the social network, independently of it, while other algorithms will be executed in the social network, using the social network itself as a data transport medium, in order to secure further communications among the social network users.

OSN external algorithms deal with the creation and maintenance of the Secret Interest Group outside of the social network: as real friendship happens outside of the social network and is then used to create some links inside of it, a secret group is created outside of the social network and it is used to secure friendship links established within. *OSN internal* algorithms instead deal with authentication, handshaking and encryption of content among users of the social network, using the social network as a data transport medium for cryptographic messages.

7.3.1 OSN external

All OSN external algorithms must take into account that the SIG is an ad-hoc group. We will therefore adopt three design guidelines: (i) there should be no central entity, (ii) any entitled user should be able to act as a central entity and (iii) algorithms should be "thresholdized", in the sense that to be performed successfully they need the cooperation of at least a minimum number of entitled users. Respecting these guidelines results in the role of the central entity being split and spread among entitled users.

We assume that the creation of a new SIG group is the task of an initial set of *SIG managers*, that creates the group and handles the joining of the first *SIG members*. The initial set of SIG managers may reserve the ability of handling the joining of other users to themselves only, or may endorse other users with this capability as well. The difference between SIG managers and SIG members is substantially that SIG managers are normal SIG members who are also responsible for appointing new SIG managers and to allow new members to join the SIG. From this, we derive the first two requirements of SIG Join:

- **RExt1**: at any point in time, the set of SIG managers must be non-empty;
- **RExt2**: only a subset of SIG managers can appoint new SIG managers;

To enhance the security of SIGs, we require that one SIG manager alone is not able to perform either tasks: both procedures require instead a minimum number of SIG managers to be involved in their execution.

- **RExt3**: appointing new SIG managers and handling the joining of new members are distributed tasks, that no SIG manager alone can handle; instead, a joint effort of a minimum number of t SIG managers is required;

Prior his enrollment in the SIG through a join operation, a user must undergo offline verification carried out by a SIG manager; the purpose of this verification is to ensure that all members are consistent with the SIG join policy. This procedure is group-specific and requires different controls depending on the nature of the SIG: for example, in a SIG revolving around political militancy, party membership, background check or face-to-face interview could be the check that a user is required to pass before being admitted in the group; for a SIG representing a project consortium, it may be instead sufficient to send the membership Credentials using the consortium email address. The same must be true for SIG managers.

- **RExt4**: SIG managers will admit new SIG members or new SIG managers only after checking their compliance to the SIG join policy;

SIG members and SIG managers receive membership Credentials and managership Credentials to certify their roles. An additional requirement aims at making sure that these Credentials cannot be forged or modified:

- **RExt5**: no coalition of less then t SIG members or SIG managers is able to forge a new Credential (both membership or managership);

The existence of Credentials comes along with the requirement for revocation. We will treat the revocation of SIG managership Credential differently from simple SIG membership Credentials. Credential revocation can be performed based on either reactive techniques or proactive ones. The proactive technique consists of embedding time-limits in the Credentials, or forcing periodic updates of the Credentials with short lifetime. The reactive technique singles out revoked Credentials by publishing a revocation handle for each revoked Credential to an authenticated public list.

The fact that a SIG manager Credential falls into the hand of an attacker is not so dangerous, thanks to requirement **RExt3**: an attacker would indeed need to get hold of at least t SIG managership Credentials, and since t is a parameter of the system, this threshold can be set based on the desired degree of security. The most suitable solution to counter the theft of SIG manager Credentials is therefore a proactive strategy: managership Credentials can be periodically updated so as to make sure that the probability that an attacker gets hold of at least t valid managership Credentials is arbitrarily low.

As for SIG membership Credentials, revocation is required when either a SIG member got his membership token stolen or when he no longer qualifies for membership. The loss of a SIG membership Credential is more critical then the loss of SIG managership Credentials, since SIG membership Credentials can be used directly to authenticate to another SIG member or to access content of a SIG member, as we shall see in the next Section; therefore a reactive revocation approach is required. Promptly detecting that a Credential has fallen in the hands of an attacker is a vital requirement: for stolen Credentials this is quite straightforward, since the legitimate user that suffered the theft can realize it and report it (the situation is a bit more complex in case Credentials are stolen due to a virus/trojan compromising the system but leaving the attacker unaware, but this aspect is out of the scope of this thesis). Much more complex is the case of detecting a once legitimate user betraying his allegiance to the SIG and colluding with an attacker, or becoming one himself: however this aspect is orthogonal to our work and represents a separate area of research. We therefore assume that the following assumption holds:

- **RExt6**: stolen SIG membership Credentials or Credentials belonging to a user that has become malicious are eventually detected as such;

Notice that Credential theft or an insider becoming malicious are very different issues from identity theft and identity spoofing, against our scheme offers a protection. Indeed the latter are attacks that are very simple to mount, requiring to fake a real user profile with some publicly available data, and to lure some individual into believing that he is interacting with the legitimate user. The former instead requires to steal Credentials that are kept secret and can be very well protected, or to convince a user – who initially was granted membership to the SIG – to betray his allegiance to the SIG.

7.3.2 OSN internal

A SIG member eventually wants to add another alleged SIG member to his list of friends through the standard social network invitation process, exchange content or chat in a secure way with that member, both operations that occur within the social network. Two main algorithms are therefore required: mutual authentication of two SIG members and encryption of content for fellow SIG members. The first requirement is then straightforward:

- **RInt1**: only a legitimate SIG member can successfully authenticate to another SIG member or receive content from the latter;

Keeping in mind that SIGs are by definition secret or privacy sensitive, the invitation process is crucial, because during the invitation process, a legitimate member (the inviter, the invitee or both) does not yet know whom he is interacting with. Indeed, sending a friendship request on the grounds of common SIG membership would imply admitting to belong to the SIG for the inviter; accepting the request would mean the same for the invitee. Inviter and invitee have no incentive in disclosing their SIG membership unless they can be sure that they are interacting with another SIG member.

Such authentication problem can be solved by Secret Handshakes. Upon handshake between two SIG members, due to the nature of SIGs, users are reluctant to disclose their affiliation to the SIG: they want to do so just when they are sure to be interacting to another legitimate SIG member. From this observation, we derive the following requirement:

- **RInt2**: when two OSN users are trying to authenticate as SIG members, either both learn that they both belong to the SIG or they do not learn anything at all;

This implies that proof of membership and verification of membership happen simultaneously and atomically. Revocation should respect this requirement, in that revocation of a SIG membership Credential should prevent its owner from both proving or verifying membership to the SIG.

7.3.3 Security and Adversarial Model

There are mainly four different actors in our SIG framework: the OSN provider, the OSN user who is not a SIG member, the OSN user who is a simple SIG member and finally the OSN user who is a SIG manager. Each of these actors have different goals and different capabilities.

The OSN provider can be modeled as a Dolev-Yao [DY83] type of intruder: although unlikely to meddle with the users' content, the OSN provider effectively controls the network and in theory has the possibility to read, modify and drop each and every message that is being exchanged in the OSN. In reality, the OSN provider is more likely to behave as a Honest-But-Curious adversary, whose goal is to perform data-mining and to profile users so as to gather data that – depending on the regulations of the country where the OSN provider has its legal body – can be sold to third parties, or to simply use this information to offer specific ads to users thus gaining revenues from advertising. From the point of view of a SIG, the OSN provider is interested in passive, monitoring attacks, such as discovering whether a user is part of a SIG, in discovering the nature of a SIG and the type of content that is being protected before exchange among SIG members. The OSN provider can of course spawn fake OSN users, but cannot become a SIG member since SIG membership is given only after compliance check with the SIG join policy (**RExt4**). The main goals of this type of attacker are therefore linking different users to the same SIG, thus creating a list of SIG members; also, tracing the same member over multiple executions of SIG related operations.

Simple OSN users can also be modeled as Dolev-Yao intruders for the following reason: the OSN provider (a Dolev-Yao attacker by definition) can both spawn OSN users or collude with real OSN users. Thus, the objectives of misbehaving simple OSN users are similar to the OSN provider, with the addition of active attacks, where the attacker engages in the authentication protocol with legitimate SIG users with the objective of masquerading as a legitimate SIG user.

A SIG manager/member has already access to the information a simple OSN user/OSN provider is looking for. Under the assumption that the revocation mechanisms hold, a SIG member/manager that loses or maliciously gives away Credentials will eventually be caught. The goal of such type of attacker is therefore to generate fresh SIG membership Credentials to circumvent revocation, or to appoint new SIG managers without the consent of a majority of managers.

7.4 The SIG Framework

In order to build a scheme that satisfies the aforementioned requirements, we leverage on a number of well established cryptographic primitives, introduced in Section 6.3.1.

In this Section we describe our SIG framework based on the algorithms described in Section 6.3.3.

The SIG starts with the invocation of Setup by the initial set of n SIG managers, the SIG initiators. At the end of the algorithm, the SIG managers have picked a message that will be used later on for the handshake between two users on the social network, they have agreed on a known public key and they have distributed secret shares of an (unknown) private key, representing the SIG managership token. Notice that the actual private key is unknown; it could be known if at least t SIG managers colluded and reciprocally revealed their shares, however the threshold t can be set accordingly in order to discourage such attempts. In situations where hierarchy of SIG managers is important, each SIG managers can be supplied a different number of shares according to their importance.

The algorithm UpdateManagers can be invoked by at least t SIG managers to redistribute *different* shares of the *same* SIG private key to a *different* set of SIG managers, fixing a new threshold t'. The reasons for invoking the UpdateManagers are mainly twofold: (i) if the group grows, new SIG managers need to be appointed and therefore receive shares $SK^{i}_{\mathcal{SIG}}$ of the private key; (ii) when a group of (less then t) SIG managers needs to be revoked[1], t other SIG managers can generate a different set of shares, not related with the old set, and distribute the new shares to all managers but the ones whose manager rights need to be revoked, thus effectively preventing the latter from further executing SIG manager tasks. Regular invocations of UpdateManagers can be scheduled so that new shares are proactively being distributed and the population of SIG managers can be both purged of elements that are no longer trustworthy and enriched with new trusted ones.

t SIG managers can also issue SIG membership tokens, that users can use to authenticate on the social network. To this end, t SIG managers execute GrantMembership, issuing to the new SIG member a signature representing the SIG membership token. Once group members receive their membership tokens, they can interact more securely on the social network platform, using the following algorithms:

[1]Notice that revoking the SIG managership of t SIG managers or more is not feasible since in principle they can reconstruct the SIG secret key $SK_{\mathcal{SIG}}$; nonetheless, this can be prevented by setting t with a value that is sufficiently large.

EncryptForSIGMember : this algorithm is executed by a SIG member (sender) that wants to encrypt some content prior to its publication on the social network platform, to be received by a second SIG member (receiver); receiver and sender engage in the OSBERound algorithms, acting as P1 and P2 respectively; output of OSBERound is a key that the sender will use to encrypt the content before its publication on the social network. Only if the receiver is a legitimate SIG member, will he be able to reconstruct the key and therefore decrypt the content of the message;

SIGMembersHandshake : this algorithm is executed by two SIG members that want to authenticate to one another as members of the SIG; the two members engage in an execution of Handshake, at the end of which, each party obtains two keys; the two parties then have to prove one another knowledge of both keys simultaneously; if this is successful, the Handshake is successful;

EncryptForSIGMember can be used for instance when trying to send a message or to post some content that is stored in the OSN servers and displayed to the recipient upon its access to the OSN; SIGMembersHandshake instead is the ideal candidate for securing the friendship invitation process in the OSN, but can also be used to secure synchronous events like chat sessions. For clarity's sake, SIGMembersHandshake is summarized in Figure 7.1: two users, U_1 (holding the signature (w_1, v_1)) and U_2 (holding the signature (w_2, v_2)) engage in two OSBERound sessions establishing two keys each. Then, they engage in a challenge-response protocol to prove to one another knowledge of the keys thus computed: as an example for the latter, we have decided to use a challenge-response protocol similar to the one used in Kerberos [NT94], with the addition that E is an authenticated encryption mode of operation for cryptographic block ciphers, such as OCB [RBBK01] (Offset Codebook Mode): this way U_1, upon receipt of the last message, is already able to tell whether he is interacting with a legitimate SIG member.

7.5 Implementation in Facebook

In this Section we describe the implementation of a working proof-of-concept of the SIG framework. We have chosen to focus our implementation efforts only to the OSN internal part, and have picked Facebook as our target OSN. These choices are motivated

$$\boxed{\begin{array}{lll}
U_1 \longrightarrow U_2 & w_1 = g^{e_1}, g^{r_1} \\
U_2 \longrightarrow U_1 & g^{r_2}, w_2 = g^{e_2} \\
U_1 & \text{computes } K_1 = \left((g^x)^{w_2}\, w_2^{h(M_{\mathcal{SIG}})}\right)^{r_1} \\
 & K_1' = (g^{r_2})^{v_1} \text{ and } k_1 = H(K_1 || K_1') \\
U_2 & \text{computes } K_2' = \left((g^x)^{w_1}\, w_1^{h(M_{\mathcal{SIG}})}\right)^{r_2} \\
 & K_2 = (g^{r_1})^{v_2} \text{ and } k_2 = H(K_2 || K_2') \\
U_2 \longrightarrow U_1 & E_{k_2}(N) \\
U_1 \longrightarrow U_2 & E_{k_1}(N+1)
\end{array}}$$

Figure 7.1: SIGMembersHandshake and relative challenge-response upon friendship invitation.

by the fact that the implementation of the OSN external algorithms does not raise any interesting challenge; the choice of Facebook as OSN platform is motivated by its popularity, which could ease the acceptance and spreading of our SIG framework.

The OSN internal part of our SIG framework has been implemented as a java http proxy. The intended use case is the following: a SIG member runs the proxy, which intercepts only requests toward Facebook servers. The proxy modifies requests and responses, running the Secret Handshake protocol upon membership invitation and chat events; notification of the success or failure of the protocol is provided to the user through modifications of the html that is displayed in the browser. Among the features that still need to be implemented, on which we are concentrating our efforts, the two most prominent ones are the encryption of messages and a feasibility study for the porting of the software to a standard Facebook application.

The main challenge in implementing the OSN internal algorithms as part of a real social network, is that users of a social network are not always online, and cannot communicate directly, whilst the Secret Handshake protocol – for instance – is an interactive protocol. The challenge is therefore to adapt an interactive protocol to a non-interactive environment. In the sequel of this Section we will therefore describe in details how the proxy operates upon a friendship invitation event at both the inviter and the invitee.

In Figure 7.2 we can see how the proxy operates upon a friendship request, triggered by message 1. The message is not forwarded immediately to the Facebook servers; instead the proxy looks up the profile of the invitee, extracting from the "About me"

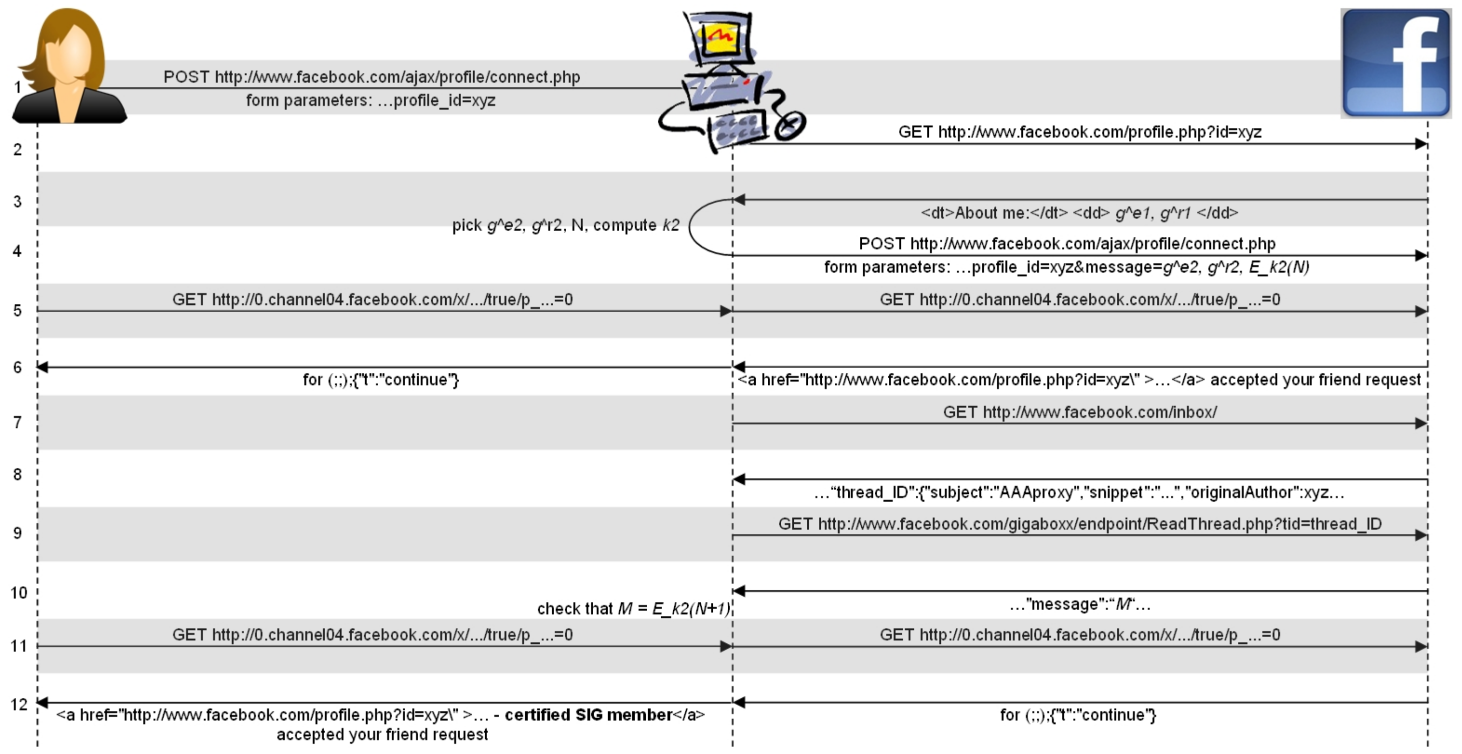
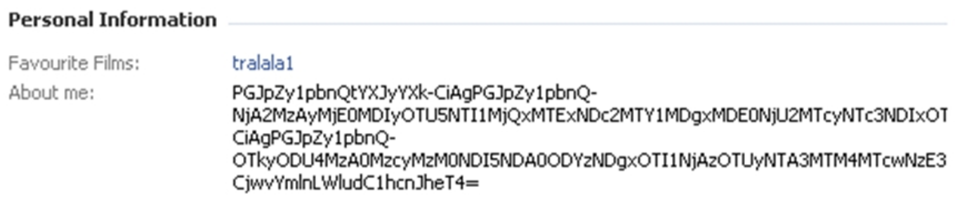

1
POST http://www.facebook.com/ajax/profile/connect.php
form parameters: …profile_id=xyz
2
GET http://www.facebook.com/profile.php?id=xyz
3
<dt>About me:</dt> <dd> g^e1, g^r1 </dd>
pick g^e2, g^r2, N, compute k2
4
POST http://www.facebook.com/ajax/profile/connect.php
form parameters: …profile_id=xyz&message=g^e2, g^r2, E_k2(N)
5
GET http://0.channel04.facebook.com/x/…/true/p_...=0
GET http://0.channel04.facebook.com/x/…/true/p_...=0
6
for (;;);{"t":"continue"}
<a href="http://www.facebook.com/profile.php?id=xyz\" >…</a> accepted your friend request
7
GET http://www.facebook.com/inbox/
8
…"thread_ID":{"subject":"AAAproxy","snippet":"…","originalAuthor":xyz…
9
GET http://www.facebook.com/gigaboxx/endpoint/ReadThread.php?tid=thread_ID
10
check that M = E_k2(N+1)
…"message":"M"…
11
GET http://0.channel04.facebook.com/x/…/true/p_...=0
GET http://0.channel04.facebook.com/x/…/true/p_...=0
12
<a href="http://www.facebook.com/profile.php?id=xyz\" >… - certified SIG member</a> accepted your friend request
for (;;);{"t":"continue"}

Figure 7.2: Operations of the proxy upon a friendship request.

field of the profile (as can be seen in Figure 7.3) the invitee's handshake (messages 2 and 3).



Personal Information

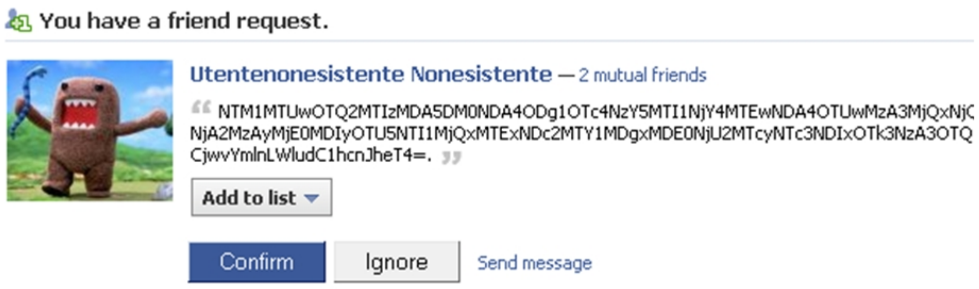
Favourite Films: tralala1

About me:

PGJpZy1pbnQtYXJyYXk-CiAgPGJpZy1pbnQ-
NjA2MzAyMjE0MDIyOTU5NTI1MjQxMTExNDc2MTY1MDgxMDE0NjU2MTcyNTc3NDIxOT
CiAgPGJpZy1pbnQ-
OTkyODU4MzA0MzcyMzM0NDI5NDA0ODYzNDgxOTI1NjAzOTUyNTA3MTM4MTcwNzE3
CjwvYmlnLWludC1hcnJheT4=

Figure 7.3: *"About Me"* section of a profile containing the first round of the handshake.

Then the proxy creates its own handshake message, derives the key which is used to encrypt a random nonce N. Handshake message and encrypted nonce are then serialized, base64'ed, and included as a POST parameter "message" of message 1. This parameter usually contains the message that a user can send upon a friendship invitation to the invitee, as shown in Figure 7.4.



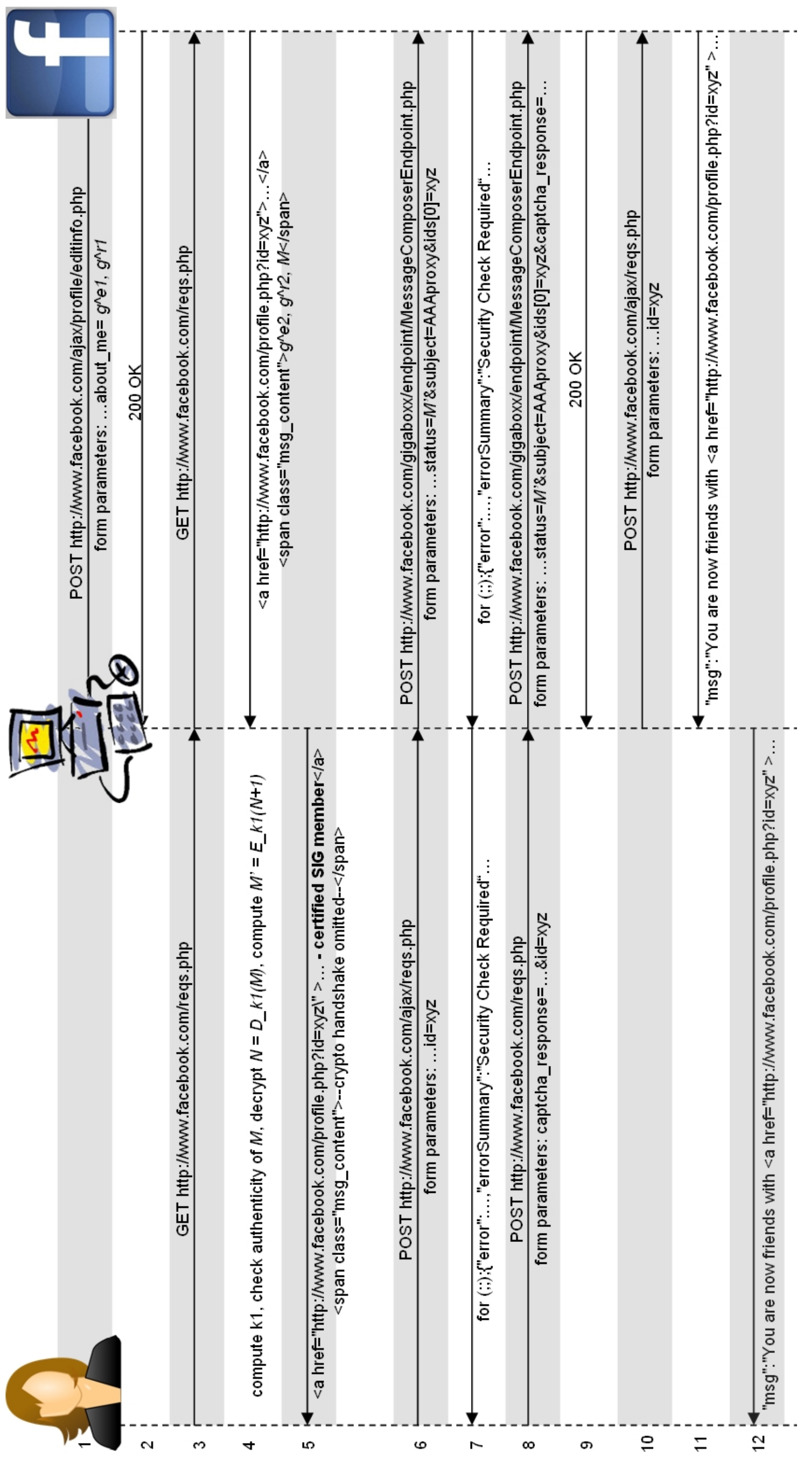
You have a friend request.

Utentenonesistente Nonesistente — 2 mutual friends

" NTM1MTUwOTQ2MTIzMDA5DM0NDA4ODg1OTc4NzY5MTI1NjY4MTEwNDA4OTUwMzA3MjQxNjC
NjA2MzAyMjE0MDIyOTU5NTI1MjQxMTExNDc2MTY1MDgxMDE0NjU2MTcyNTc3NDIxOTk3NzA3OTQ
CjwvYmlnLWludC1hcnJheT4=. "

Figure 7.4: Message sent upon friendship invitation from the inviter to the invitee.

The resulting message is finally forwarded to the Facebook servers (message 4). Let us assume that the invitee eventually accepts the request; the inviter is notified in a number of ways depending on whether he is online or not at the moment of the acceptance: the proxy intercepts this event in all its possible forms; as an example, in message 5 and 6, the browser is notified through the response to an infinite javascript loop that originates AJAX requests to fetch the updates. As can be seen, the confirmation message is not sent back directly to the browser. Instead, the proxy searches in the Facebook inbox for a message from the invitee with the response to the challenge (messages 7 to 10). If none is found or the message (after base64 decoding and deserialization) cannot be decrypted to be $N + 1$, then the standard acceptance of message 6



1 POST http://www.facebook.com/ajax/profile/editinfo.php
form parameters: ...about_me= *g^e1, g^r1*

2 200 OK

3 GET http://www.facebook.com/reqs.php
GET http://www.facebook.com/reqs.php

4 compute k1, check authenticity of *M*, decrypt *N* = *D_k1(M)*, compute *M'* = *E_k1(N+1)*
<a href="http://www.facebook.com/profile.php?id=xyz">...</a>
<span class="msg_content">*g^e2, g^r2, M*</span>

5 <a href="http://www.facebook.com/profile.php?id=xyz\" >... **- certified SIG member**</a>
<span class="msg_content">--crypto handshake omitted--</span>

6 POST http://www.facebook.com/ajax/reqs.php
form parameters: ...id=xyz
POST http://www.facebook.com/gigaboxx/endpoint/MessageComposerEndpoint.php
form parameters: ...status=*M'*&subject=AAAproxy&ids[0]=xyz

7 for (;;);{"error":...,"errorSummary":"Security Check Required"...
for (;;);{"error":...,"errorSummary":"Security Check Required"...

8 POST http://www.facebook.com/reqs.php
form parameters: captcha_response=...&id=xyz
POST http://www.facebook.com/gigaboxx/endpoint/MessageComposerEndpoint.php
form parameters: ...status=*M'*&subject=AAAproxy&ids[0]=xyz&captcha_response=...

9 200 OK

10 POST http://www.facebook.com/ajax/reqs.php
form parameters: ...id=xyz

11 "msg":"You are now friends with <a href="http://www.facebook.com/profile.php?id=xyz" >...

12 "msg":"You are now friends with <a href="http://www.facebook.com/profile.php?id=xyz" >...

Figure 7.5: Operations of the proxy upon a friendship response.

is forwarded back to the client; otherwise a modified confirmation message that highlights the SIG membership of the invitee, like the one of message 12, is fed back to the browser.

Figure 7.5 shows instead how the proxy operates on the invitee's side. At first, the proxy publishes the invitee's handshake message in the "About me" section of the invitee's profile (messages 1 and 2). The operations begin with the client visiting the page with the pending requests (message 3). The page is fetched and the message that the inviter has included in the invitation is decoded and deserialized to the handshake message and the inviter's challenge M. Then the proxy, using the handshake message that had been serialized an base64'ed in the "About me" section of the profile (message 4), derives the key and attempts the decryption of M into N; since the encryption scheme – AES OCB – provides authenticity, a successful decryption implies that the inviter is a certified SIG member. In this case, the html that is sent back to the client (message 5) is modified so as to notify that one of the inviters is a SIG member. At this point, the user may decide to accept a friendship request. If so, the acceptance message is not forwarded right away; first, a Facebook message is composed, with the string "AAAProxy" as subject, and with the encryption of $N + 1$ in the body (message 6). Facebook uses captcha to prevent proxies like ours from performing automated actions (message 7); the proxy cannot clearly solve the captcha challenge, but can nevertheless forward the request back to the client to have it solved instead. The message, with the solution to the captcha is then sent to the Facebook server (message 8). Upon receipt of the confirmation (message 9), the proxy finally accepts the original friendship request and forwards the response of the server back to the client (messages 10 to 12). Right after the friendship acceptance has been sent, the steps of messages 1 and 2 are repeated, thus creating a new handshake message to be used for a new request. An arbitration protocol needs to be devised in case two invitations use the same handshake message.

7.5.1 What still needs to be implemented

There are several missing features in our framework, which we list in this Section.

First of all, we have not implemented the EncryptForSIGMember algorithm: this algorithm could be used for instance to encrypt Facebook mails that are exchanged between users. However, encrypting messages posted on user's Walls would require a different algorithm. Indeed Facebook mails can be modeled as one-to-one communications, whereas a post on a Wall follows a one-to-many pattern, and would require a different type of algorithm.

The SIGMembersHandshake algorithm can however be used to encrypt a different type of communications, namely chat events: indeed a chat is a synchronous one-to-one communication, perfectly suited to be secured by running a Secret Handshake first, agreeing on a key and using it to encrypt the sentences that are exchanged between the two users.

Another aspect that may require improvement is that simple base64 encoding is used to exchange cryptographic messages on Facebook. This choice gives raise to two issues: first, the OSN provider may refuse to be used to exchange binary unintelligible information over fields that are meant to exchange short text strings. If our framework were to become popular, Facebook would most certainly attempt to stop it by banning profiles that misuse these fields. Second, the fact that a user publishes base64'ed strings in the About Me section of his profile, may raise suspicions on the fact that the user may be a member of some group, although it can be showed that from an information security perspective, the information published therein does not leak any information on group membership. These two problems may be solved by using a different way of encoding the binary information, perhaps one that maps binary data to a plausible text in some language.

Finally, the way our application reacts to a successful or unsuccessful handshake can be extended. Indeed, in its current state, the application reacts as follows: the invitee is the first one to know if the inviter is a SIG member. The choice whether to accept or not the friendship is however still in the hands of the user. On the other hand, the inviter knows whether the invitee is a SIG member or not only after the friendship has been confirmed by the invitee. Potentially, a malicious invitee disposes of a window of time in order to access the inviter's profile regardless of its membership. The application can be improved as follows: the inviter sets the access rights for the (still unverified) invitee to a very minimum. Then, when the friendship has been confirmed and the application receives a response as to the invitee's membership, the application can either cancel the newly established friendship in case of a negative response, or enhance the access rights for the (now verified) group member.

7.6 Conclusion

In this Chapter we have focused on the problem of securing user interaction in online social network, mainly through the creation of self-managed user groups that hand out Credentials to their members. Then, secret-handshake based authentication and content encryption are used in the social network. To this end we have defined a set of requirements, sketched a security model, presented a framework of cryptographic protocols and introduced a proof-of-concept java implementation, working in the ever growing Facebook platform.

The step from design to implementation has raised interesting challenges. Indeed the protocol that we have presented may guarantee a degree Unlinkability of Users by exchanging one-time Credentials; the ad-hoc and threshold nature of the protocol can make this approach realistic, since it could be assumed that at any point in time, at least t SIG managers are available and able to supply new Credentials. However, if we desire to use Facebook as a data transport medium for cryptographic messages, linking is by definition possible since upon joining the Facebook network, users are assigned a univocal profile identifier, which is carried at any message exchange. It is therefore interesting to notice that a protocol that in theory preserves a security property loses it once it is implemented on a given platform.

This work is also relevant for another aspect: it constitutes a significant real-life application for Secret Handshakes. It could be debatable that not every Secret Interest Group requires such high security requirements, but certainly – seeing how Facebook and other OSNs are becoming the arena for exchange of very sensitive and private information – we expect that this requirement will arise, and our scheme represents a first step towards meeting this requirement.

Chapter 8

RFID-Based Supply Chain Partner Authentication

8.1 Introduction

The growing use of Radio Frequency IDentification (RFID) in supply chains brings along an indisputable added value from the business perspective, but raises a number of new interesting security challenges. One of them is the authentication of two participants of the supply chain that have possessed the same tagged item, but that have otherwise never communicated before. The situation is even more complex if we imagine that participants to the supply chain are business competitors.

In this Chapter we present a novel cryptographic scheme that solves this problem. In our solution, users exchange tags over the cycle of a supply chain and, if two entities have possessed the same tag, they can agree on a secret common key they can use to protect their exchange of sensitive business information. No rogue user can be successful in a malicious authentication, because it would either be traceable or it would imply the loss of a secret key, which provides a strong incentive to keep the tag authentication information secret and protects the integrity of the supply chain. The protocol presented in this Chapter shares some similarities with Secret Handshakes. We provide game-based security proofs of our claims, without relying on the random oracle model.

8.2 Motivation

Radio Frequency IDentification (RFID) is a modern technology that supports tracking and tracing of tagged items in supply chains. Each item is equipped with an RFID tag that carries a unique identifier. This tag can be read via radio frequency communication and multiple tags can be read at once.

There are active and passive RFID tags. Active RFID tags have their own power supply while passive tags solely operate on the power of the signal emitted by the reader. The reader is a special device that can interoperate with the tags and read the identifiers stored in their memory. More complex and powerful tags can store information in memory and even perform simple cryptographic operations such as hashing.

A major application of RFID is supply chain management [WB08; BWL06; AM05]. In the supply chain each item can now be tracked using its unique identifier. The benefit of this tracking and tracing technology unleashes its true potential, when the supply chain partner share their event data. An event happens when a tag is read. At its most basic level this generates a tuple

$$\langle organization, identifier, timestamp \rangle$$

This tuple is usually augmented with additional information, such as reader identifier, type of event (e.g. receiving, shipping, unpacking, etc.), and additional fields depending on the type of event.

Companies are interested in sharing information linked to events for many reasons: first, a consumer may be interested in knowing the steps that the product he purchased has gone through. A company may need to recall flawed products and be interested in knowing the list of retailers that have actually sold them. Results from collaborative research projects such as Bridge or Ko-RFID [Kon] show that companies are reluctant to reveal their participations to a supply chain for a number of different reasons, such as fear of espionage or because it could possibly be embarrassing to reveal participation to the production of a severely flawed product recalled from the market. As a result, information is stored safely by each partner and willingly disclosed only after successful authentication.

In order to share the data linked with events, companies connect to a global network, currently being standardized by the EPCglobal consortium. This network contains a discovery service, which stores all companies that have event data for a specific tag. In order to retrieve all information about a tag, one contacts the discovery service with its request which then returns the list of all companies that handled the product. Then one can contact each company individually and ask to retrieve its event data.

The main challenge with this system, is that on the one hand companies have an incentive to share this information so as to facilitate their business, on the other, this information is highly confidential and (possibly competing) companies are reluctant to trust one another. Therefore, a big concern is the possibility of espionage of a competitor's supply chain [SS08], carried out for instance by retrieving the event data about items in a competitor's supply chain.

Imagine two companies – which might have never communicated before – that contact each other with the help of the discovery service and need to mutually authenticate: the only thing they have ever had in common is that they have both been in possession of the same tag at some point. These companies need to prove to each other that they have possessed the same tag.

There are a number of attacks that might happen in this scenario:

1. An impostor might request information about tags he has never possessed, for example in order to track the supply chain of his competitor.

2. A malicious company might supply rogue information about tags he had never possessed, for instance so as to hide the origin of counterfeited products.

A simple solution to this problem is to store a shared secret on the tag, so that everyone who possessed that tag knows it and can use it to secure subsequent communications.

Unfortunately, this solution has many disadvantages. First, there is no incentive for someone who possessed the item to preserve the shared secret, since the disclosure of the secret cannot be traced back to him. Second, the tag could be maliciously read by an outsider who does not legitimately belong to the supply chain.

Therefore, in order to have a secure solution to the problem at hand, we need to develop a scheme more complex than a simple shared secret-based mechanism.

8.2.1 Overview of the Solution

In this Chapter we present a novel scheme that solves the aforementioned problem. The intuition is that the information stored on the tag is tied to an identity. Only the holder of the trapdoor information (that identity's private key) can actually prove possession of the tag. The information stored on the tag is updated as the tag changes possession; the update is performed through the help of a trusted third party (TTP). The involvement of the TTP makes it possible to trace the item throughout the supply chain.

Our solution overcomes the disadvantages of the simple solution. If someone illegitimately requests information for a tag, he can be exposed by the TTP. On the other hand, impersonation is possible only if a party's private trapdoor is exposed. A supply chain partner that wants to let another party authenticate, must then decide to either be traceable or to reveal his secret key and relinquish all the business sensitive information to him.

An interesting property of our solution, is that RFID can be effectively used together with complex cryptographic primitives: indeed, tags just act as carrier of cryptographic envelopes, that are then used off-the-tag to perform complex security protocols. Our technology already works with the simplest tags specified by the EPCglobal standard (class 1 tags). The sole requirement is that tags must be able to store the minimum amount of information required to perform the cryptographic operations. Tags do not necessarily need to be rewritable: one could simply replace the old tag with a new tag containing the new information: the decreasing price of hardware can justify this choice.

8.3 Related Work

The security of RFID-based systems has been object of intense research over the past few years, given the many threats related to the adoption of this technology [Jue06]. Many papers have focused on privacy-related issues [GJP05; LK06; JW07]. RFID authentication protocols have received a lot of attention as well [YP08]. An interesting key distribution application for RFID using advanced cryptography has been presented in [JPP08]. Its main advantage is the use of aggregate packaging along the supply chain while maintaining user's privacy. However, to the best of our knowledge, our scheme is among the first works that address secure interaction among participants of an RFID-enhanced supply chain.

8.4 Supply Chain Partner Authentication

Assume Trent is a trusted third party that supports users in updating the information stored on the tag as the tag changes of possession. Then our supply chain partner authentication consists of the following algorithms or protocols.

Setup: Trent publishes some public parameters about the system known to every participant.

Register: A new company Alice wants to join the supply chain and is contacting Trent to register. They setup public/private information tied to Alice's identity. Trent returns to Alice her public and private information.

Initialize: Alice just created a new item and attached a tag to it. She creates the secret information on the tag. She does so without Trent's intervention.

Ship: Alice intends to ship the item to Bob and contacts Trent seeking for his support. Trent receives from Alice the information that Alice read from the tag, computes the updated information for Bob and delivers it back to Alice.

Receive: Bob receives the item and stores its information, so that he can later authenticate request for event data.

Authenticate: Alice and Bob want to mutually authenticate on the grounds of having owned the same tag. They exchange random challenges to salt the instance of the protocol, and mutually verify whether the other has owned the same tag. Afterward, they share a key which they use to protect subsequent communications.

8.4.1 Preliminaries

Given a security parameter k, let $(\mathbb{G}_1, *)$ and $(\mathbb{G}_2, *)$ be two groups of order p for some large prime p, where the bitsize of p 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)* problem is assumed to be hard. In what follows, we denote $\mathbb{Z}_q^* = \{1, \ldots, p-1\}$.

Our solution uses Identity-based Encryption as a building block. Identity-based encryption (IBE) was introduced in [Sha84] as an alternative to public-key encryption.

In IBE any string can be used as an encryption key, e.g. one can encrypt an e-mail using the recipient's e-mail address. The recipient then obtains the decryption key (for his identity) from a trusted third party after successful authentication.

The procedures of an IBE scheme are

- **Setup**: The TTP publishes public parameters.
- **Encrypt**: transforms a plain text using an arbitrary string as key into a ciphertext.
- **Get Decryption Key**: The TTP issues a decryption key for an identity.
- **Decrypt**: transforms the ciphertext using the decryption key into its original plaintext again.

The first practical IBE scheme was presented in [BF03b]. It is based on cryptographic pairings and is proved IND-ID-CCA secure under the bilinear decisional Diffie-Hellman assumption in the random oracle model.

In our scheme every party has an identity; following the Alice and Bob convention, we refer to a message encrypted for Alice as $IBE_A(m)$. We will treat identity based encryption as a building block and will build our scheme independently on top of it.

Finally we describe the hash function that is used in the scheme that we present. Our construction leverages on the work proposed by Boneh and Boyen in [BB04] and later improved by Waters in [Wat05].

Let $g \overset{R}{\leftarrow} \mathbb{G}_1$; let us also choose $n+1$ random values $u_0, u_1, \ldots, u_n \overset{R}{\leftarrow} \mathbb{Z}_q^*$; we assign $U_0 = g^{u_0}, U_1 = g^{u_1}, \ldots, U_n = g^{u_n}$. If $v \in \{0,1\}^n$ is an n-bit string, let us define $h(v) = u_0 + \sum_{i \in V} u_i \in \mathbb{Z}_q^*$, where $V \subseteq \{1, \ldots, n\}$ is the set of indexes i for which the ith bit of v is equal to 1. We also define $H(v) = U_0 \prod_{i \in V} U_i = g^{h(v)} \in \mathbb{G}_1$.

With such an approach, we can represent in $\mathbb{G}_1$ strings of size n, or alternatively, strings of arbitrary length, pre-processed with a hash function whose block size is n.

8.4.2 The Scheme

In this Section we present a solutions to the problem presented in this Chapter. We describe the implementation of the algorithms and protocols that we introduced in the last Section. RFIDAuth consists of the following algorithms:

- Setup

  According to the security parameter k, Trent chooses $(p, \mathbb{G}_1, \mathbb{G}_2, g, \hat{e})$, where g is a random generator of $\mathbb{G}_1$. He also picks $u_0, u_1, \ldots, u_n \xleftarrow{R} \mathbb{Z}_q^*$ and assigns $U_0 = g^{u_0}, U_1 = g^{u_1}, \ldots, U_n = g^{u_n}$. Finally, he picks $\alpha \xleftarrow{R} \mathbb{Z}_q^*$ and sets $S = g^{\alpha}$ and $S' = g^{\alpha^{-1}}$. The system's public parameters are $\mathsf{params} = \{p, \mathbb{G}_1, \mathbb{G}_2, g, S, S', \hat{e}, U_0, \ldots, U_n\}$. The values $u_0, u_1, \ldots, u_n$ and α are instead kept in Trent's secret storage.

  Trent finally initializes an IBE scheme and distributes its public parameters to every user in the system.

- Register

  Alice wants to register with Trent to enter the supply chain partner network. She just needs to prove her identity to Trent, and then she receives from Trent the secret key associated with her identity. In addition, she receives the value $I_A = H(A)^{\alpha}$ which she stores secretly.

- Initialize

  Alice wants to initialize a new tag. She chooses a random value $t_{tag} \xleftarrow{R} \mathbb{Z}_q^*$ and she computes $X_1 = S^{t_{tag}}\, {I_A}^r$, $X_2 = g^r$. Alice stores the pair $(X1, X2)$ corresponding to Alice's Credentials that certify her possession of the tag; Alice also stores a witness value $W = H(A)^r$, whose role will be clearer later on. Alice should destroy t_{tag} immediately. Notice that the Initialize phase does not require the intervention of Trent.

- Ship

  Alice intends to ship the item with the tag to Bob. Alice contacts Trent and indicates her intention by sending the tag's ID, A, B, the pair $(X1, X2)$ and the witness W, read from the tag upon reception. We remind the reader that $X_1 = S^{t_{tag}}\, {I_A}^r$, $X_2 = g^r$ and $W = H(A)^r$. Thanks to the witness values, Trent can check whether $\hat{e}(W, g) = \hat{e}(H(A), X_2)$, thus making sure that the tuple really corresponds to the identity of Alice. Trent in turn picks $s \xleftarrow{R} \mathbb{Z}_q^*$ and computes

$$\begin{cases} X_1' = \dfrac{X_1}{W^{\alpha}}\, {I_B}^s = S^{t_{tag}}\, {I_B}^s \\ X_2' = g^s \\ W' = H(B)^s \end{cases}$$

Then, Trent gives back the pair (X_1', X_2') and the corresponding witness W' to Alice. Upon receipt, Alice overwrites the old tuple with the new tuple on the tag.

For tracking purposes, Trent can compute $S^{t_{tag}} = \frac{X_1}{W^{\alpha}}$ and store the triple $\langle S^{t_{tag}}, A, B\rangle$ in his database. The identifier $S^{t_{tag}}$ is unique per tag and is never changed, such that the TTP can build a complete forwarding pedigree of the tag. Trent can then clearly identify any party divulging authentication information if an impostor for a tag is identified.

A $\longrightarrow$ T	tag ID, A, B, $S^{t_{tag}}$ ${I_A}^r$, g^r and $H(A)^r$
T $\longrightarrow$ A	$S^{t_{tag}}$ ${I_B}^s$, g^s and $H(B)^s$

Figure 8.1: Ship Protocol.

- Receive

Bob receives a tag and reads the pair (X_1, X_2) and the witness W. Bob checks whether $\hat{e}(W, g) = \hat{e}(H(B), X_2)$, to verify if the received tuple was indeed destined to his identity. If he ships the item further, he applies the ship protocol. Otherwise he stores the pair (X_1, X_2) in his database, associating it with the ID of the tag, to be used later on for authentication.

- Authenticate

Let us assume Bob and Alice want to authenticate, proving to one another that they have possessed a given tag. Let us assume that Bob initiates the handshake by sending the tag identifier to Alice. Notice that both Alice and Bob possess the pairs (X_1, X_2) stored upon Receive; let us add the subscript A (resp B) to identify Alice's (resp. Bob's) tuple.

Alice picks a random nonce $n_A \in \mathbb{Z}_q^*$, and then computes $IBE_B(H(B)^{n_A}, (S')^{n_A})$ and sends it to Bob. Similarly Bob picks a random nonce $n_B \in \mathbb{Z}_q^*$ and computes the pair $IBE_A(H(A)^{n_B}, (S')^{n_B})$ and sends it back to Alice.

If both Alice and Bob have taken part to the supply chain for the product identified by the tag, they can derive a common shared key,

$$\begin{aligned}
K = & \left(\frac{\hat{e}(X_{1A}, (S')^{n_B})}{\hat{e}(H(A)^{n_B}, X_{2A})}\right)^{n_A} \\
= & \left(\frac{\hat{e}(S^{t_{tag}}\ {I_A}^{r}, (S')^{n_B})}{\hat{e}(H(A)^{n_B}, g^{r})}\right)^{n_A} \\
= & \hat{e}(g, \tilde{g})^{t_{tag} n_A n_B} \\
= & \left(\frac{\hat{e}(S^{t_{tag}}\ {I_B}^{s}, (S')^{n_A})}{\hat{e}(H(B)^{n_A}, g^{s})}\right)^{n_B} \\
= & \left(\frac{\hat{e}(X_{1B}, (S')^{n_A})}{\hat{e}(H(B)^{n_A}, X_{2B})}\right)^{n_B}
\end{aligned}$$

thus proving to each other that they have legitimately handled the tag. In order to seal the handshake, they can use any challenge-response protocol known in the literature in order to prove to one another knowledge of the shared key without leaking it.

B $\longrightarrow$ A	ID
A $\longrightarrow$ B	$IBE_B(H(B)^{n_A}, (S')^{n_A})$
B $\longrightarrow$ A	$IBE_A(H(A)^{n_B}, (S')^{n_B})$
A $\longleftrightarrow$ B	challenge-response based on K
A $\longleftrightarrow$ B	data exchange protected with K

Figure 8.2: Handshake.

Subsequent communications are protected using K to setup a secure channel.

8.5 Security Analysis

In this Section we analyze the security of the presented scheme. An authentication and key agreement scheme calls for a proof showing that no attacker can fool a legitimate user into thinking that *he is* somebody and that *he has owned* a given tag.

We therefore present a single game where we prove that an attacker is not able to impersonate any other user: this includes protection from forgery, illegitimate shipping and attacks to the authentication and key agreement scheme. In the security proofs we do *not* rely on the random oracle model [BR93].

In our scheme every ship event needs the involvement of the TTP and the security proofs confirm that this is unavoidable. Our scheme allows therefore complete tracing of the shipping events. Trent can thus built an entire shipping graph for each item. No participant outside of that graph can perform a successful handshake.

This fact has an interesting consequence: if an unauthorized party is successful in an illegitimate handshake, he could be blamed, and the TTP could also trace which participant leaked the information. Therefore there is a strong incentive not to disclose the information on the tag on purpose and the entire supply chain is tightly controlled. Let us at first state a well-known hard problem upon which the security of our scheme is based.

Definition 15 (Hardness of the *Bilinear Decisional Diffie-Hellman* Problem)**.** *The Bilinear Decisional Diffie-Hellman Problem (BDDH) is hard if, for all probabilistic, polynomial-time algorithms B,*

$$\mathsf{AdvBDDH}_B := Pr[B(g, g^a, g^b, g^c, g^x) = \top \ \textit{if}\ x = abc] - \tfrac{1}{2}$$

is negligible in the security parameter.

This probability is taken over random choice of $g \in \mathbb{G}_1$, $a, b, c \in \mathbb{Z}_q^*$. x is equal to abc with probability $\frac{1}{2}$ and is equal to a random number otherwise. This complexity assumptions is well known in the cryptographic community, and has been used in the proofs of many cryptographic schemes, for instance [CPP05].

8.5.1 Security of RFIDAuth

In this Section we investigate the security of the RFIDAuth protocol. Let us first of all analyze a simple attack and show how the scheme prevents it. A user could eavesdrop the communications that occur upon the Ship protocol, or just simply get hold of a tag and extract the pair (X_1, X_2), and try to use that tuple to engage in a successful handshake with a legitimate participant.

The scheme leverages on Identity Based Encryption to prevent this type of attack: it is clear that – since handshake challenges are destined to a user and encrypted under the public key of his identity – mere eavesdropping of tag information will not help to break the secrecy of the challenge sent, and therefore every attack of this kind is vain. We do not need to prove this, as we assume the existence of a perfect IBE scheme that does not leak any information.

Nonetheless, in order to show that the scheme is sound, we present in the next Section the security proof of resistance to impersonation, wherein we “switch off” IBE (or similarly, we give all the private keys to the attacker). What we prove, in short, is that with all the information in the hands of the adversary *but* the one associated with a challenge tag and a challenge user, the adversary is not able to impersonate the latter. This game is broad enough as to include privacy of the key exchange from an eavesdropper, collusion of several participants, forgery of rogue tag information and so forth.

8.5.1.1 Impersonation Resistance

Consider an adversary A that has as its goal to run a successful handshake – thus convincing another user that he has actually owned a tag – without disposing of the legitimate information. In particular, A does not have the pair (X_{1v_*}, X_{2v_*}) for a given user v_* and a given tag, both object of the challenge.

A is allowed to freely perform all the algorithms of the protocol. Then the simulator B Initializes a challenge tag, and yet the adversary is able to get the information to perform a successful handshake for that tag as any user of his choice (except the one object of the challenge).

Finally, the attacker picks a challenge user v_* and is required to run a successful handshake, convincing the simulator that he is user v_* having owned the challenge tag. In particular, at the end of the game, the attacker is required to output the key K. We call this game Impersonate.

Lemma 21. *If an adversary A has a non-null advantage*

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

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 a random instance (g, g^a, g^b, g^c, g^x) of the BDDH problem and wishes to use A to check whether $x = abc$. The algorithm B simulates an environment in which A operates.

Setup The simulator B sets an integer $m = 4q$ where q is an upper bound on the number of identities that the adversary will consider throughout his queries to the various protocols. B then chooses $k \overset{R}{\leftarrow} \{0, n\}$ and chooses two random vectors

$X = \{x_i\}_{i=1}^n \overset{R}{\leftarrow} \{0, m-1\}^n$ and $Y = \{y_i\}_{i=1}^n \overset{R}{\leftarrow} \mathbb{Z}_p^{*n}$. Following Boneh and Boyen [BB04] and Waters [Wat05], we define the functions $F(v) = (p - mk) + x_0 + \sum_{i \in V} x_i$, $J(v) = y_0 + \sum_{i \in V} y_i$ and $K(v)$ as

$$K(v) = \begin{cases} 0, & \text{if } x_0 + \sum_{i \in V} x_i = 0 \; mod \; m; \\ 1, & \text{otherwise}; \end{cases}$$

The simulator sets g as the one received from the BDDH challenge; the the simulator sets $U_0 = (g^b)^{p-km+x_0} g^{y_0}$ and $U_i = (g^b)^{x_i} g^{y_i}$; he then picks $\alpha \overset{R}{\leftarrow} \mathbb{Z}_q^*$, sets $S = g^\alpha$ and $S' = g^{\alpha^{-1}}$ and publishes the public parameters according to the rules of the protocol. Notice that now, $H(v) = U_0 \prod_{i \in V} U_i = g^{bF(v)+J(v)}$, where V is the set of indexes i for which the ith bit of the string at hand equals to 1.

Queries First of all, the attacker receives *all* IBE private keys: this way, the protection of IBE is disabled. In the rest of this proof therefore, we omit the notation $IBE.(\cdot)$.

The attacker can Register at his will as any identity v_i he chooses, different from v_*, receiving from Trent the value I_{v_i}.

A can Initialize any tag as any user of his choice. We remind the reader that he can perform this operation autonomously without the involvement of the simulator.

Upon execution of the Ship protocol, the attacker A sends to B the ID of a tag, two identities v_i and v_j, the pair $(X_1 = S^{t_{tag}} {I_{v_i}}^r,\ X_2 = g^r)$ and the witness $W = H(v_i)^r$. B computes

$$\begin{cases} X_1' = \dfrac{X_1}{W^\alpha} {I_{v_j}}^s = S^{t_{tag}} {I_{v_j}}^s \\ X_2' = g^s \\ W' = H(v_j)^s \end{cases}$$

as mandated by the Ship protocol, and sends the pair (X_1', X_2') and the witness W' back to A.

Finally, A can perform the receive protocol by simply storing the received tuple and associating it with the tag id.

B then Initializes a new tag, which will be the object of the challenge. A is then entitled to receive – for any user v_i of his choice – the information necessary to run a successful handshake as that user, i.e. the pair $(X1, X2)$. A therefore sends

v_i to B. If $K(v_i) = 0$, B aborts and outputs a random guess. If not, B picks a $r \xleftarrow{R} \mathbb{Z}_q^*$ and computes

$$\begin{cases} X_1 & = \left((g^a)^{\frac{-J(v_i)}{F(v_i)}} \left((g^b)^{F(v_i)} g^{J(v)} \right)^r \right)^\alpha \\ & = g^{\alpha ab} \left(g^{bF(v_i)+j(v_i)} \right)^{\alpha \tilde{r}} \\ & = S^{ab}\, {I_{v_i}}^{\tilde{r}} \\ X_2 & = (g^a)^{\frac{-1}{F(v_i)}} g^r = g^{\tilde{r}} \end{cases}$$

where $\tilde{r} = r - a/F(v_i)$. Notice that with the pair (X_1, X_2), the attacker can perform any handshake he wants, but cannot perform the ship protocol due to the missing witness W.

Challenge The attacker A then chooses an identity v_* he has not queried before; if $x_0 + \sum_{i \in V} x_i \neq km$ the simulator aborts and submits a random guess. Otherwise we have $F(v_*) = 0 \; mod \; p$, which means that $H(V_*) = g^{J(v_*)}$. B then sends as challenge the pair $(H(v_*)^c = (g^c)^{J(v_*)}, S'^c)$ according to the description of the handshake protocol. A answers with $(H(v_i)^r, S'^r)$, and then outputs the key K.

Analysis of A's response If A has won the game, $K = \hat{e}(g,g)^{abcr}$. Therefore, B can solve the BDDH problem by checking whether $\hat{e}\left(g^x, (S'^r)^{\alpha^{-1}}\right) = K$ holds. A detailed analysis of the probability that B does not need to abort has been presented in [Wat05] and we therefore omit it here.

$\square$

8.6 Conclusion

In this Chapter, we have presented a novel scheme to replace shared secrets when forwarding items tagged with RFID. Our scheme discourages disclosure of authentication information by tying it to a secret key or identity. Either one discloses the secret key or forwards the information according to the protocol specification, but if one does so, he is traceable by a trusted third party.

We have presented a protocol implementing this scheme and proved it secure in a game-based security definition based on common security assumptions. Our scheme can be applied even to the simplest tags if the information is sent along over the network.

Our scheme presents a novel approach to the problem that reaches beyond current security applications in securing the integrity of supply chains. We anticipate that, due to its simplicity in application and strong security guarantees, our scheme has wide applications in securing RFID-supported supply chains. Future work is to incorporate its security into the query answer of the discovery service.

Chapter 9

Conclusion and Future Work

In this Chapter we first review the work presented in this thesis. Then we describe few topics that we could not address in this manuscript or that could have been addressed in more details. Finally, we outline possible directions for future work.

9.1 Summary

A Secret Handshake is a protocol wherein two users disclose information about themselves to one another; this information exchange happens only under certain provisions, for instance, under the provision that the remote user is a member of the same group as the local one. On the contrary, if the provisions are not satisfied, it is required that no information is leaked as to the reason why the protocol failed. Additional requirements are that protocol instances should not be linkable to a common user or to a common certified information, such as a group or a property.

We start off with a study of the family of Secret Handshake protocols: from this study, many security and functional requirements can be derived. From a mapping between the latter and the various protocols in the state of the art, a number of unexplored combinations and, consequently, of missing protocols can be derived.

These initial considerations guide the first part of our work. At first we introduce the concept of Dynamic Controlled Matching, whereby users possessing different properties can perform a successful Secret Handshake round; however the protocol is successful only if each user has the right to match the other user's property. The central authority that generates the cryptographic tokens used in the execution of the protocol, Credentials and Matching References, retains the control over the matching ability of users. Dynamic Controlled Matching not only presents a solution that completes the range of existing Secret Handshake protocols, but it is also a generalization of other Secret Handshake schemes, namely of classic Secret Handshakes, wherein users can only match within a common group or property, and Secret Handshakes with Dynamic Matching, where users can match properties different from their own, and have the freedom to choose among properties they intend to match.

We then address the interesting topic of revocation of Credentials in Secret Handshake schemes. This topic proves to be particularly challenging for the following reason: one of the requirements of Secret Handshake schemes is Unlinkability of users. According to this security requirements, no adversary should be able to trace the same user over multiple Secret Handshake executions. Revocation however requires means of pinning Credentials to single users and labeling them so that – once they need to be revoked – the label can be revealed and interaction with such Credentials can be refused. However this approach in principles violates the requirement of Unlinkability of users. To solve this problem we present two schemes that bring revocation support to Secret Handshake with Dynamic Controlled Matching and to Secret Handshake with Dynamic Matching.

We then consider another problem, namely the role of a certification entity in Secret Handshake protocols. When introducing Dynamic Matching and Dynamic Controlled Matching, we are confronted with a scenario where a single centralized entity hands out Credentials for different groups or different properties. However in some cases this is not realistic, and different groups require different certification entities. To this end we create a scheme where a federation of independent CAs can administer such a scenario: each CA collaborates with the others but retains the control over properties falling under its control. Successful handshakes are possible also in hybrid scenarios, with users having Credentials and Matching References from different CAs. We also consider a different scenario where the role of the certification entity is distributed among all users of the scheme, thus actually eliminating such entity: new Credentials can be generated by a consensus of users vouching in for new users.

After these rather theoretical contributions, we have focused our attention on real-world use cases. At first we have presented a framework that users of an online social network can adopt to create spontaneous secret group. Users can then leverage on group Credentials in order to secure their interactions over the social network. We have also described some of the challenges that we have faced when implementing part of this framework to suit an online social network platform, namely Facebook. Finally, we have presented an authentication scheme that members of a supply chain can adopt to mutually authenticate on the grounds of having performed some work over an item at some point during the lifecycle on the supply chain. The Credentials that are used during the authentication phase are exchanged using simple RFID tags.

9.2 What is missing?

This thesis has produced contributions both from the more abstract perspective of new cryptographic protocols and through solutions to real-world use-cases. However, there are still some topics that could not be addressed or that could have been treated in more details. We discuss some of these in the reminder of this Section.

The security proofs that we have presented for the protocols introduced in Chapters 4, 5 and in Section 6.2 studies the security of isolated protocol instances. An interesting model suggested by Jarecki and colleagues in [JKT08] that we have not considered is the so called arbitrary composition model, where the adversary can initiate arbitrary concurrent Secret Handshake instances and reveal the outcome of some of them.

Another aspect that we have not investigated thoroughly is the security of the scheme introduced in Section 6.3, namely, the Secret Handshake scheme with ad-hoc group creation.

As for the second part of the thesis, in Chapter 7 we have only been able to implement the friendship invitation process, leaving several features – such as the secure publication of content – unimplemented. In addition, the implementation is only at proof-of-concept phase; in particular, several aspects can be improved, namely the exchanged cryptographic material may be better disguised in order to avoid the simple suspicion that a user might be a member of some group based on the publication of unintelligible information on the “About me” section of their profile.

9.3 Future Work

Finally, we give an overview of the possible lines of research that could be carried out based on the results presented in this thesis.

One limitation of all the protocols that we have presented in this thesis is that only one attempt to a matching is possible. There are however situations in which this could be a strong limitation. Imagine scenario where a distributed workflow has to be conducted secretly and reaches a point where, in order to continue, it needs to interact with somebody possessing either one of two properties. This could be generalized to allow for Secret Handshake schemes where user can conduct a successful matching of arbitrarily complex logical conjunctions and disjunctions of properties possessed by the remote party. Through standard Secret Handshake this could only be possible by iteratively retrying after a failed handshake, requiring possibly $O(n^2)$ attempts for a simple conjunctive query. A major piece of future work would be to improve this situation devising a protocol to address this problem.

Another interesting aspect is the following: throughout our work we have introduced the concept of Dynamic Controlled Matching; we have thus effectively decoupled the rights to prove and to verify. The right to prove is embodied in the Credential, whereas the right to verify is represented by the possession of a Matching Reference, handed out by the certification entity. In the thesis we have shown how it is possible to revoke the right to prove, i.e. to revoke Credentials. However we have not addressed the topic of revocation of Matching References. This aspect seems to be particularly challenging because in all the protocols that we have shown, Matching References are always only used locally, so once a user has received them, it is extremely difficult to forbid their use later on. A simple solution would be a complete rekeying, whereby the secret associated with the old property are changed; however, in the present schemes, this would invalidate all Credentials and Matching References for that property. It is likely that a possible solution would involve either an active CA participating in every instance of Secret Handshake, or an epoch-based revocation mechanism for Matching References or conducting the matching as a two-party computation, by sending both Credentials and Matching References to the other party.

Finally, an option that we did not pursue is the possibility of achieving Secret Handshakes using cryptographic accumulators; given an updated version of an accumulator containing all the identifiers of users having valid Credentials for a given property, users can achieve one side of a Secret Handshake by conducting a zero-knowledge proof that an identifier belongs to the accumulator. Turning this into a complete Secret Handshake seems a promising direction for research.

References

[ACHdM05] G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles, 2005.

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS*, 2007.

[AM05] A. Asif and M. Mandviwalla. Integrating the supply chain with rfid: A technical and business analysis. In *Communications of the Association for Information Systems, vol. 15*, pages 393–427, 2005.

[Aso98] N. Asokan. *Fairness in electronic commerce.* PhD thesis, Waterloo, Ont., Canada, Canada, 1998.

[BA93] Josh Benaloh and Giordano Automation. One-way accumulators: A decentralized alternative to digital signatures. pages 274–285. Springer-Verlag, 1993.

[Bag06] Walid Bagga. *Policy-based cryptography : theory and applications.* PhD thesis, PhD Thesis, 12 2006.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, pages 1–15, 1996.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, pages 26–45, 1998.

[BDS$^+$03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[BF03a] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[BF03b] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[BG85] Robert W. Baldwin and Wayne C. Gramlich. Cryptographic protocol for trustable match making. *Security and Privacy, IEEE Symposium on*, 0:92, 1985.

[Bla79] G.R. Blakley. Safeguarding cryptographic keys. In *AFIPS Conference Proceedings*, volume 48, pages 313–317, 1979.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *EUROCRYPT*, pages 156–171, 2000.

[Boy86] C. Boyd. Digital multisignatures. Cryptography and Coding, 1986.

[BP] Ronald H. Brown and Arati Prabhakar. Digital signature standard (dss).

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, 1993.

[BSBK09] Leyla Bilge, Thorsten Strufe, Davide Balzarotti, and Engin Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *WWW*, pages 551–560, 2009.

[BWL06] Ygal Bendavid, Samuel Fosso Wamba, and Louis A. Lefebvre. Proof of concept of an rfid-enabled supply chain in a b2b e-commerce environment. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 564–568, New York, NY, USA, 2006. ACM.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CJT04] Claude Castelluccia, Stanislaw Jarecki, and Gene Tsudik. Secret handshakes from ca-oblivious encryption. In *ASIACRYPT*, pages 293–307, 2004.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proceedings of Crypto 2002, volume 2442 of LNCS*, pages 61–76. Springer-Verlag, 2002.

[Cob07] Cobis Consortium. Collaborative business items: chemical drums use-case. `http://www.cobis-online.de/files/live.stream.wvx`, 2007.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[CT09] Blog on Data Security CryptoBlog and Information Theory. Social networks and social security numbers. `http://cryptoblog.wordpress.com/2009/07/08/social-networks-and-social-security-numbers/`, 2009.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In *CRYPTO*, pages 120–127, 1987.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, pages 307–315, 1989.

[DH03] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, January 2003.

[DJ97] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and applications, 1997.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[Eur07] Europol and Eurojust and Thomas Van Cangh and Abdelkrim Boujraf. Wp3-cs2: The Eurojust-Europol Case Study. `http://www.r4egov.eu/resources`, 2007.

[fac] Facebook. `http://www.facebook.com/`.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *FOCS*, pages 427–437, 1987.

[GJP05] S.L. Garfinkel, A. Juels, and R. Pappu. Rfid privacy: an overview of problems and proposed solutions. *Security & Privacy, IEEE*, 3(3):34–43, May-June 2005.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO*, pages 171–185, 1986.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[GPW$^+$04] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *CHES*, pages 119–132, 2004.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO*, pages 339–352, 1995.

[HMV03] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[Hoe08] Jaap-Henk Hoepman. Private handshakes. *CoRR*, abs/0804.0074, 2008.

[JKT06] Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Authentication for paranoids: Multi-party secret handshakes. In *ACNS*, pages 325–339, 2006.

[JKT08] Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Beyond secret handshakes: Affiliation-hiding authenticated key exchange. In *CT-RSA*, pages 352–369, 2008.

[JL07] Stanislaw Jarecki and Xiaomin Liu. Unlinkable secret handshakes and key-private group key management schemes. In *ACNS*, pages 270–287, 2007.

[JN03] Antoine Joux and Kim Nguyen. Separating decision diffie-hellman from computational diffie-hellman in cryptographic groups. *J. Cryptology*, 16(4):239–247, 2003.

[JPP08] Ari Juels, Ravikanth Pappu, and Bryan Parno. Unidirectional key distribution across time and space with applications to rfid security. In *USENIX Security Symposium*, 2008.

[Jue06] Ari Juels. RFID Security and Privacy: A Research Survey. *IEEE Journal on Selected Areas in Communications*, 24(2):381–394, February 2006.

[JW07] Ari Juels and Stephen A. Weis. Defining strong privacy for rfid. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 342–347, March 2007.

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Kon] Konstantin Tarmyshov. Ko-RFID Teilprojekt 4: Providermodelle und Integration von RFID in ERP-gestëtzte Infrastrukturen.

[LDB05] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. *Distributed Computing*, 17(4):293–302, 2005.

[LK06] Hyangjin Lee and Jeeyeon Kim. Privacy threats and issues in mobile rfid. *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 5 pp.–, April 2006.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199, 1999.

[Mea86] Catherine 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.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In *CRYPTO*, pages 417–426, 1985.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MvOV96] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[NT94] B.C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, Sep 1994.

[NT06] Samad Nasserian and Gene Tsudik. Revisiting oblivious signature-based envelopes. In *Financial Cryptography*, pages 221–235, 2006.

[Ped91a] Torben P. Pedersen. Distributed provers with applications to undeniable signatures. In *EUROCRYPT*, pages 221–242, 1991.

[Ped91b] Torben P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, volume 547, pages 522–526. Springer-Verlag, 1991.

[Pfi97] Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. pages 480–494. Springer-Verlag, 1997.

[PG99] Henning Pagnia and Felix C. Gärtner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Darmstadt University of Technology, March 1999.

[PH08] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`, February 2008. v0.31.

[Pir08] Chris Pirillo. Pownce: Social networks aren't identity networks. `http://chris.pirillo.com/pownce-social-networks-arent-identity-networks/`, 2008.

[PK96] Choonsik Park and Kaoru Kurosawa. New elgamal type threshold digital signature scheme. 1996.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. Ocb: A block-cipher mode of operation for efficient authenticated encryption. pages 196–205. ACM Press, 2001.

[RSA83] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

[SG08] Ji Sun Shin and Virgil D. Gligor. A new privacy-enhanced matchmaking protocol. In *NDSS*, 2008.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[SS08] B. Dos Santos and L. Smith. Rfid in the supply chain: panacea or pandora's box? *Communications of the ACM*, 51(10), 2008.

[TW09] Robert Tait and Matthew Weaver. How neda soltani became the face of iran's struggle. `http://www.guardian.co.uk/world/2009/jun/22/neda-soltani-death-iran`, 2009.

[Ver05] Damien Vergnaud. Rsa-based secret handshakes. In *WCC*, pages 252–274, 2005.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[WB08] Samuel Fosso Wamba and Harold Boeck. Enhancing information flow in a retail supply chain using rfid and the epc network. *J. Theor. Appl. Electron. Commer. Res.*, 3(1):92–105, 2008.

[wik10] Wikipedia, the free encyclopedia. `http://www.wikipedia.org`, 2010.

[XY04] Shouhuai Xu and Moti Yung. k-anonymous secret handshakes with reusable credentials. In *ACM Conference on Computer and Communications Security*, pages 158–167, 2004.

[YP08] Y. Yousuf and V. Potdar. A survey of rfid authentication protocols. *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1346–1350, March 2008.

[ZN01] Kan Zhang and Roger Needham. A private matchmaking protocol. `http://citeseer.nj.nec.com/71955.html`, 2001.

Appendices

Appendix A

Résumé en Français

A.1 Introduction

Le travail de cette thèse a été motivé par une analyse des cas d'utilisation présentés par les Cobis projet européen dans [Cob07]. Dans un des scénarios du projet, des barils contenant des produits chimiques sont stockés dans un entrepôt, mais les règles de sécurité imposent des restrictions sur la disposition de ces barils. Par exemple, des barils contenant des produits chimiques réactifs ne peuvent pas être stockés à proximité les uns des autres: une petite fuite de produits chimiques pourrait avoir des conséquences potentiellement désastreuses.

La solution élaborée est d'équiper chaque baril avec un dispositif sans fil. Chaque appareil échange des informations en clair sur le contenu du baril: cela permet d'effectuer des inférences sur la disposition des barils et de prendre des contre-mesures en cas de combinaisons interdites.

L'évaluation du risque de ces scénarios montre de nombreuses lacunes: la vérification des produits chimiques complémentaires est possible uniquement si le contenu des barils est diffusé en clair. Le fait que les transmissions soient en clair peut toutefois mener à des attaques liées au terrorisme ou à l'espionnage industriel.

Beaucoup de solutions cryptographiques peuvent être considérées pour résoudre ce problème: une étude des solutions possibles montre qu'elles sont très semblables mais qu'elles présentent toutefois des différences subtiles. L'objectif de cette thèse est donc l'analyse de la famille de ces protocoles, appelées Poignées de Main Secrètes.

A.2 Poignées de Main Secrètes

Une *Poignée de Main Secrète* est une forme distincte de poignée de main exprimant l'appartenance au club, groupe ou confrérie [wik10]. Habituellement, une Poignée de Main Secrète s'effectue en menant la poignée de main d'une manière spéciale afin d'être reconnaissable comme telle par les autres membres tout en paraissant normale pour des non-membres. La nécessité d'un tel échange secret initial est motivée par l'existence dans la société de rassemblements de personnes autour de sujets sensibles et donc secrets par nature.

Vu le rôle très important des communications électroniques dans notre société, il est naturel de s'attendre à ce que la discipline de l'informatique saisisse l'essence des Poignées de Main Secrètes et crée des protocoles qui peuvent être exécutées automatiquement par des appareils électroniques. Plus précisément, étant donné le caractère secret et sensible des scénarios motivant ces protocoles, il est naturel de s'attendre à ce qu'ils soient des protocoles cryptographiques.

A.2.1 Scénarios

Dans cette section nous présentons des différents scénarios où les protocoles pour la Poignée de Main Secrète peuvent être requis.

Considérons un agent secret en mission, ayant besoin de s'authentifier auprès d'un autre agent ou d'un serveur appartenant au service de renseignements. Les agents sont tenus de suivre la politique du service de renseignements de ne jamais divulguer leur Titres de Preuve, sauf s'ils sont certains d'être en relation avec des collègues membres du service de renseignements. Une conséquence intéressante des interactions suivants cette politique est une impasse: aucun des agents n'accepte de révéler ses Titres de Preuve en premier, et par conséquent la communication s'arrête.

Nous allons maintenant tourner notre attention vers un utilisateur, Alice, qui vit dans un pays où les mérites en matière de droits de l'homme sont plus que douteux. Elle est membre d'un mouvement pro-démocratie. Les membres de ce mouvement se réunissent périodiquement de façon secrète; souvent, Alice rencontre également des nouveaux membres qu'elle n'a jamais rencontré auparavant. Lorsque cela se passe, Alice a naturellement peur d'avoir affaire à des membres de la police secrète de cet état, dont le but est de découvrir les membres du mouvement pro-démocratie et les arrêter. Néanmoins, les membres légitimes ont besoin d'interagir entre eux afin de poursuivre les activités du mouvement.

Considérons maintenant les forces de police d'une fédération d'états, qui ont besoin de coopérer entre eux afin de résoudre des affaires pénales transfrontalières. Les règlements de la fédération définissent des processus officiels qui doivent impérativement être suivis par les agents de police: ces processus décrivent notamment comment les institutions doivent coopérer dans chaque cas particulier. Par exemple, un membre des forces de police d'un État doit coopérer avec le membre correspondant des forces de police d'un autre État, pour enquêter sur un scandale interne présumé. Les deux agents peuvent avoir besoin de se rencontrer secrètement, et de s'authentifier à la volée. Les deux vont certainement être réticents à divulguer leur affiliation.

Imaginons maintenant un consortium de projet dont les membres veulent devenir amis sur un réseau social et utiliser l'infrastructure du réseau social comme un outil de collaboration. Les membres du consortium nécessitent des moyens pour sécuriser le processus d'invitation: cela aide à éviter des faux négatifs, refuser la demande d'un membre du consortium légitime, ou des faux positifs, accepter l'invitation d'un utilisateur non autorisé et par conséquent interagir avec lui. Le consortium du projet peut bien exiger ces garanties pour se protéger contre l'espionnage industriel.

Ces différents scénarios montrent certaines exigences communes: dans chacun de ces exemples, les utilisateurs sont intéressés à s'engager dans un protocole d'authentification et à révéler leur allégeance, à condition que cela ne se fasse que lorsque l'utilisateur espéré est au bout du fil: un agent secret dans le premier exemple, un membre du mouvement pro-démocratie dans le second, un fonctionnaire de justice de l'état prévu dans le troisième et un membre du consortium dans le dernier. Si ces conditions ne s'appliquent pas, les utilisateurs exigent qu'aucune information ne soit divulguée.

A.3 La Poignée de Main Secrète

Les Poignées de Main Secrètes consistent en deux utilisateurs qui s'engagent dans un protocole avec l'objectif d'échanger des informations sur une *propriété*. Il y a deux actions que chaque utilisateur effectue au cours d'une Poignée de Main Secrète: une *preuve* et une *vérification*. La preuve est le moyen de convaincre l'autre utilisateur que l'on possède la propriété qui est l'objet de la poignée de main. La vérification à son tour, consiste à vérifier que l'autre partie possède effectivement la propriété l'objet de la poignée de main.

L'objectif principal des Poignées de Main Secrètes peut donc être défini comme suit:

Définition 1 (Poignée de Main Secrète)**.** *Une Poignée de Main Secrète est un protocole dans lequel deux utilisateurs u_i et u_j appartenant à un univers d'utilisateurs $\mathcal{U}$ s'authentifient comme possesseurs d'une propriété commune p_* appartenant à un univers de propriétés $\mathcal{P}$.*

Un protocole simple pour atteindre cet objectif est illustré en Figure A.1. Les utilisateurs u_i et u_j reçoivent une valeur secrète K_{p_*} associée à la propriété p_*. Les deux utilisateurs échangent deux valeurs aléatoires n_i et n_j. Après l'échange, chaque utilisateur peut calculer $k = MAC_{K_{p_*}}(n_i||n_j)$, en utilisant un code d'authentification de messages comme [BCK96]; les deux utilisateurs calculeront la même valeur k seuls si les deux possèdent la même valeur secrète K_{p_*}.

$u_i \longrightarrow u_j$	n_i
$u_j \longrightarrow u_i$	n_j
$u_j \longleftrightarrow u_i$	démontrer la connaissance de $k = MAC_{K_{p_*}}(n_i\|\|n_j)$

Figure A.1: Un protocole simple pour la Poignée de Main Secrète.

Tout d'abord, nous pouvons voir que le résultat de ce protocole est une valeur, k. Les deux utilisateurs peuvent obtenir la preuve et la vérification de la propriété p_* avec une preuve de connaissance de la valeur k. Cette valeur peut éventuellement être utilisée par les deux utilisateurs pour calculer une clé utilisée pour sécuriser les communications suivantes.

Une limitation du protocole exposé en Figure A.1 est que les actions de preuve et de vérification ne peuvent pas être séparées, car les deux sont accomplies en même temps par la preuve de la connaissance de k; à son tour, k est une fonction des valeurs aléatoires n_i et n_j et de K_{p_*}: donc, dans le protocole simple de la Figure A.1, la connaissance de K_{p_*} donne en même temps le droit de prouver et de vérifier la propriété p_*. Nous définissons maintenant le concept de séparabilité:

Définition 2 (Séparabilité)**.** *Un protocole pour la Poignée de Main Secrète est séparable si la capacité de prouver peut être accordée sans la possibilité de vérifier (et vice versa).*

Selon la Définition 2, le protocole décrit dans la Figure A.1 est non-séparable. La séparabilité se traduit notamment dans le fractionnement des secrets associés à une propriété – comme K_{p_*} dans notre exemple précédent – en deux composantes distinctes: *Titres de Preuve* et *Titres de Vérification* . Les Titres de Preuve accordent le droit de prouver à un autre utilisateur la possession d'une propriété. Les Titres de Vérification accordent la possibilité de vérifier si un autre utilisateur possède une propriété. Maintenant que nous avons officiellement présenté les Titres de Preuve et les Titres de Vérification, nous pouvons souligner le fait que, dans les protocoles pour Poignées de Main Secrètes, seuls les détenteurs légitimes des Titres de Preuve devraient être en mesure de prouver la possession d'une propriété, et que seuls les porteurs légitimes des Titres de Vérification devraient être en mesure de vérifier la possession d'une propriété. Nous pouvons donc affiner la Définition 1 comme suit:

Définition 3 (Poignée de Main Secrète)**.** *Une Poignée de Main Secrète est un protocole dans lequel deux utilisateurs u_i et u_j appartenant à un univers d'utilisateurs $\mathcal{U}$ s'authentifient comme possesseurs d'une propriété commune p_* appartenant à un univers de propriétés $\mathcal{P}$. L'authentification réussit si les deux utilisateurs possèdent les Titres de Preuve et Titres de Vérification légitimes pour la propriété p_*.*

La légitimité des Titres de Preuve et des Titres de Vérification dépend de la façon particulière dont ils sont générés. En effet, des politiques différentes de génération des Titres de Preuve et des Titres de Vérification jouent un rôle crucial sur le contrôle sur *"qui peut prouver la possession d'une propriété"* et sur *"qui peut vérifier la possession d'une propriété"*. Nous appellerons ces deux concepts *contrôle de preuve* et *contrôle de vérification.*

Par exemple, si une autorité de certification génère des Titres de Preuve et les donne seulement à des utilisateurs sélectionnés, il conserve le contrôle sur la capacité de prouver. La même chose se produit pour les Titres de Vérification.

A.3.1 Anonymat et *"Unlinkability"*

Dans cette section, nous étudions la quantité des informations divulguée à un observateur dans l'exécution d'un protocole pour Poignée de Main Secrète. D'abord, nous présentons quelques définitions, prises de [PH08].

Définition 4 (Anonymat). *L'Anonymat d'un utilisateur signifie que l'utilisateur n'est pas identifiable dans l'ensemble des utilisateurs.*

Définition 5 (*"Unlinkability"*). *"Unlinkability" de deux ou plusieurs points d'intérêt (comme, par exemple, sujets, messages, actions, ...) du point de vue d'un observateur signifie que dans le système l'observateur ne peut distinguer de manière suffisante si ces points d'intérêt sont liés ou non.*

D'abord, nous notons que l'Anonymat se rapporte toujours aux utilisateurs et à leur identifiants, alors que l' *"Unlinkability"* se rapporte à des objets d'intérêt général, qui ne sont pas nécessairement limités aux utilisateurs. Cependant, nous nous concentrons sur les utilisateurs. L'ensemble des utilisateurs est constitué par l'univers des utilisateurs $\mathcal{U}$ que nous avons introduit dans la Définition 1. Nous disons qu'un protocole de Poignée de Main Secrète garantit l'Anonymat si les identifiants des utilisateurs concernés ne sont pas révélées au sein de son exécution. L' *"Unlinkability"* des utilisateurs se rapporte plutôt à la capacité d'un observateur de relier le même utilisateur à plusieurs instances de Poignée de Main Secrète. Afin de rendre l'observateur aussi puissant que possible, nous supposons que l'observateur est l'un des deux utilisateurs du protocole. Nous pouvons donc dire qu'un protocole de Poignée de Main Secrète garantit l' *"Unlinkability"* des utilisateurs si – après l'exécution de deux instances distinctes de Poignée de Main Secrète– un observateur n'est pas capable de dire s'il est en interaction avec le même utilisateur ou avec deux utilisateurs différentes.

Nous allons maintenant tourner notre attention vers l' *"Unlinkability"* des propriétés. Suivant la même approche, nous disons qu'un protocole de Poignée de Main Secrète garantit l' *"Unlinkability"* des propriétés si – après l'exécution de deux instances distinctes de Poignée de Main Secrète– un observateur n'est pas capable de dire s'il est en interaction avec des utilisateurs détenant des Titres de Preuve pour la même propriété ou avec des utilisateurs détenant des Titres de Preuve pour des propriétés différentes; cela est nécessaire seulement dans le cas où la Poignée de Main Secrète a échoué, car en cas de succès, la vérification des propriétés est possible par définition.

A.3.2 Sur l'équité dans les protocoles pour Poignée de Main Secrète

Nous introduisons maintenant la notion d'équité selon la définition de Asokan [Aso98] et nous essayons de comprendre ses relations avec les protocoles pour la Poignées de Main Secrètes.

Définition 6 (Equité)**.** *Un protocole d'échange garantit l'équité si, à la fin du protocole, soit chaque utilisateur reçoit l'élément qu'il attend, soit aucun des deux utilisateurs ne reçoit d'information.*

Pour les protocoles pour la Poignée de Main Secrète, cette définition se traduit avec l'exigence que soit les utilisateurs apprennent qu'ils possèdent tous les deux une propriété commune, ou ils n'apprennent rien du tout. Comme nous l'avons vu, la preuve de connaissance de la clé calculée est ce qui permet aux utilisateurs d'apprendre si la poignée de main a réussi. Par conséquent, l'équité ne peut être atteinte que si les utilisateurs peuvent exécuter un protocole qui leur permet d'échanger – avec équité – les résultats d'une preuve de connaissance des deux valeurs, par exemple un protocole défi-réponse.

Malheureusement, un résultat de Pagnia et Gärtner [PG99] montre que l'équité dans les protocoles d'échange est impossible à atteindre sans un tiers de confiance. Les protocoles pour Poignée de Main Secrète peuvent cependant parvenir à une forme plus limitée d'équité. D'abord nous définissons le prédicat suivant:

$\mathfrak{P}$:= "*les deux participants au protocole de Poignée de Main Secrète possèdent la propriété objet de la poignée de main*"

Nous pouvons alors introduire la notion d'équité dans les protocoles pour Poignée de Main Secrète:

Définition 7 (Equité dans les protocoles pour Poignée de Main Secrète)**.** *Après une exécution complète ou incomplète d'un protocole pour Poignée de Main Secrète, soit au moins une partie apprend le prédicat $\mathfrak{P}$, ou personne n'apprend rien au delà de $\neg\mathfrak{P}$.*

avec $\neg\mathfrak{P}$ nous entendons la négation du prédicat $\mathfrak{P}$.

La définition 7 reconnaît le manque d'équité dans les protocoles pour Poignée de Main Secrète, mais elle limite seulement à des circonstances limitées l'avantage que l'un des deux utilisateurs, p_{adv}, peut avoir. En effet, p_{adv} peut apprendre $\mathfrak{P}$ seulement s'il possède la propriété objet de la poignée de main. p_{adv} ne peut sinon apprendre que $\neg\mathfrak{P}$.

A.3.3 Récapitulation des exigences

Nous récapitulons ici les exigences principales que nous avons identifiés dans cette section:

- *Séparabilité* traite de la possibilité d'accorder le droit de prouver séparément du droit de vérifier, et vice versa.
- *Contrôle de preuve* se rapporte à l'entité qui peut accorder le droit de prouver la possession d'une propriété.
- *Contrôle de vérification* se rapporte à l'entité qui peut accorder le droit de vérifier la possession d'une propriété.
- *Anonymat* nécessite qu'un utilisateur – après une exécution de Poignée de Main Secrète– ne soit pas identifiable parmi les autres utilisateurs.
- *"Unlinkability" des utilisateurs* exige qu'un observateur ne puisse pas lier deux exécutions de Poignée de Main Secrète à un seul utilisateur.
- *"Unlinkability" des propriétés* exige qu'un observateur ne puisse pas lier deux exécutions de Poignée de Main Secrète à une seule propriété.
- *Equité* est garantie par la Poignée de Main Secrète si, après l'exécution du protocole, un seul, ou les deux utilisateurs découvrent soit que les deux possèdent les Titres de Preuve correspondant aux Titres de Vérification de l'autre utilisateur, ou personne n'apprend rien.

A.4 Protocoles

La première partie de cette thèse contient des contributions en termes de nouveaux protocoles cryptographiques, blocs de construction génériques qui peuvent être utilisés pour répondre à plusieurs scénarios de Poignée de Main Secrète.

A.4.1 Poignée de Main Secrète avec Vérification Dynamique Contrôlée

Dans les protocoles de Poignée de Main Secrète qui permettent la Vérification Dynamique Contrôlée, les utilisateurs sont tenus de posséder Titres de Preuve et Titres de Vérification délivré par une autorité de certification de confiance afin d'être en mesure de prouver et de vérifier la possession d'une propriété donnée. Par conséquent, l'autorité de certification conserve le contrôle sur qui peut prouver quoi et qui peut vérifier quels Titres de Preuve. Cependant la vérification est dynamique, en ce qu'elle n'est pas limitée à une seule propriété, par opposition aux approches proposées dans [BDS+03; Ver05; SG08; Mea86; XY04].

Soulignons d'abord que ce nouveau protocole est d'une utilité pratique évidente. Par exemple, il répond aux besoins identifiés par le projet de l'UE R4EGov [Eur07]. Dans un des cas d'utilisation du projet, les forces de justice de l'UE coopèrent entre elles afin de résoudre des affaires pénales transfrontalières. Les réglementations de l'UE définissent des processus officiels qui doivent impérativement être suivis par les agents de police: en particulier, ces processus décrivent comment les institutions doivent coopérer sur chaque cas particulier. Au cours d'une collaboration, par exemple, un membre du *Ministère de la Défense* Français doit coopérer avec un membre du *Bundesnachrichtendienst*, le service de renseignement Allemand, pour enquêter sur un scandale interne. Les deux agents peuvent avoir besoin de se rencontrer secrètement, et de s'authentifier à la volée. Les deux sont certainement réticents à divulguer leur affiliation et leur but à d'autres personne.

Il est évident qu'ils ne peuvent pas utiliser des Poignées de Main Secrètes simples: en effet, ces dernières ne permettent d'effectuer des vérifications qu'au sein de la même organisation. Les poignées de main avec Vérification Dynamique ne fournissent pas non plus une solution adaptée au problème. La liberté de choisir la propriété à vérifier donne une marge trop large aux fonctionnaires, qui doivent plutôt se conformer strictement aux règlements de l'UE. En effet, ces fonctionnaires agissent au nom de l'Etat et du peuple: ils doivent suivre les règles et ne doivent pas faire de choix personnels.

Les acteurs d'un protocole pour Poignée de Main Secrète avec Vérification Dynamique Contrôlée sont représentés par des utilisateurs appartenants à un ensemble d'utilisateurs $\mathcal{U}$ et une une autorité de certification unique et fiable. Chaque utilisateur possède des propriétés appartenantes à un ensemble de propriétés $\mathcal{P}$. L'objectif des utilisateurs est d'effectuer des poignées de main, afin de prouver et de vérifier mutuellement la possession des propriétés. L'autorité de certification – au démarrage du système – exécute l'algorithme Setup pour générer les paramètres publics et privés du système. Dans un scénario avec Vérification Dynamique Contrôlée, les utilisateurs ont besoin de recevoir des Titres de Preuve et des Titres de Vérification de la part par de l'autorité de certification – qui est la seule entité habilitée à les générer – afin d'exécuter une Poignée de Main Secrète fructueuse. A cet effet, l'autorité de certification expose deux algorithmes, $\mathsf{Certify}$ et Grant, que les utilisateurs peuvent invoquer pour recevoir des

Titres de Preuve et des Titres de Vérification, respectivement. Enfin, deux utilisateurs peuvent exécuter le protocole Handshake; le protocole retourne succès si le premier utilisateur a un Titre de Preuve pour la propriété associée avec le Titre de Vérification du deuxième, et inversement si le deuxième utilisateur a un Titre de Preuve pour la propriété associée avec le Titre de Vérification du premier.

Une Poignée de Main Secrète avec Vérification Dynamique Contrôlée est définie par les algorithmes suivants:

- $\mathsf{Setup}(k) \rightarrow (\mathsf{param}, \mathsf{secret})$ est un algorithme non déterministe en temps polynomial exécuté par l'autorité de certification en prenant un paramètre de sécurité k en entrée et générant en sortie les paramètres publics param et les paramètres privés mathsf secret;

- $\mathsf{Certify}(u, p, \mathsf{secret}) \rightarrow (cred_p)$ est un algorithme non déterministe en temps polynomial exécuté – sur demande d'un utilisateur – par l'autorité de certification. L'entité de certification vérifie d'abord que l'utilisateur $u \in \mathcal{U}$ possède la propriété $p \in \mathcal{P}$ dont la possession sera prouvée lors de l'exécution du protocole; après une vérification réussie, l'entité de certification donne à u le Titre de Preuve $cred_p$. L'utilisateur peut vérifier l'exactitude du Titre de Preuve. Si la vérification réussit, l'utilisateur accepte le Titre de Preuve, annule sinon;

- $\mathsf{Grant}(u, p, \mathsf{secret}) \rightarrow (match_{u,p}, X_u)$ est un algorithme non déterministe en temps polynomial exécuté – sur demande d'un utilisateur – par l'autorité de certification. Tout d'abord l'entité de certification vérifie que – selon les politiques du système – l'utilisateur u est en droit de vérifier qu'un autre utilisateur possède une propriété $p \in \mathcal{P}$. Après vérification, l'entité de certification délivre le Titre de Vérification approprié formé par la couple $(match_{u,p}, X_u)$; l'utilisateur peut vérifier l'exactitude du Titre de Vérification. Si la vérification réussit, l'utilisateur accepte le Titre de Vérification, annule sinon;

- Handshake est un algorithme non déterministe en temps polynomial exécuté par deux utilisateurs, u_i et u_j; l'algorithme est composé de trois sous-algorithmes:

  - $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}) \rightarrow (n, \mathsf{state_{upd}})$ produit une valeur aléatoire n et met à jour l'état interne state;

- $\mathsf{Handshake.RandomizeCredentials}(\mathsf{param}, cred_p, \mathsf{state}, n) \to (SH, \mathsf{state_{upd}})$ reçoit en entrée les paramètres du système, le Titre de Preuve de l'utilisateur $cred_p$, la valeur aléatoire n reçue de l'autre partie et produit le message de la Poignée de Main Secrète SH; cet algorithme met également à jour l'état interne state;
- $\mathsf{Handshake.Match}(\mathsf{param}, SH, (match_{u,p}, X_u), \mathsf{state}) \to (K)$ reçoit en entrée les paramètres du système, le message de la Poignée de Main Secrète SH reçu de l'autre utilisateur, l'état local et le Titre de Vérification $(match_{u,p}, X_u)$ et produit la clé K;

L'algorithme fonctionne comme suit:

```
$u_i$                    : $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_i) \to (n_i, \mathsf{state}_{i,\mathsf{upd}})$
$u_i \longrightarrow u_j$ : $n_i$
$u_j$                    : $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_j) \to (n_j, \mathsf{state}_{j,\mathsf{upd}})$
$u_j \longrightarrow u_i$ : $n_j$
$u_j$                    : $\mathsf{Handshake.RandomizeCredentials}(cred_{p_1}, \mathsf{state}_j, n_i)$
                           $\to (SH_j, \mathsf{state}_{j,\mathsf{upd}})$
$u_j \longrightarrow u_i$ : $SH_j$
$u_i$                    : $\mathsf{Handshake.RandomizeCredentials}(cred_{p_2}, \mathsf{state}_i, n_j)$
                           $\to (SH_i, \mathsf{state}_{i,\mathsf{upd}})$
$u_i \longrightarrow u_j$ : $SH_i$
$u_i$                    : $\mathsf{Handshake.Match}(SH_j, (match_{u_i,p_3}, X_{u_i}), \mathsf{state}_i) \to (K_i)$
$u_j$                    : $\mathsf{Handshake.Match}(SH_i, (match_{u_j,p_4}, X_{u_j}), \mathsf{state}_j) \to (K_j)$
```

Figure A.2: L'algorithme Handshake exécuté par deux utilisateurs u_i et u_j.

Les deux clés K_i et K_j sont égales dans les conditions suivantes: $p_1 = p_3$ et $p_2 = p_4$. Autrement, une des deux clés est une valeur aléatoire avec une probabilité très élevé.

A.4.2 Révocation pour Poignées de Main Secrètes.

Le support pour la révocation des Titres de Preuve dans les protocoles pour Poignée de Main Secrète est une tâche difficile. D'un côté, Anonymat et *"Unlinkability"* des utilisateurs et des propriétés est une condition souhaitée. De l'autre côté, la plupart des solutions au problème de la révocation des Titres de Preuve exigent que les Titres de Preuve soient – d'une manière ou d'une autre – marqués, pour faire en sorte que les Titres de Preuve révoqués puissent être reconnus comme tels: cela viole ouvertement les exigences de *"Unlinkability"*.

Dans cette section nous définissons formellement un protocole de Poignée de Main Secrète avec support pour la révocation des Titres de Preuve. Les algorithmes qui composent le protocole sont les suivants:

- $\mathsf{Setup}(k) \rightarrow (\mathsf{param}, \mathsf{secret})$ est un algorithme non déterministe en temps polynomial exécutée par l'autorité de certification en prenant un paramètre de sécurité k en entrée et générant les paramètres publics param et les paramètres privés mathsf secret;

- $\mathsf{Certify}(u, p, \mathsf{secret}) \rightarrow (cred_{u,p}, x_{u,p})$ est un algorithme non déterministe en temps polynomial exécuté – sur demande d'un utilisateur – par l'autorité de certification. L'entité de certification vérifie d'abord que l'utilisateur $u \in \mathcal{U}$ possède la propriété $p \in \mathcal{P}$ dont la possession sera prouvée lors de l'exécution du protocole; après vérification, l'entité de certification donne à u le Titre de Preuve formé par la couple $(cred_{u,p}, x_{u,p})$. L'utilisateur peut vérifier l'exactitude du Titre de Preuve. Si la vérification réussit, l'utilisateur accepte le Titre de Preuve, annule sinon;

- $\mathsf{Grant}(u, p, \mathsf{secret}) \rightarrow (match_p)$ est un algorithme non déterministe en temps polynomial exécuté – sur demande d'un utilisateur – par l'autorité de certification. Tout d'abord l'entité de certification vérifie que – selon les politiques du système – l'utilisateur u est en droit de vérifier qu'un autre utilisateur possède une propriété $p \in \mathcal{P}$. Après vérification, l'entité de certification délivre le Titre de Vérification approprié $match_p$; l'utilisateur peut vérifier l'exactitude du Titre de Vérification. Si la vérification réussit, l'utilisateur accepte le Titre de Vérification, annule sinon;

- $\mathsf{Revoke}(u, p, \mathsf{secret}, \mathsf{rl}) \rightarrow (\mathsf{rl}_{\mathsf{upd}})$ est un algorithme non déterministe en temps polynomial exécuté par l'entité de certification lorsque le Titre de Vérification pour la propriété p détenu par l'utilisateur u doit être révoqué. L'entité de certification met à jour une liste de révocation publique rl, en ajoutant les informations de révocation appropriés;

- $\mathsf{Handshake}$ est un algorithme non déterministe en temps polynomial exécuté par deux utilisateurs, u_i et u_j; l'algorithme est composé de quatre sous-algorithmes:

  – $\mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}) \rightarrow (n, \mathsf{state}_{\mathsf{upd}})$ produit une valeur aléatoire n et met à jour l'état interne state;

  – $\mathsf{Handshake.RandomizeCredentials}(\mathsf{param}, (cred_{u,p_1}, x_{u,p_1}), \mathsf{state}, n) \rightarrow (SH,$ $\mathsf{state}_{\mathsf{upd}}$, $K_1)$ reçoit en entrée les paramètres du système, le Titre de Preuve de l'utilisateur, la valeur aléatoire n reçue de l'autre partie et produit le message de la Poignée de Main Secrète SH et la clé K_1 liée à la preuve de possession de la propriété p_1; cet algorithme met également à jour l'état interne state;

  – $\mathsf{Handshake.CheckRevoked}(\mathsf{param}, SH, \mathsf{rl}, \ldots) \rightarrow (b)$ reçoit en entrée le message de la Poignée de Main Secrète SH reçu de l'autre utilisateur, les paramètres du système, une version actualisée de la liste de révocation mathsf rl et des paramètres supplémentaires; l'algorithme génère une valeur booléenne à l'état vrai si le Titre de Preuve utilisé pour générer SH a été révoqué;

  – $\mathsf{Handshake.Match}(\mathsf{param}, SH, match_{p_2}, \mathsf{state}) \rightarrow (K_2)$ reçoit en entrée les paramètres du système, le message de la Poignée de Main Secrète SH reçu de l'autre utilisateur, l'état local et le Titre de Vérification $match_{u,p}$; l'utilisateur exécute tout d'abord le sous-algorithme $\mathsf{Handshake.CheckRevoked}$ pour vérifier si le Titre de Preuve de l'autre utilisateur a été révoqué; si $\mathsf{Handshake.CheckRevoked}$ donne une réponse négative, l'algorithme génère une clé K_2, liée à la vérification de possession de la propriété p_1;

  Supposons que deux utilisateurs u_i et u_j exécutent l'algorithme $\mathsf{Handshake}$; l'entrée de l'utilisateur u_i est son Titre de Preuve $(cred_{u_i,p_2}, x_{u_i,p_2})$ pour la propriété p_2 et un Titre de Vérification pour la propriété p_3, $match_{p_3}$; l'entrée de l'utilisateur u_j est son Titre de Preuve pour la propriété p_1, $(cred_{u_j,p_1}, x_{u_j,p_1})$ et un Titre de Vérification pour la propriété p_4, $match_{p_4}$. L'algorithme fonctionne comme décrit dans la figure A.3.

Les deux paires de clés $K_{i,1}$, $K_{j,2}$, et $K_{i,2}$, $K_{j,1}$ sont égales dans les conditions suivantes: $p_1 = p_3$ et $p_2 = p_4$. Autrement, une des deux clés est une valeur aléatoire avec une probabilité très élevé.

$$\begin{array}{lcll}
u_i & & : & \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_i) \rightarrow (n_i, \mathsf{state}_{i,\mathsf{upd}}) \\
u_i & \longrightarrow\ u_j & : & n_i \\
u_j & & : & \mathsf{Handshake.Init}(\mathsf{param}, \mathsf{state}_j) \rightarrow (n_j, \mathsf{state}_{j,\mathsf{upd}}) \\
u_j & \longrightarrow\ u_i & : & n_j \\
u_j & & : & \mathsf{Handshake.RandomizeCredentials}((cred_{u_j,p_1}, x_{u_j,p_1}), \mathsf{state}_j, n_i) \\
 & & & \rightarrow (SH_j, \mathsf{state}_{j,\mathsf{upd}}, K_{j,1}) \\
u_j & \longrightarrow\ u_i & : & SH_j \\
u_i & & : & \mathsf{Handshake.RandomizeCredentials}((cred_{u_i,p_2}, x_{u_i,p_2}), \mathsf{state}_i, n_j) \\
 & & & \rightarrow (SH_i, \mathsf{state}_{i,\mathsf{upd}}, K_{i,1}) \\
u_i & \longrightarrow\ u_j & : & SH_i \\
u_i & & : & \mathsf{Handshake.Match}(SH_j, match_{p_3}, \mathsf{state}_i) \rightarrow (K_{i,2}) \\
u_j & & : & \mathsf{Handshake.Match}(SH_i, match_{p_4}, \mathsf{state}_j) \rightarrow (K_{j,2})
\end{array}$$

Figure A.3: L'algorithme Handshake exécuté par deux utilisateurs u_i et u_j.

A.5 Cas d'utilisation

La deuxième partie de ce manuscrit présente des cas d'utilisation où les protocoles présentés dans la partie précédente servent comme solutions pratiques aux problèmes du monde réel.

A.5.1 Groupes secrets dans les réseaux sociaux

Dans cette thèse, nous présentons le premier système qui permet la création de Groupes d'intérêt Secrets (GIS) dans les réseaux sociaux en ligne; des GIS sont des groupes autogérés formés à l'extérieur du réseau social, autour des sujets secrets, sensibles ou privés. Les membres du group échangent des Titres de Preuve qui peuvent être utilisés à l'intérieur du réseau social pour authentifier les demandes d'amitié ou pour sécuriser les contenus générés par les utilisateurs.

A.5.1.1 Motivation

Les réseaux sociaux en ligne font partie des technologies de communication les plus populaires. Les plates-formes comme Facebook comptent maintenant des millions d'utilisateurs qui partagent quotidiennement leurs informations. Comme le contenu est hébergé très souvent par le fournisseur du réseau, les utilisateurs peuvent être profilées et le fournisseur du réseau peut leur offrir des annonces de publicité ciblées.

Un problème qui affecte particulièrement les utilisateurs des réseaux sociaux est l'usurpation d'identité [CT09; Pir08]. L'origine du problème est que dans de nombreux réseaux sociaux en ligne il y a presque aucune vérification que l'identité de la personne qui rejoint le réseau social est véritablement celle qu'elle prétend d'être. Cette lacune peut être combinée à une seconde: les utilisateurs des réseaux sociaux basent leur décision d'accepter une requête sur le nom de l'utilisateur, sur des photos et des fragments de texte; ces informations sont facilement accessibles ailleurs sur l'Internet. Il est donc relativement simple [BSBK09] pour une entité malveillante de mettre en place un faux profil sur un réseaux sociaux en ligne et puis de convaincre les autres utilisateurs d'accepter la demande d'amitié, afin d'avoir accès à leurs informations personnelles.

Pour améliorer la sécurité de ce processus, les utilisateurs pourraient être invités à fournir des Titres de Preuve lors de la demande d'amitié. Cependant, pour être significatifs, les Titres de Preuve ne peuvent pas être autogérées, mais ils doivent être générés et maintenus après vérification par un tiers: cette tâche est lourde et coûteuse et d'une part, il est irréaliste de s'attendre à ce qu'une entité centrale (le fournisseur de service de réseau social par exemple) s'en occupe gratuitement, et d'autre part, les utilisateurs n'accepteraient pas de payer pour ce service.

L'une des solutions envisageables est la suivante: les utilisateurs peuvent créer des groupes de confiance ad-hoc à *l'extérieur du réseau social*. Les membres du groupe reçoivent des Titres de Preuve et ils les utilisent lors du processus d'invitation dans le réseau social; le même phénomène se produit avec l'amitié, qui est formé en dehors du réseau social et utilise le réseau social pour favoriser la communication entre amis.

Une évolution naturelle des groupes susmentionnés sont les Groupes d'intérêt Secret (GIS), créés par les utilisateur avec une attention particulière à des sujets confidentiels ou tout simplement liés à la vie privée. En effet, les utilisateurs des réseaux sociaux en ligne échange aussi souvent du matériel personnel et sensible; en plus, les réseaux sociaux en ligne sont de plus en plus le théâtre de débat politique, religieux et sont souvent utilisés comme un moyen d'échanger des documents confidentiels qui ne peuvent pas passer par les voies officielles; cela a été le cas par exemple en Iran pendant les remous post-électorales [TW09].

L'objectif est donc la création d'un système qui prend en charge la formation et l'évolution des groupes d'intérêt Secret. Grâce à notre système, les utilisateurs sont en mesure de gérer l'entrée et la sortie des membres, de révoquer ou accorder le privilège d'administration aux membres et d'accorder des Titres de Preuve que les membres peuvent utiliser pour assurer leur relation avec les autres membres dans un réseau social. La conception du système pose deux défis: le système doit (i) répondre aux exigences de groupes ad hoc, il ne doit pas exiger une entité centralisée, mais fonctionner de manière distribuée et (ii) il doit être possible de le mettre en œuvre dans un réseau social en ligne réel, avec toutes les contraintes de celle-ci, par exemple, la rareté des liaisons directes entre les utilisateurs, les modes de communication centralisée sur les serveurs du réseau social et le fait que les utilisateurs ne sont pas nécessairement en ligne au même temps.

Le système que nous avons *conçu et développé*, soutient les exigences techniques susmentionnées ainsi que les exigences de sécurité élevées typiques des groupes secrets. Ces groupes sont secrets dans le sens que l'appartenance à un GIS est une information sensible que les utilisateurs sont réticents à exposer publiquement. Un GIS peut être par exemple autour des intérêts religieux, politiques tels que les utilisateurs sont très intéressés d'interagir avec d'autres utilisateurs qui partagent le même intérêt, mais très réticents à admettre publiquement qu'ils appartiennent à un groupe d'intérêt. Un GIS peut être aussi centrée autour de thèmes moins secrètes, qui représentent quand même la vie privée des sujets. Le système proposé répond donc au cas le plus complexe des groupes secrets, mais peut être appliqué à de simples groupes d'utilisateurs aussi.

A.5.2 Authentification des partenaires d'une chaîne logistique basée sur RFID

L'utilisation des radio-étiquettes (RFID) dans les chaînes d'approvisionnement a une valeur ajoutée indiscutable du point de vue productivité, mais soulève un certain nombre de nouveaux défis liés à la sécurité. L'un d'eux est l'authentification des paires de participants à une chaîne d'approvisionnement qui ont possédé le même élément, mais qui n'ont par ailleurs jamais communiqué auparavant. La situation est encore plus complexe si nous imaginons que les participants à la chaîne d'approvisionnement sont des concurrents.

Dans cette thèse, nous présentons un nouveau protocole cryptographique qui résout ce problème. Dans notre solution, les utilisateurs échangent des radio-étiquettes au cours du cycle d'une chaîne d'approvisionnement et, si les deux entités ont possédé la même étiquette, ils peuvent convenir d'une clé secrète commune qu'ils peuvent utiliser pour protéger leurs échanges d'informations. Aucune entité malveillante ne peut régénérer la bonne clé secrète, parce cet action sera soit traçable ou impliquerait la perte d'une clé secrète, ce qui constitue une incitation forte à garder le secret sur les informations d'authentification.

A.5.2.1 Motivation

La radio-identification est une technologie moderne qui supporte le suivi et le traçage des articles marqués dans les chaînes d'approvisionnement. Chaque élément est équipé d'une radio-étiquettes qui porte un identifiant unique. Cette radio-étiquette peut être lu via communication sur fréquence radio et plusieurs radio-étiquettes peuvent être lues à la fois.

Il y a des radio-étiquettes actives et passives. Les étiquettes actives ont leur propre alimentation; les étiquettes passives fonctionnent uniquement grâce à la puissance du signal émis par le lecteur. Le lecteur est un appareil spécial qui peut interagir avec les étiquettes et lire les identifiants stockés dans leur mémoire. Des étiquettes plus complexes peuvent stocker plus d'informations dans la mémoire et même effectuer des simples opérations cryptographiques telles que le hachage.

Une application importante de la radio-identification est la gestion de la chaîne d'approvisionnement [WB08; BWL06; AM05]. Chaque objet peut être suivi au moyen de son numéro d'identification unique. Le potentiel de cette technologie s'accroît si tous les partenaires de la chaîne logistique partagent les données liées à la chaîne d'approvisionnement. Ces informations contiennent généralement au moins ce qui suit:

$$\langle organization, identifier, timestamp \rangle$$

Ce triplet est généralement enrichi par des informations supplémentaires, tel que l'identifiant du lecteur, le type d'événement (réception, expédition, déballage, etc), et des champs supplémentaires en fonction du type d'événement.

Les entreprises sont intéressés au partage des informations liées à des événements pour de nombreuses raisons: premièrement, un consommateur peut être intéressé à connaître le pedigree du produit. Une entreprise peut avoir besoin de rappeler les produits défectueux et peut-être intéressé de connaître la liste des détaillants qui les ont vendues. Les résultats des projets de recherche collaboratifs tels que Bridge ou Ko-RFID [Kon] montrent que les entreprises sont réticentes à révéler leurs participations à une chaîne d'approvisionnement pour un certain nombre de raisons, comme par exemple la peur de l'espionnage ou parce qu'il pourrait être embarrassant de révéler la participation à la production d'un produit avec des graves problèmes, qui a été retiré du marché. En conséquence, les informations sont stockées en toute sécurité par chaque partenaire et volontairement divulguées seulement après une authentification.

Afin de partager les données liées aux événements, les entreprises se connectent à un réseau mondial, actuellement en cours de normalisation par le consortium EPC-global. Ce réseau contient un service de découverte, qui stocke des pointeurs vers toutes les entreprises qui disposent de données d'événements pour chaque radio-étiquettes spécifique. Afin de récupérer toutes les informations sur une étiquette, les entreprises doivent contacter le service de découverte; ce dernier renvoie alors la liste de toutes les entreprises ayant manipulé le produit. Ensuite, les entreprises peuvent communiquer individuellement et échanger les données d'événement.

Le principal défi de ce système est que d'un côté les entreprises voudraient partager cette information de manière à faciliter leurs activités, de l'autre côté, ces informations sont strictement confidentielles et les entreprises hésitent à faire confiance les uns les autres à cause de la possibilité d'espionnage par un concurrent [SS08], réalisé par exemple en récupérant les données d'événement sur les éléments de la chaîne d'approvisionnement d'un concurrent.

Imaginez deux entreprises qui n'ont jamais communiqué ensemble et qui ont été mises en contact via un service de découverte; ces deux entreprises doivent s'authentifier mutuellement: la seule chose qu'elles ont jamais eu en commun, c'est qu'elles ont toutes les deux été en possession de la même radio-étiquettes à un moment donné. Ces entreprises ont besoin de prouver l'une l'autre qu'elles ont possédé la même étiquette.

Il y a un certain nombre d'attaques qui pourraient se produire dans ce scénario:

3. Un imposteur peut demander des informations sur des radio-étiquettes qu'il n'a jamais possédé, par exemple afin de suivre la chaîne d'approvisionnement de son concurrent.

4. Une société malveillante pourrait fournir des informations sur des radio-étiquettes qu'elle n'a jamais possédé, par exemple afin de dissimuler l'origine des produits contrefaits.

Une solution simple à ce problème est de stocker un secret partagé sur l'étiquette, pour faire en sorte que tous ceux qui l'ont possédé puissent utiliser ce secret pour sécuriser les communications ultérieures.

Malheureusement, cette solution présente de nombreux inconvénients. Tout d'abord, il n'y a pas obligation pour qu'une entreprise qui possède le secret partagé, le garde pour secret, car la révélation du secret ne peut pas lui être attribuée. Deuxièmement, la radio-étiquettes peut être lu de manière frauduleuse par un étranger qui n'a pas fait légitimement partie de la chaîne d'approvisionnement.

Par conséquent, afin d'avoir une solution sécurisée pour ce problème, nous avons besoin de développer un système plus complexe qu'un simple mécanisme basé sur des secrets partagés.

A.5.2.2 Détail de la Solution

Dans cette thèse, nous présentons un nouveau protocole qui résout le problème ci-dessus. La logique veut que les informations stockées sur l'étiquette sont liée à une identité. Seul le détenteur de la clé privée liée à l'identité peut effectivement prouver la possession de la radio-étiquette. Les informations stockées sur l'étiquette sont mises à jour chaque fois que la radio-étiquette change de propriétaire; la mise à jour est effectuée grâce à l'aide d'un tiers de confiance. La participation du tiers de confiance permet de suivre chaque objet tout au long de la chaîne d'approvisionnement.

Notre solution permet de surmonter les inconvénients de la solution simple présentée ci-dessus. En effet, si une entité malveillante demande des informations restreintes pour une étiquette, il peut être compromis par le tiers de confiance. D'autre part, l'usurpation d'identité n'est possible que si un des utilisateurs légitimes révèle ses informations secrètes. Un partenaire permettant à une entité malveillante de s'authentifier doit donc décider soit d'être traçable ou de révéler la clé secrète, renonçant à toutes ses informations sensibles.

Une propriété intéressante de notre solution, est que les radio-étiquettes peuvent être efficacement utilisées avec des primitives cryptographiques complexes: en effet, les étiquettes agissent simplement comme transporteurs d'enveloppes cryptographiques, qui sont ensuite utilisés pour exécuter des protocoles de sécurité complexes. Notre technologie fonctionne avec les étiquettes plus simples spécifiées par la norme EPCglobal (étiquettes de classe 1). La seule exigence est que les étiquettes doivent être capable de stocker la quantité minimale d'information nécessaire pour effectuer les opérations cryptographiques. Les étiquettes n'ont pas nécessairement besoin d'être réinscriptible: elles peuvent tout simplement être remplacées par de nouvelles étiquettes contenant les nouvelles informations: la baisse des prix du matériel peut justifier ce choix.