Implications of Position in Cryptography

THÈSE N^{O} 8981 (2018)

PRÉSENTÉE LE 23 NOVEMBRE 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE DE SÉCURITÉ ET DE CRYPTOGRAPHIE
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Handan KILINÇ ALPER

acceptée sur proposition du jury:

Prof. A. Lenstra, président du jury
Prof. S. Vaudenay, directeur de thèse
Prof. G. Avoine, rapporteur
Prof. F. D. Garcia, rapporteur
Prof. M. Payer, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

To my mother, my father and my brother

Abstract

In our daily lives, people or devices frequently need to learn their location for many reasons as some services depend on the absolute location or the proximity. The outcomes of positioning systems can have critical effects e.g., on military, emergency. Thus, the security of these systems is quite important. In this thesis, we concentrate on many security aspects of position in cryptography.

The first part of this thesis focuses on the theory of distance bounding. A distance bounding protocol is a two-party authentication protocol between a prover and a verifier which considers the distance of the prover as a part of his/her credential. It aims to defeat threats by malicious provers who try to convince that they are closer to the verifier or adversaries which seek to impersonate a far-away prover. In this direction, we first study the optimal security bounds that a distance bounding protocol can achieve. We consider the optimal security bounds when we add some random delays in the distance computation and let the prover involve distance computation. Then, we focus on solving the efficiency problem of public-key distance bounding because the public-key cryptography requires much more computations than the symmetric-key cryptography. We construct two generic protocols (one without privacy, one with) which require fewer computations on the prover side compared to the existing protocols while keeping the highest security level. Then, we describe a new security model involving a tamper-resistant hardware. This model is called the secure hardware model (SHM). We define an all-in-one security model which covers all the threats of distance bounding and an appropriate privacy notion for SHM.

The second part of this thesis is to fill the gap between the distance bounding and its real-world applications. We first consider contactless access control. We define an integrated security and privacy model for access control using distance bounding (DB) to defeat relay attacks. We show how a secure DB protocol can be converted to a secure contactless access control protocol. Regarding privacy (i.e., keeping anonymity in a strong sense to an active adversary), we show that the conversion does not always preserve privacy, but it is possible to study it on a case by case basis. Then, we consider contactless payment systems. We design an adversarial model and define formally the contactless payment security against malicious cards and malicious terminals. Accordingly, we design a contactless payment protocol and show its security in our security model.

The last part of this thesis focuses on positioning. We consider two problems related to positioning systems: localization and proof of location. In localization, a user aims to find its position by using a wireless network. In proof of location, a user wants to prove his/her position e.g., to have access to a system or authorize itself. We first formally define the problem of localization and construct a formal security model. We describe algorithms and protocols for localization which are secure in our model. Proof of location has been considered formally by Chandran et al. in CRYPTO 2009 and it was proved that achieving security is not possible in the vanilla model. By integrating the localization and the secure hardware model, we obtain a model where we can achieve proof of location.

Keywords: Distance Bounding, Localization, Secure Positioning, Access Control, Contactless Payment, Proof of Location, Relay Attack, Position-based cryptography, RFID, NFC

Résumé

Dans la vie quotidienne, les gens ou les appareils connectés ont souvent besoin de connaître leur position pour de multiples raisons, car certains services dépendent de l'emplacement absolu ou de la proximité . Les résultats des systèmes de positionnement peuvent avoir des effets critiques, par exemple, pour l'armée ou les services d'urgence. La sécurité de ces systèmes est donc importante. Dans cette thèse, nous nous concentrons sur de nombreux aspects de la sécurité liée à la position en cryptographie.

La première partie de cette thèse se concentre sur la théorie de protocoles délimiteurs de distance. Un protocole délimiteur de distance est un protocole d'authentification entre un prouveur et un vérificateur qui considère la distance du prouveur comme faisant une partie de ses identifiants. Dans ce sens, nous étudions d'abord les limites de la sécurité optimale qu'un protocole délimiteur de distance peut atteindre. Ensuite, nous nous concentrons sur la résolution du problème d'efficacité des protocoles délimiteurs de distance par clé publique car la cryptographie à clé publique nécessite beaucoup plus de calculs que la cryptographie à clé symétrique. Nous construisons deux protocoles génériques qui nécessitent moins de calculs du côté du prouveur que dans les protocoles existants, tout en conservant le plus haut niveau de sécurité. Ensuite, nous décrivons un nouveau modèle de sécurité impliquant un matériel inviolable. Ce modèle est appelé le modèle matériel sécurisé. Nous définissons un modèle de sécurité tout-en-un qui couvre toutes les menaces de délimiteur de distance et une notion de confidentialité appropriée pour le modèle matériel sécurisé.

La deuxième partie comble le fossé entre le protocole délimiteur de distance et ses applications dans le monde réel. Nous considérons d'abord le contrôle d'accès sans contact. Nous définissons un modèle intégré de sécurité et de confidentialité pour le contrôle d'accès en utilisant le délimiteur de distance pour vaincre les attaques de relais. Nous montrons comment un protocole délimiteur de distance sécurisé peut être converti en un protocole sécurisé de contrôle d'accès sans contact. En ce qui concerne la vie sphère privée, nous montrons que la conversion ne la préserve pas toujours, mais qu'il est possible de l'étudier au cas par cas. Ensuite, nous considérons les systèmes de paiement sans contact. Nous concevons un modèle de sécurité pour le paiement sans contact contre les cartes et les terminaux malveillants. En conséquence, nous concevons un protocole de paiement sans contact et montrons sa sécurité dans notre modèle.

La dernière partie porte sur le positionnement. Nous considérons deux problèmes liés aux systèmes de positionnement : la localisation et la preuve de localisation. Pour la localisation, un utilisateur vise à trouver sa position en utilisant un réseau sans fil. Pour la preuve de localisation, un utilisateur veut prouver sa position, par exemple, avoir accès à un système ou s'authentifier. Nous définissons d'abord formellement le problème de la localisation et construisons un modèle de sécurité formel. Nous décrivons des algorithmes et des protocoles de localisation sécurisés dans notre . La preuve de localisation a été prouvé que la réalisation de la sécurité n'est pas possible dans le modèle standard. En intégrant la localisation et le modèle matériel sécurisé, nous obtenons un modèle où nous pouvons obtenir une preuve de localisation.

Acknowledgments

When I started my PhD studies, I knew that it will be a long and challenging journey. Now, when I looked back, I see that I was not wrong but these two adjectives are not enough to describe my journey. My PhD journey was sometimes monotonous, sometimes pleased, often stressful, rarely disappointed, always informative and more. Although the difficulties I faced, I feel happy, satisfied and extremely proud in the end.

First, my gratitude and appreciation go to my supervisor Prof. Serge Vaudenay for accepting me into his lab and allowing me to work with him. Serge, you taught me a lot during my PhD: how to do a proper research, how to deal with the problems, how to be critical against my work and others' work. You were always patient against my mistakes, my questions, my writing skills. Thanks for all your patience, reading my papers many times, spending your time to improve my research. In addition, thank you for your friendly and welcoming approach. Indeed, I feel very lucky to have a supervisor like you. It was an honor to work with you.

Secondly, I would like to thank the members of my PhD thesis committee. I thank Prof. Arjen Lenstra my committee president. I also express my gratitude towards Prof. Gildas Avoine and Prof. Flavio Garcia who accepted to be my external jury examiners. Finally, I wish to thank Prof. Mathias Payer for accepting to be my internal jury examiner. Thank you all for your valuable comments and remarks.

I would also like to express my gratitude towards Prof. Alptekin Küpçü who was my supervisor in my master studies. Thanks to his advises and encouragements, I decided to do PhD. He always showed his trust and motivated me to do research. Thank you Alptekin, for teaching me cryptography, introducing me the world of research, your motivational talks and your support.

Having an ideal lab environment is essential when doing a PhD which can be daunting at times. Therefore, my next thanks go to the present and former LASEC members: Prof. Serge Vaudenay, Dr Philippe Oechslin, Martine Corval, Dr Petr Sušil, Dr Alexandre Duc, Dr Sonia Mihaela Bogos, Dr Damian Vizár, Fatih Balli, Gwangbae Choi, Khashayar Barooti, Dr Subhadeep Banik, Dr Fatma Betül Durak, Dr Iraklis Leontiadis.

I especially would like to thank my ex-officemate Damian Vizár with whom I shared the office INF 240 almost three years. I was very lucky to have such a smart, kind and positive person as an oficemate. He was always available to help and give support. Thank you Damian for all your help, informative and funny chats, the artistic drawing on my board and all nuts that you shared with me. I also would like to thank Sonia Bogos who is the former member of LASEC. Thank you Sonia, for your friendly attitude, your positive energy, your help and support. I would like to kindly thank Martine Corval, our secretary. Martine was always very helpful and always welcoming us with a smile in her office. Thank you Martine for all the help to solve our problems.

When I first arrived at EPFL, I had almost no friend in Switzerland. At the end of four years, luckily, I made so many good friends. I first would like to thank my friends Agata Mosinska, Konrad Domanski, Elie Najm with whom I shared the stressful fellowship year. Thank you all for all the funny moments in the middle room and having lunches with a lot of laughter. Secondly, I would like to thank my Turkish fellows Berker Ağır, Buğra Tekin, Egeyar Bağcıoğlu, Can Baltacı, Dilan Çelebi, Erhan Gündoğdu, Erkin Arslan, Merve Gürel, İlke Ilgaz, Tuğba Kılıç, Eray Sabancılar, Işık Sırmatel and Onur Yürüten. Thank you Buğra, Can, Erhan, Erkin for being such a great lunch team. Thank you İlke, Dilan, Işık, and Eray for being such an easygoing executive board of TURQUIA 1912. Thank you Onur for nice dinners at your home and your relaxing talks when I feel bad. Thank you Egeyar for introducing me EPFL and being such a good friend. Thank you Merve for your support in my stressful moments, relaxing girls nights, being such a generous, caring and loving friend, and also thank you for your efforts to be with me in my wedding.

Berker was my one of the first people that I met when I arrive at EPFL. From this time to now, he always helped me with every kind of problem that I faced. Thank you Berker for being a good listener, your motivational talks, being such a good and reliable friend, made me watch so many good movies which I prejudged.

Dilan is more than a friend for me. She is like a sister. We shared so many moments together during these years. She is the warmest, the most affectionate and most sympathetic person that I have ever meet. Thank you Dilan for making our flat like a home, standing for my stressful moments, your efforts to make me feel better, funny dish-washings, courageous skiing, and crazy dancings.

I would like to also thank my dear friend Buket Yüksel outside EPFL. Although the distance, she was always with me during these years. Thank you Buket for relaxing phone calls, making me laugh with your funny stories, and visiting me here.

My next thanks are for my husband Cem Alper. When I started my PhD, he was in my life as a friend and then he became my family. There are millions of things to thank him. I am sure my last four years would be very difficult for me without him. His support and trust were the biggest motivation for me. Thank you Cem for making my life easier, calming me every time, finding solutions to my problems, listening to my research related works with complete attention, and being such a great husband.

Last but not least, I want to thank my parents and my brother for their love and unconditional help. My father Süleyman Kılınç and my mother Aynur Kılınç always supported me for my decisions, believed in me and showed their love. Thank you mom and dad for being the best parents ever. My brother Orhun Kılınç always helped me and provided solutions to my problems during my PhD. Thank you Orhun for your encouragements and being such a supportive brother. These sentences are not enough to express my gratitude towards my parents and my brother. I am very lucky to have them.

Table of Contents

Chapter 1

Introduction

The importance of position in cryptography shows up out of necessity with the technological developments of contactless systems and intelligent systems related to navigation. The typical objectives related to position can be categorized as: *localization* which is determining current location of a device, *secure positioning* which is proving one's own position to an authority, and *proximity proofs* which is proving an upper bound on the distance from a device. In this thesis, we define and construct systems that provide security with respect to each of these objectives.

Proof of Proximity: In the proof of proximity, there exist two parties where one party (the prover) proves its distance from the other party and the other party (the verifier) verifies the proof. All contactless authentication protocols such as contactless payment (e.g. NFC), access control in a building, remote keyless system (e.g. car keys) require proximity proofs to defeat the relay attack. In the relay attack, a malicious party intends to impersonate a party (e.g., a smart card) during the authentication process by extending the transmission range of the signals. In Figure 1.1, we illustrate an example attack scenario. Using the proof of proximity, the party who verifies this authentication process (the verifier) can check if the party (the prover) is out of range (i.e., far-away from the verifier) during the authentication process. Beyond the relay attack, the proof of proximity is necessary for the following attacks as well:

Mafia Fraud (MiM) [Des88]: A man-in-the-middle (MiM) adversary between a verifier and a far-away honest prover makes the verifier accept the access of the prover as in Figure 1.1. In a MiM attack, the adversary can do more than just relaying (e.g., replace messages).

There are also threats by malicious provers:

Distance Fraud (DF): A malicious far-away prover tries to prove that he is close enough to the verifier to make the verifier accept.

Distance Hijacking (DH) [CRSČ12]: A far-away malicious prover takes advantage of some honest and active provers who are close to the verifier to make the verifier grant privileges to the far-away prover.

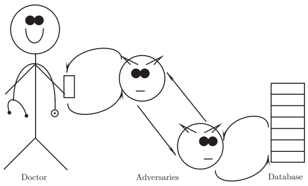

Figure 1.1 – The adversaries retrieve information from a hospital database by relaying the messages between the database reader and the doctor's card. Here, the doctor is far-away from the database. Arrows show receiving or sending messages.

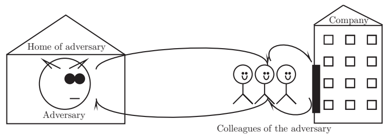

Figure 1.2 – The adversary who is an employee of the company accesses to the door of the company which shows that he arrived at work although he is at home. Here, the adversary can use one of his colleagues who is just next to the door. Arrows show receiving or sending messages.

Terrorist fraud (TF) [Des88]: A far-away malicious prover, with the help of an adversary, tries to make the verifier accept the access of the prover.

Figure 1.2 shows an attack scenario for DH. The same scenario is valid for DF and TF as well.

The most promising solution for the proof of proximity is distance bounding (DB). Distance bounding was first introduced by Brands and Chaum [BC93]. It was inspired from Beth and Desmedt [BD91] who suggested to use time of flight of messages to detect relay attacks. In DB, the verifier generally verifies if the prover is in a range by computing the round-trip time of sending a challenge and receiving a response (they are generally 1 or 2 bit(s)) in many rounds. In the end, if too many rounds have too long round trip times or too many incorrect responses, the verifier rejects because it implies that the prover is out of range. DB's security is based on the fact that the communication speed cannot be faster than the speed of light.

Localization and Secure Positioning: Localization and secure positioning have the same setup consisting of many bases whose physical location is known and a user. However, the parties in this setup have completely different objectives. In localization setup, the user does not know his/her location and wants to obtain his/her physical location by getting help from bases. The adversarial intention in a localization setup is to make the user obtain a wrong location. In secure positioning setup, each party knows his/her own location but the bases want to be convinced there is a party at the claimed location by the user. The aim of an adversary in secure positioning is proving that there is a user at a location even though no one is there.

We can see in many real world examples that the security of localization and secure positioning is important. For example, consider the autonomous cars which can navigate without human involvement by using their sensors and algorithms. One important issue regarding the autonomous cars is the security of their systems. What if have these cars had insecure localization systems that can easily be fooled or mislead? In this case, they may not arrive where they are supposed to arrive, they may enter an area that they

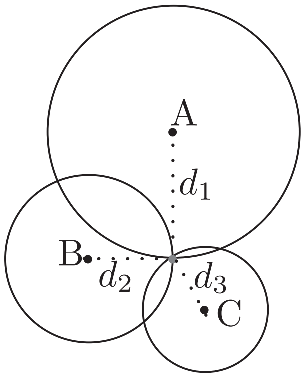

Figure 1.3 – Trilateration Method. A, B and C are the known locations, d_1,d_2 and d_3 are the distances from the unknown location (red point) to A, B and C, respectively.

normally should not enter such as a field, a schoolyard, a hospital area and possibly damage them.

Another example is related to secure positioning. The electronic tagging is used for people who have been sentenced for electronic monitoring. These systems should be secure so that they cannot be fooled by the prisoners or their accomplices who want to change their place.

The notions localization and secure positioning have completely different objectives but they use the similar algorithms such as triangulation and trilateration. In this thesis, we propose secure protocols and algorithms for these objectives using trilateration. One of the reason for choosing this technique is its accuracy as it is not affected by environmental changes as triangulation [TI10] algorithms. The other reason is that it requires proof of proximity which is also a part of this thesis. The trilateration algorithm, as it can be seen in 1.3, outputs the intersection of circles whose radius is the distance between the known location and unknown one. In general, in the localization setup, the user computes his/her distances to locations of bases and obtain its location using the trilateration algorithm and in the secure-positioning setup, the bases computes the distance of the user from their own location in order to see that the output of the trilateration algorithm is the same as the claimed one.

We see that the attacks of proof of proximity (MiM, DF, DH and TF) also affect the security of localization and secure positioning protocols using trilateration method. For example, consider the localization setup. A MiM adversary can execute a MiM attack during the distance computation and make the user believe that it is closer to the location of a base. So, the trilateration algorithm outputs a wrong location. Let's also consider the secure positioning problem. The user claims that (s)he is at a location and execute one of the attacks DF, DH or TF and makes some bases compute wrong distances so that the trilateration algorithm outputs the claimed location.

Consequently, the implications of position in cryptography can be seen in three notions: localization, secure positioning and proximity proof. Securely realizing them is not possible without secure distance computation. The existing solution for this is distance bounding. Therefore, in this thesis, we first focus on distance bounding and then suggest solutions for localization and secure positioning.

Outline of the Thesis

This thesis consists of three parts, where the first part is related to the theory of distance bounding, the second part is about integration of distance bounding in specific applications, and the last part is about localization problems. Before starting all these parts, we first give in Chapter 2 the existing security model for distance bounding and some other useful security definitions which are used in the following chapters.

Part I gives original contributions to the theory of distance bounding. In Chapter 3, we study how introducing random delays and having the prover to measure time can improve optimal protocols. Then, in Chapter 4, we concentrate on constructing efficient and secure public-key distance bounding protocols. In this direction, we construct the most efficient public-key DB protocols comparing with the other protocols with the same level or lower level of security. In Chapter 5, we study a distance bounding protocol based on secure hardware to make full TF security is achievable.

In Part II, we aim to construct application specific security models and protocols on top of distance bounding. Therefore, in Chapter 6, we develop a security model for contactless access control and show how to achieve security and privacy with using only distance bounding protocols. Similarly, in Chapter 7, we provide a security model for contactless payment systems. We also analyze the security of the existing contactless payment protocol that we use in our daily lives.

Part III includes solutions for the problems related to positioning. In Chapter 8, we formalize the localization problem and provide a security definition. We also propose localization protocols and prove their security formally. Using the security model of localization and the distance bounding based on a secure hardware model from Chapter 5, we develop a model called proof of location for secure positioning in Chapter 9.

Personal Bibliography

Below is the list of publications that were published during this thesis. Entries in bold are included in this thesis.

[1] Handan Kılınç and Alptekin Küpçü. Optimally efficient multi-party fair exchange and fair secure multi-party computation. *In: Nyberg K. (eds) Topics in Cryptology — CT-RSA 2015. CT-RSA 2015. Lecture Notes in Computer Science, vol 9048. Springer, Cham*

[2] **Handan Kılınç and Serge Vaudenay. Optimal Proximity Proofs Revisited. *In: Malkin T., Kolesnikov V., Lewko A., Polychronakis M. (eds) Applied Cryptography and Network Security. ACNS 2015. Lecture Notes in Computer Science, vol 9092. Springer, Cham***

[3] Handan Kılınç and Alptekin Küpçü. Efficiently Making Secure Two-Party Computation Fair. *In: Grossklags J., Preneel B. (eds) Financial Cryptography and Data Security. FC 2016. Lecture Notes in Computer Science, vol 9603. Springer, Berlin, Heidelberg*

[4] **Handan Kılınç and Serge Vaudenay. Efficient Public-Key Distance Bounding Protocol.** ***In: Cheon J., Takagi T. (eds) Advances in Cryptology – ASIACRYPT 2016. ASIACRYPT 2016. Lecture Notes in Computer Science, vol 10032. Springer, Berlin, Heidelberg***

[5] **Handan Kılınç and Serge Vaudenay. Contactless Access Control Based on Distance Bounding.** ***In: Nguyen P., Zhou J. (eds) Information Security. ISC 2017. Lecture Notes in Computer Science, vol 10599. Springer, Cham***

[6] **Handan Kılınç and Serge Vaudenay. Formal Analysis of Distance Bounding with Secure Hardware.** ***In: Preneel B., Vercauteren F. (eds) Applied Cryptography and Network Security. ACNS 2018. Lecture Notes in Computer Science, vol 10892. Springer, Cham***

[7] **Handan Kılınç and Serge Vaudenay. Secure Contactless Payment.** ***In: Susilo W., Yang G. (eds) Information Security and Privacy. ACISP 2018. Lecture Notes in Computer Science, vol 10946. Springer, Cham***

Chapter 2

Preliminaries

2.1 Notations

We let sk and pk denote a secret key and public key, respectively. We show the owner by using a subscript: e.g., $(\mathsf{sk}_P, \mathsf{pk}_P)$ is the secret/public pair of a party P. We denote by s a symmetric key.

We denote the location of a party I by loc_I. We use $d(I,J)$ as a metric which gives the distance between locations loc_I and loc_J. I is called *close* to J, if $d(I,J) \leq B$ and *far* from J, if $d(I,J) > B$ where B is a common distance bound.

An encryption scheme is a tuple $(\mathsf{Enc}, \mathsf{Dec})$ where Enc is the encryption algorithm and Dec is the decryption algorithm. Similarly, a signature scheme is a tuple $(\mathsf{Sign}, \mathsf{Verify})$ where Sign is the signing algorithm and Verify is the verification algorithm. The subscript used on these algorithm specifies the key: e.g. $\mathsf{Enc}_{\mathsf{sk}}(M)$ is encryption of message M with the key sk.

We let Γ_i denote a game and p_i denote the probability that an adversary succeeds Γ_i where $i \in \{0,1,2,...\}$. We also give some special names to some games (e.g., $\mathsf{Game}(.)$). If the game outputs 1, we say the game is won: e.g. $\mathsf{Game}(.) = 1$. If it is 0, the game is not won.

$\mathsf{Pr}[E]$ is used to define the probability of an event E.

A function $\mathsf{negl}(x) : \mathbb{N} \to \mathbb{R}$ is *negligible*, if for every positive polynomial $\mathsf{poly}(.)$, there exists a positive integer N such that for all $x > N$, $|\mathsf{negl}(x)| < \frac{1}{\mathsf{poly}(x)}$.

ℓ is used as a security parameter.

2.2 Distance Bounding

We first define formally a distance-bounding protocol. We have two types of it: public-key distance bounding [BC93, HPO13, GOR14a, Vau15c, Vau15a, Vau15d, KV16, ABG+17] and symmetric distance bounding [BMV13a, BMV15, BMV13b, Vau13, FO13, BV14].

Definition 2.1 (Public-key DB Protocol [Vau15c])**.** *A public key distance-bounding protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple* $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$*. Here,* $\mathcal{K}_P$ *and* $\mathcal{K}_V$ *are the key generation algorithms of* P *and* V*, respectively. The output of* $\mathcal{K}_P$ *is a secret/public key pair* $(\mathsf{sk}_P, \mathsf{pk}_P)$ *and similarly the output of* $\mathcal{K}_V$ *is a secret/public key pair* $(\mathsf{sk}_V, \mathsf{pk}_V)$*.* P *is the proving algorithm,* V *is the verifying algorithm where the inputs of* P *and* V *are from* $\mathcal{K}_P$ *and* $\mathcal{K}_V$*.* B *is the distance bound.* $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ *and* $V(\mathsf{sk}_V, \mathsf{pk}_V)$ *interact with each other. At the end of the protocol,* $V(\mathsf{sk}_V, \mathsf{pk}_V)$ *outputs a final message* $\mathsf{Out}_V \in \{0,1\}$ *and has* pk_P *as a private output. If* $\mathsf{Out}_V = 1$*, then* V *accepts. If* $\mathsf{Out}_V = 0$*, then* V *rejects.*

Correctness: A public-key DB protocol is correct if and only if under the honest execution, whenever a verifier instance $\mathcal{V}$ and a close (to $\mathcal{V}$) prover instance P run the protocol, then $\mathcal{V}$ always outputs $\mathsf{Out}_V = 1$ and pk_P.

Remark that Definition 2.1 combines identification with DB: pk_P is not an input of the algorithm V, but it is an output. So, V learns the identity of P during the protocol.

Now, we give the definition of symmetric DB. It is very similar to the definition of public-key DB.

Definition 2.2 (Symmetric DB Protocol [BV14])**.** *A symmetric distance-bounding protocol is a two-party PPT protocol and it consists of a tuple* $(\mathcal{K}, V, P, B)$*. Here,* $\mathcal{K}$ *is the key generation algorithm,* P *is the proving algorithm and* V *is the verifying algorithm. The inputs of* P *and* V *is the output* s *of* $\mathcal{K}$*.* B *is the distance bound.* $P(s)$ *and* $V(s)$ *interact with each other. At the end of the protocol,* $V(s)$ *outputs a final message* $\mathsf{Out}_V \in \{0,1\}$*. If* $\mathsf{Out}_V = 1$*, then* V *accepts. If* $\mathsf{Out}_V = 0$*, then* V *rejects.*

Correctness: A symmetric DB protocol is correct if and only if under honest execution, whenever a verifier instance $\mathcal{V}$ and a close (to $\mathcal{V}$) prover instance P run the protocol, then $\mathcal{V}$ always outputs $\mathsf{Out}_V = 1$.

The DB protocols in this thesis follow the common structure defined by Boureanu and Vaudenay [BV14]. The DB protocols in common structure have some phases, where one of them corresponds to the phase of distance computation (challenge phase). Identifying DB protocols based on this structure makes the protocol descriptions, the security definitions and the security proofs easy to explain.

Definition 2.3 (Common Structure [BV14])**.** *A DB protocol* $(\mathcal{K}, V, P, B)$ *(or* $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$*) based on the common structure with parameters* $(n, \mathsf{num}_c, \mathsf{num}_r)$ *consists of three phases which are 'initialization phase', 'verification phase' and between them 'challenge phase'. Here,* num_c *is the cardinality of the challenge set,* num_r *is the cardinality of the response set and* n *is the number of rounds in the challenge phase. In the challenge phase, the verifier sends challenges to the prover and receives responses from the prover. Here, the verifier measures the elapsed time between sending the challenge and receiving the response in each round. Time is not used otherwise and provers do not measure time. If the elapsed time is less than what needed for information to travel in a distance* $2B$*, the response is called on time.*

We now give another structure which is a variation of the common structure.

Definition 2.4 (Canonical Structure [Vau15c]). *A symmetric DB protocol* $(\mathcal{K},V,P,B)$ *follows the canonical structure, if there exist an initialization/challenge/verification phases,* P *does not use* s *during the initialization phase,* $\mathcal{V}$ *does not use* s *at all except for computing the final* Out_V*, and the verification phase is not interactive.*

In real life, the channel is noisy. So, the challenges or responses do not always arrive correctly [HK05, CHKM06, KAK+08, KAK+09, SP07], even though no adversary exists. This means that in a noisy environment, the condition which is the number of correct responses has to be equal to n can cause false negatives. To overcome this, we give a definition τ-completeness.

Definition 2.5 (τ-complete [BV14]). *A DB protocol* $(\mathcal{K},V,P,B)$ *based on the common structure with parameters* $(n,\tau,\mathsf{num}_c,\mathsf{num}_r)$ *is called* τ*-complete when the algorithm* V *outputs* $\mathsf{Out}_V = 1$ *if and only if at least* τ*-rounds have correct and on-time responses in the challenge phase.*

When we set up τ, we should consider the noise tolerance of the channel. Here, we assume that each round in a challenge phase is corrupted with probability p_{noise}. Therefore, the probability of τ-completeness in the case of a close-by honest prover is $\mathsf{Tail}(n,\tau,1-p_{\mathsf{noise}})$ [BV14, BMV13b, BMV15] where:

$$\mathsf{Tail}(n,\tau,\rho) = \sum_{i=\tau}^{n} \binom{n}{i} \rho^i (1-\rho)^{n-i} \tag{2.1}$$

Accordingly, the probability of failure is negligible in terms of n when $\frac{\tau}{n} < c < 1 - p_{\mathsf{noise}}$ for some constant c due to the Chernoff-Hoeffding bound [Che52, Hoe63].

We note that we assume $\tau = n$ in the next chapters, except Chapter 3, for the sake of clarity. Depending on p_{noise}, this assumption can change for all protocols given in this thesis.

2.2.1 Security of Distance Bounding

The security formalism in DB started by Avoine et al. [ABK+09, ABK+11]. Then, the first complete model was introduced by Dürholz et al. [DFKO11] where the threat models are defined according to the number of tainted time critical phase. The SKI model by Boureanu et al. [BMV13a, BMV13b, BMV15] is another formal model which includes a clear communication model between parties in DB. The last model BV model [BV14] by Boureanu and Vaudenay is a more natural multi-party security model. In this thesis, we use the security and the communication model (BMV model) by Boureanu et al. [BMV13a, BMV15, BMV13b]. The details of the model are as follows:

Adversarial and Communication Model: In the DB model, we have parties called provers (P), a verifier (V) and other actors. Each party has instances and each instance I has its own location loc_I. The communication between two instances I and J takes time which depends on the distance between I and J. The parties have common notions of time, time-unit, and measurable distance. The communication follows the laws of physics, e.g., communication cannot be faster than the speed of light (c). Namely, a message sent by I at time t can arrive J at time $t' \geq t + \frac{d(I,J)}{c}$. By abuse of notion, we thus measure time with a distance unit ($t' \geq t + d(I,J)$).

The verifier is always honest and its instances always run the specified algorithm. However, provers can be malicious. An instance of a malicious prover runs an arbitrary algorithm. The honest instances cannot be run in parallel.

Without loss of generality, we say that the other actors are malicious. They may run any algorithm. We assume that actors (adversaries) have very special hardware which can intercept a message and change its destination without any delay. Similarly, they can update a message and send it to any destination with this hardware without any delay. So, if an instance I sends a message at time t_1, and the adversary reads or updates the message at time t_2 and the adversary sends it to an instance J at time t_3, then the arrival time of the message to J is still lower bounded by $t_1 + d(I,J)$. However, the adversary could send a message to J before seeing the one from I. Then, the adversary blocks the delivery of the correct message. In this case, the message would arrive to J before time $t_1 + d(I,J)$ but would be independent from the message sent by I as proven in Fundamental Lemma (Lemma 2.7).

Adversaries can activate honest prover instances with some special signals. The special signal $\mathsf{Activate}(P)$ activates the only activatable instance of P. After receiving this signal, further activation signals are ignored by this instance. An instance can be terminated by one of the following signals: $\mathsf{Terminate}(P)$ and $\mathsf{Move}(P, loc')$. $\mathsf{Terminate}(P)$ terminates the instance execution, but it remains "active". The special signal $\mathsf{Move}(P, loc')$ orders to terminate and move the prover to loc'. It means that the instance becomes inactive and that only one unused instance of P at location loc' can be activated. The terminated instance sends a special signal Go which, when received by this unused instance at location loc', will make it activatable (Go signals cannot be sent by malicious parties; they are here only to enforce that a prover cannot move faster than a signal propagation). After, it may receive another $\mathsf{Activate}(P)$ as a new instance of the same prover at location loc'. **These signals model the provers being at a single location and moving (as influenced by the adversary) to run other instances.** Besides, it models that instances of the same prover cannot be run concurrently.

Definition 2.6 (DB experiment)**.** *An experiment* exp *for a distance-bounding protocol with the tuple* $(\mathcal{K}, P, V, B)$ *or* $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$ *is a setting* $(\boldsymbol{P}, \boldsymbol{V}, \boldsymbol{A})$ *with several PPT instances of participants, at some locations.*

We denote by $\mathsf{exp}(\mathcal{V})$ a distinguished experiment where we fix a verifier instance $\mathcal{V}$ called the distinguished verifier.

Lemma 2.7 (Fundamental Lemma [BV14, Vau15d]). *Consider an experiment where a party V broadcasts a message c at time t to all other parties and waits for a response r. The parties who are further than B are in a set* Far *and the others are in a set called* Close. *We let E be an event in which V receives r no later than $t+2B$. We denote the view of a party I just before seeing c (before $t+d(V,I)$) by* View_I. *A message sent by U is called independent if it is sent before $t+d(V,I)$. If E happens, then there exists an algorithm* $\mathsf{Alg}(\mathsf{View}_{\mathsf{Close}}, c, \mathsf{Other}) \to r$. *Here,* $\mathsf{View}_{\mathsf{Close}}$ *is the set of all* View_I *where* $I \in \mathsf{Close}$ *and* Other *is all independent messages from parties* $I \in \mathsf{Far}$.

The Fundamental Lemma states that a close-party cannot get online help from a far-away party to respond correctly and on time.

Now, we explain the security definitions for distance fraud (DF), man-in-the-middle (MiM) and distance hijacking (DH) from [Vau15c].

DF security captures security against a malicious and far away prover which does not get any help from anyone else.

Definition 2.8 (Distance-Fraud Security in Public-key DB [Vau15c]). *The game begins by running the key setup algorithm $\mathcal{K}_V(1^\ell)$ which outputs* $(\mathsf{sk}_V, \mathsf{pk}_V)$. *The game includes instances of the verifier including the distinguished one $\mathcal{V}$ and instances of an adversary. Given* pk_V, *the adversary (malicious prover) generates its key* $(\mathsf{sk}_P, \mathsf{pk}_P)$ *with an arbitrary key setup algorithm* $\mathcal{K}^*(\mathsf{pk}_V)$ *(instead of $\mathcal{K}_P$). The adversary wins if $\mathcal{V}$ outputs* $\mathsf{Out}_V = 1$, $\mathsf{POut}_V = \mathsf{pk}_P$, *and there is no participant close to $\mathcal{V}$. A public-key DB protocol is DF-secure, if for any such game, the adversary wins with negligible probability in the security parameter ℓ.*

The symmetric DB version is defined in a very similar way.

Definition 2.9 (Distance-Fraud Security in Symmetric DB [Vau15c]). *The game includes instances of the verifier including the distinguished one $\mathcal{V}$ and instances of an adversary. The adversary (malicious prover) generates its key s with an arbitrary key setup algorithm $\mathcal{K}^*$ (instead of $\mathcal{K}$). The adversary wins if $\mathcal{V}$ outputs* $\mathsf{Out}_V = 1$ *and there is no participant close to $\mathcal{V}$. A symmetric DB protocol is DF-secure, if for any such game, the adversary wins with negligible probability in the security parameter ℓ.*

As we can see, a malicious and far-away prover can setup his key maliciously.

The other security definition to protect against malicious prover is distance hijacking. Here, the malicious and far-away prover can get advantage of an honest prover without the honest prover being aware of it. We use the DH-definition specified for the distance-bounding protocol in the common structure (Definition 2.3) as it is easier to have security proofs.

Definition 2.10 (Distance-Hijacking Security in Public-key DB [Vau15c]). *The game consists of instances of the verifier, instances of a malicious prover* P, *and also instances of an honest prover* P'. *A DB protocol* $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ *having an initialization, a challenge and a verification phases is DH-secure if for all PPT algorithms $\mathcal{K}_P^*$ and $\mathcal{A}$, the probability of* P *to win the following game is negligible in the security parameter ℓ.*

- *The game generates secret/public keys of the honest prover and the verifier:* $\mathcal{K}_V(1^\ell) \to (\mathsf{sk}_V, \mathsf{pk}_V)$, $\mathcal{K}_P(\ell) \to (\mathsf{sk}_{P'}, \mathsf{pk}_{P'})$.
- *Malicious prover* P *runs* $\mathcal{K}_P^*(\mathsf{pk}_{P'}, \mathsf{pk}_V) \to (\mathsf{sk}_P, \mathsf{pk}_P)$ *and if* $\mathsf{pk}_P = \mathsf{pk}_{P'}$*, the game aborts. Instances of* P *run an algorithm* $\mathcal{A}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, \mathsf{pk}_{P'})$.
- P *can interact with instances of the honest prover and the verifier during the initialization phase and verification phases concurrently.*
- *One instance of the honest prover and one instance of the verifier* $\mathcal{V}$ *continue interacting with each other in their challenge phase and* P *remains passive even though it sees the exchanged messages.*

The adversary wins if $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* pk_P.

Definition 2.11 (Distance-Hijacking Security in Symmetric DB [Vau15c])**.** *The game consists of instances of the verifier, instances of a malicious prover* P*, and also instances of an honest prover* P'*. A DB protocol* $(\mathcal{K}, V, P, B)$ *having an initialization, a challenge and a verification phases is DH-secure if for all PPT algorithms* $\mathcal{K}^*$ *and* $\mathcal{A}$*, the probability of* P *to win the following game is negligible in the security parameter* ℓ.

- *The game generates the secret key of honest prover* P': $\mathcal{K}(1^\ell) \to s'$.
- *Malicious prover* P *runs* $\mathcal{K}^* \to s$ *to generate the its secret key. Instances of* P *runs* $\mathcal{A}(s)$.
- P *can interact with instances of the honest prover and the verifier during the initialization phase and verification phases concurrently.*
- *One instance of the honest prover and one instance of the verifier* $\mathcal{V}$ *continue interacting with each other in their challenge phase and* P *remains passive even though it sees the exchanged messages.*

The adversary wins if $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$.

There exists also a weaker DH-security definition called *one-time DH (OT-DH)*. It can be defined as in Definition 2.10 and 2.11 by changing the game setting with only one instance of the verifier and of the honest prover.

Now, we give the security definitions to achieve security against non-prover adversaries. These definitions also cover impersonation fraud.

Definition 2.12 (MiM Security in Public-key DB [Vau15c])**.** *The game begins by running the key setup algorithms* $\mathcal{K}_V(1^\ell)$ *and* $\mathcal{K}_P(1^\ell)$ *which output* $(\mathsf{sk}_V, \mathsf{pk}_V)$ *and* $(\mathsf{sk}_P, \mathsf{pk}_P)$*, respectively. The adversary receives* pk_V *and* pk_P*. The game consists of instances of the verifier including the distinguished one* $\mathcal{V}$*, instances of a prover* P *and instances of the adversary. The adversary wins if* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$, pk_P*, and there exists no instance of prover* P *close to* $\mathcal{V}$*. A public-key DB protocol is MiM-secure if for any such game, the probability of an adversary to win is negligible in the security parameter* ℓ.

$\underline{V}(s)$	**initialization phase**	$\underline{P}\ (s)$
pick $m \in \{0,1\}^{2n}$	$\xrightarrow{\quad m \quad}$	$a = s \oplus m$
	challenge phase	
	for $i = 1$ to n	
pick $c_i \in \{0,1\}$, start timer_i	$\xrightarrow{\quad c_i \quad}$	$r_i = a_{2i+c_i-1}$
stop timer_i	$\xleftarrow{\quad r_i \quad}$	
	verification phase	
$a = s \oplus m$, check $\mathsf{timer}_i \leq 2B, r_i = a_{2i+c_i-1}$	$\xrightarrow{\quad \mathsf{Out}_V \quad}$	

Figure 2.1 – OTDB

Definition 2.13 (MiM Security in Symmetric DB [Vau15c])**.** *The game begins by running the key setup algorithm $\mathcal{K}(1^\ell)$ which outputs s. The game consists of instances of the verifier including the distinguished one $\mathcal{V}$, instances of a prover* P *and instances of the adversary. The adversary wins if $\mathcal{V}$ outputs* $\mathsf{Out}_V = 1$ *and there exists no instance of prover* P *close to $\mathcal{V}$. A symmetric DB protocol is MiM-secure if for any such game, the probability of an adversary to win is negligible in the security parameter ℓ.*

There exists also a weaker MiM-security definition called *one-time MiM (OT-MiM)*. It can be defined as in Definition 2.12 and 2.13 by changing the game setting with only one instance of the verifier and of the prover.

In the next chapters, we give our DB protocols which are secure against MiM, DF and DH adversaries. Some of these protocols are constructed on top of one-time secure protocols. Therefore, we give an example of one-time secure symmetric DB protocol OTDB by Vaudenay [Vau15c] which is a symmetric DB adapted from Hancke-Kuhn protocol [HK05]. The OTDB protocol follows the canonical structure (See Definition 2.4), only requires one xor operation before the challenge phase on the prover side and it is DF, OT-MiM and OT-DH secure [Vau15c].

OTDB (Figure 2.1): During the initialization phase, the verifier picks a $2n$-bit long string m, where n is the number of rounds in the challenge phase, and sends it to the prover. Then, the prover obtains $a = s \oplus m$. In the challenge phase, in each round i, the verifier picks a challenge $c_i \in \{0,1\}$ and sends it to the prover. The prover responds each challenge c_i of the verifier with a_{2i+c_i-1}. In each round i, the verifier computes the round trip time (timer_i) of sending the challenge and receiving the response. At the end, the verifier checks whether all responses are correct (i.e., $c_i = a_{2i+c_i-1}$) and all responses are arrived on time (i.e., $\mathsf{timer}_i \leq 2B$).

2.2.2 Privacy in DB

In some applications of distance bounding such as access control, the privacy of the prover becomes important. There are different levels of privacy for DB depending on the power of the adversary. Vaudenay [Vau07] identified these levels such as strong, weak and narrow. We give below the HPVP-privacy game [HPVP11] and then describe different levels of privacy.

Definition 2.14 (HPVP Privacy Game [HPVP11]). *The privacy game for a public-key distance bounding* $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ *with a bit* $b \in \{0,1\}$ *is the following: The game runs the key setup algorithms* $\mathcal{K}_P(1^\ell)$ *for a number* t *of provers and* $\mathcal{K}_V(1^\ell)$ *for the verifier. Then, it lets the adversary* $\mathcal{A}$ *play the game* $\mathsf{Priv}^{O}_{b,\mathcal{A}}(\ell)$ *with the following oracles which are in* O*:*

- `CreateP`$(ID) \to P_i$: *It creates a new prover identity of* ID *and returns its identifier* P_i.
- `Launch`$() \to \pi$: *It launches a new protocol with a verifier instance* V_j *and returns the session identifier* π.
- `Corrupt`(P_i) : *It returns the current state of* P_i*. Current state includes all values in* P_i*'s current memory. It does not include volatile memory.*
- `DrawP`$(P_i, P_j) \to vtag$: *It draws either* P_i *(if* $b = 0$*) or draws* P_j *(if* $b = 1$*) and returns the virtual tag reference* $vtag$*. If one of the provers had already been an input of* `DrawP` *which outputted* $vtag'$ *and* $vtag'$ *has not been released yet, then it outputs* $\emptyset$.
- `Free`$(vtag)$: *It releases* $vtag$ *which means* $vtag$ *can no longer be accessed.*
- `SendP`$(vtag, m) \to m'$: *It sends the message* m *to the drawn prover and returns the response* m' *of the prover. If* $vtag$ *was not drawn or was released, nothing happens.*
- `SendV`$(\pi, m) \to m'$: *It sends the message* m *to the verifier in the session* π *and returns the response* m' *of the verifier. If* π *was not launched, nothing happens.*
- `Result`$(\pi) \to b'$: *It returns a bit that shows if the session* π *is accepted by the verifier (i.e the message* Out_V*).*

In the end of the game, the adversary outputs a bit b''*. If* $b'' = b$*, then* $\mathcal{A}$ *wins. Otherwise, it loses.*

A DB protocol is strong private *if for all PPT adversaries* $\mathcal{A}$*, the advantage of winning the privacy game is negligible in the security parameter* ℓ*, where the advantage is defined as follows:*

$$\mathsf{Adv}(\mathsf{Priv}^{O}_{b,\mathcal{A}}(\ell)) = |\Pr[\mathsf{Priv}^{O}_{0,\mathcal{A}}(\ell) = 1] - \Pr[\mathsf{Priv}^{O}_{1,\mathcal{A}}(\ell) = 1]|$$

We distinguish strong and weak privacy [Vau07]. The weak privacy game does not include any '`Corrupt`' oracle. The other kind of classification is *wide* and *narrow* private.

Wide privacy game is allowing to use the '`Result`' oracle while the narrow privacy game does not. In this thesis, we implicitly consider wide privacy by making Out_V a public message, which means we always obtain this bit without using '`Result`' oracle.

2.3 Other Security Definitions

In this section, we give known security definitions and assumptions which we use in the security proofs of our results in this thesis.

Definition 2.15 (Pseudo-Random Function (PRF))**.** *Let $f_s : \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\ell)}$ be a function where s is chosen uniformly at random from ℓ-bit strings and* poly *is a polynomial and let $\mathcal{F}$ be a set of functions from $\{0,1\}^*$ to $\{0,1\}^{\mathsf{poly}(\ell)}$. We say f_s is a PRF, if for all PPT distinguishers $\mathcal{D}$, the advantage as defined below is at most negligible in ℓ:*

$$\mathsf{Adv}(PRF) = |\mathsf{Pr}[\mathcal{D}^{f_s(.)}(1^\ell) = 1] - \mathsf{Pr}[\mathcal{D}^{F(.)}(1^\ell) = 1]|.$$

where F is chosen uniformly at random from $\mathcal{F}$.

We give another security definition called circular PRF related to PRF. The notion of circular-keying in pseudorandom functions introduced by Boureanu et al. [BMV15, BMV13b]. Circular-keying PRF has an extra assumption to the PRF $(f_s)_{s\in GF(q)^\ell}$ to handle reuse of a fixed s outside of a PRF instance f_s.

Definition 2.16 (Circular PRF [BV14])**.** *Let be s, n_1, n_2 and q some parameters. An oracle $O_{\tilde{s},F}$ is defined as $O_{\tilde{s},F}(y, L, A, B) = A \cdot L(\tilde{s}) + B \cdot F(y)$, using dot product over $GF(q)$, given $L : \{0,1\}^s \to GF(q)^{n_1}$, $F : \{0,1\}^* \to GF(q)^{n_2}$, $A \in GF(q)^{n_1}$, $B \in GF(q)^{n_2}$ and $\tilde{s} \in GF(q)^\ell$. We assume that L is taken from a set of functions with polynomially bounded representation. Let $(f_s)_{s\in GF(q)^\ell}$ be a family of functions from $\{0,1\}^*$ to $\{0,1\}^{n_2}$. The family f is a circular-PRF, if for all PPT distinguishers $\mathcal{D}$, the advantage as defined below is at most negligible in ℓ:*

$$\mathsf{Adv}(C\text{-}PRF) = |\mathsf{Pr}[\mathcal{D}^{f_s(.)}(1^\ell) = 1] - \mathsf{Pr}[\mathcal{D}^{O_{\tilde{s},F}(.,.,.,.)}(1^\ell) = 1]|$$

Additionally, we require two conditions on the list of queries:

- *for any pair of queries (y, L, A, B) and (y', L', A', B'), if $y = y'$, then $L = L'$.*
- *for any y, if (y, L, A_i, B_i), $i = 1, 2, ..., t$ is the list of queries using this value y, then for all $\lambda_1, \lambda_2, ..., \lambda_t \in GF(q)$*

  $$\sum_{i=1}^{t} \lambda_i B_i = 0 \Rightarrow \sum_{i=1}^{t} \lambda_i A_i = 0$$

  over the $GF(q)$-vector space $GF(q)^{n_2}$ and $GF(q)^{n_1}$.

Now, we define the Gap Diffie Hellman (GDH) problem which is basically the Computational Diffie-Helman (CDH) problem with the access of a Decisional Diffie-Hellman (DDH) oracle.

Definition 2.17 (Gap Diffie-Hellman (GDH) [OP01])**.** *Let $\mathbb{G}$ be a cyclic group of order $p \in \{0,1\}^\ell$ and $g \in \mathbb{G}$ be a generator. We have the following problems:*

- *CDH: Given $g, X, Y \in \mathbb{G}$ compute $Z = g^{log_g X . log_g Y}$.*
- *DDH: Given $g, X, Y, Z \in \mathbb{G}$, decide if $Z = g^{log_g X . log_g Y}$ or $Z = g^r$ where $r \in \mathbb{Z}_p$ is a random element.*

The GDH problem is solving the CDH given (g, X, Y) with the help of a DDH oracle which answers whether a given quadruple is a Diffie-Hellman quadruple.

We say that GDH problem is hard *in group $\mathbb{G}$, if for all PPT adversaries, the probability of solving the GDH problem is negligible in ℓ.*

We give two security notions related to public-key encryption schemes. The first one is chosen-ciphertext attack security (IND-CCA) and the other one is the key-privacy under chosen-plaintext attack (IK-CPA).

Definition 2.18 (IND-CCA)**.** *The IND-CCA game with bit $b \in \{0,1\}$ for the public-key encryption scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ as follows: The IND-CCA game generates a secret/public key $(\mathsf{sk}, \mathsf{pk})$ from the key generation algorithm $\mathsf{Gen}_E(1^\ell)$. Then, the game $\mathsf{CCA}_{b,\mathcal{A}}^{\mathsf{Dec}(.)}(\ell)$ starts:*

- *$\mathcal{A}$ receives pk.*
- *The adversary has access to the decryption oracle $\mathsf{Dec}(.)$ before receiving the challenge. Dec decrypts given ciphertext with sk.*
- *The adversary sends two messages m_0, m_1 and the game sends $c_b = Enc_{\mathsf{pk}}(m_b)$ as a challenge.*
- *After sending the challenge, the adversary still has an access to the decryption oracle $\mathsf{Dec}(.)$ but it is not allowed to query the challenge ciphertext c_b.*
- *The game ends when $\mathcal{A}$ outputs a bit b'. It wins if $b = b'$.*

The public-key encryption scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ is IND-CCA secure, if for all PPT adversaries $\mathcal{A}$, the following advantage is negligible in ℓ.

$$\mathsf{Adv}(\mathsf{CCA}_{b,\mathcal{A}}^{\mathsf{Dec}}(\ell)) = |\Pr[\mathsf{CCA}_{0,\mathcal{A}}^{\mathsf{Dec}(.)}(\ell) = 1] - \Pr[\mathsf{CCA}_{1,\mathcal{A}}^{\mathsf{Dec}(.)}(\ell) = 1]|$$

Definition 2.19 (IK-CPA [BBDP01])**.** *The IK-CPA game with bit $b \in \{0,1\}$ for the public-key encryption scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ as follows: The IK-CPA game generates two secret/public key pairs $(\mathsf{sk}_0, \mathsf{pk}_0)$ and $(\mathsf{sk}_1, \mathsf{pk}_1)$ from the key generation algorithm $\mathsf{Gen}_E(1^\ell)$. Then, the game $\mathsf{IK\text{-}CPA}_{b,\mathcal{A}}(\ell)$ starts:*

- $\mathcal{A}$ *receives* pk_0 *and* pk_1.
- *The adversary sends a message* m *and the game sends* $c = Enc_{\mathsf{pk}_b}(m)$ *as a challenge.*
- *The game ends when* $\mathcal{A}$ *outputs a bit* b'*. It wins if* $b = b'$.

The public-key encryption scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ *is IK-CPA secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible in* ℓ.

$$\mathsf{Adv}(\mathsf{IK\text{-}CPA}_{b,\mathcal{A}}(\ell)) = |\mathsf{Pr}[\mathsf{IK\text{-}CPA}_{0,\mathcal{A}}(\ell) = 1] - \mathsf{Pr}[\mathsf{IK\text{-}CPA}_{1,\mathcal{A}}(\ell) = 1]|$$

We also define the security of existential-forgery chosen-message attack (EF-CMA) for a signature scheme $(\mathsf{Gen}_S, \mathsf{Sign}, \mathsf{Verify})$.

Definition 2.20 (EF-CMA)**.** *The EF-CMA game for the signature scheme* $(\mathsf{Gen}_S, \mathsf{Sign}, \mathsf{Verify})$ *is as follows: The EF-CMA game generates the secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$ *from the key generation algorithm* $\mathsf{Gen}_S(1^\ell)$*. Then, the game* $\mathsf{EF\text{-}CMA}_{\mathcal{A}}^{\mathsf{Sign}(.)}(\ell)$ *starts:*

- $\mathcal{A}$ *receives* pk.
- *The adversary has access to the signing oracle* $\mathsf{Sign}(.)$*.* Sign *signs a given message with* sk *and adds the message to a list* $\mathcal{L}$.
- *The game ends when* $\mathcal{A}$ *outputs a message and a signature pair* (m, σ)*. It wins if* $\mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$ *outputs valid and* $m \notin \mathcal{L}$.

The signature scheme $(\mathsf{Gen}_S, \mathsf{Sign}, \mathsf{Verify})$ *is EF-CMA secure, if for all PPT adversaries* $\mathcal{A}$*,* $\mathsf{Pr}[\mathsf{EF\text{-}CMA}_{\mathcal{A}}^{\mathsf{Sign}(.)}(\ell) = 1]$ *is negligible in* ℓ.

Part I

Distance Bounding

Chapter 3

Optimal Proximity Proofs Revisited

Boureanu and Vaudenay [BV14] revise the threat models of distance bounding and define a structure called *common structure* (Definition 2.3). They further analyze the optimal security that we can achieve in this structure and proposed DBopt (with concrete instances DB1, DB2, DB3) which reaches the optimal security bounds.

In this chapter, we define three more new structures: when the prover can register the time of a challenge (Sync Structure), when the verifier randomizes the sending time of the challenge (Rand Structure), and the combined structure (SyncRand Structure). Then, we identify the optimal security bounds against DF and MiM in our new structures and improve the bounds showed by Boureanu and Vaudenay for the common structure. Finally, we adapt the DBopt protocol according to our new structures and we get three new distance bounding protocols. We compare the performance of the adapted protocols with instances of DBopt and we see that we have a better efficiency in terms of number of rounds. For instance, we can reduce the number of rounds in DB2 from 123 down to 5 with the same security.

The content of this chapter was published in ACNS15 [KV15].

3.1 Our Contribution

In a nutshell, we list our contributions as follows:

- We define three new structures for distance bounding protocols. The first structure is *Sync Structure* where the prover stores each challenge's arrival time. The second structure is *Rand Structure* where the verifier sends each challenge in an arbitrary time. Finally, the last structure is *SyncRand Structure* which is the combination of the first two structures.

- We show the optimal security bounds for each new structure. Compared to the common structure [BV14], we obtain better security bounds as the additional properties on these structures decrease the efficiency of adversary's attack strategies.

- We adapt the DBopt protocol [BV14] with our new structures and obtain new protocols DBoptSync, DBoptSyncRand and DBoptRand. We prove their security against DF and MiM (DH and TF resistance are unchanged compared to DBopt in the common structure). We reach the optimal security bounds for DF and MF for all of them in their respective structure.

- We analyze the performance of adapted DBopt protocols and conclude that we have a better efficiency than DBopt in the common structure in terms of number of rounds.

We note that in this chapter, the definitions and results are the same for public-key DB and symmetric DB but we give our results for symmetric DB.

Structure of the Chapter: In Section 3.2, we revise the optimal-security bounds for MiM and DF by Boureanu and Vaudenay [BV14]. Then, in Section 3.3, we define our new structures and show the optimal security bounds in them. In Section 3.4, we adapt DBopt to these new structures. We conclude this chapter with a performance analysis in terms of number of rounds in Section 3.5 and with a conclusion in Section 3.6.

3.2 Revised Security Definitions

In this section, we give optimal security bounds in a DB protocol following the common structure by Boureanu and Vaudenay [BV14].

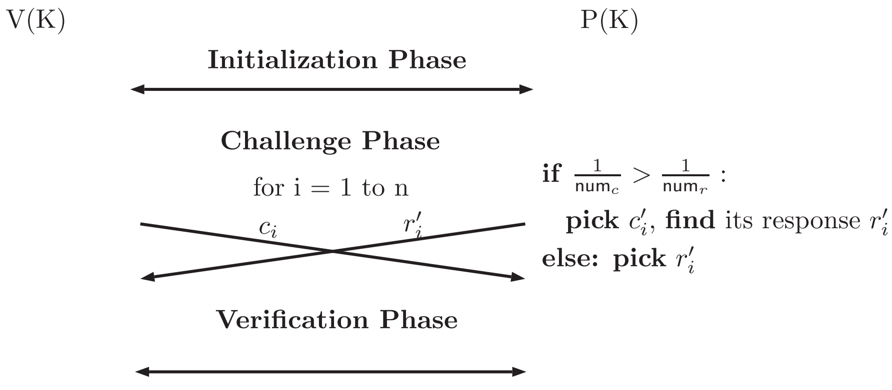

Figure 3.1 – Early-reply strategy of a DF adversary

Theorem 3.1 ([BV14]). *For any PPT adversary playing the DF game in Definition 2.8 or 2.9 with a* τ*-complete DB protocol (Definition 2.5) following the common structure (Definition 2.3) with parameters* $(n, \mathsf{num}_c, \mathsf{num}_r)$*, the probability of success is bounded by* $\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$.

We recall that Tail is defined in Equation (2.1).

This is the optimal security bound that a DB protocol can reach against a distance fraud. DB1 and DB3 protocols [BV14] which are instances of DBopt reach this bound.

The proof of Theorem 3.1 is based on the *early-reply* strategy (See Figure 3.1). In this strategy, depending on num_c and num_r, the malicious prover either guesses the challenge sent by the verifier before it receives or picks a random response in order to sent the response earlier in each round i. If the prover guesses the challenge, the probability of replying correctly at round i is $\frac{1}{\mathsf{num}_c}$ and if it guesses the response the probability of replying correctly at round i is $\frac{1}{\mathsf{num}_r}$ in each round.

Theorem 3.2 ([BV14])**.** *For any PPT adversary playing the MiM game in Definition 2.12 or 2.13 with a* τ*-complete DB protocol (Definition 2.5) following the common structure (Definition 2.3) with parameters* $(n, \mathsf{num}_c, \mathsf{num}_r)$*, the probability of success is bounded by* $\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$.

This is the optimal security bound that a DB protocol can reach against a MiM adversary. All instances of DBopt protocols [BV14] reach this bound.

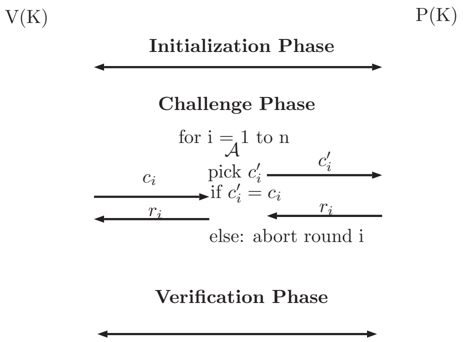

Figure 3.2 – Pre-ask attack by a MiM-adversary

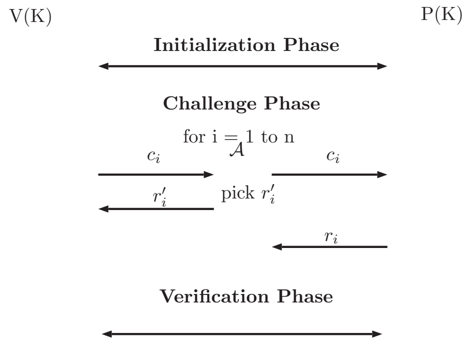

Figure 3.3 – Post-ask attack by a MiM-adversary

The proof of Theorem 3.2 is based on *pre-ask* and *post-ask* strategy (See Figure 3.2 and 3.3).

Both in pre-ask and post-ask (Figure 3.3), the adversary relays the messages between the prover and the verifier in the initialization and the verification phase. In the challenge phase, it does the following:

- *Pre-ask attack [BV14]:* Any malicious actor close to a verifier can do the following in each round i of the challenge phase: Before receiving a challenge c_i from the verifier, he guesses it and sends the guessed challenge c_i' to the far-away prover. He does it early enough to receive a corresponding response r_i' from the prover on time. Meanwhile, the malicious actor receives the challenge c_i from the verifier. If $c_i = c_i'$, then the malicious actor just relays $r_i = r_i'$. Otherwise, he may not pass the round i, especially if P and V authenticate the challenges during the verification phase. The probability that the adversary passes the round i is $\frac{1}{\mathsf{num}_c}$. So, he passes at least τ rounds with probability $\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c})$.

- *Post-ask attack [BV14]:* Any adversary close to a verifier can do the following in each round i of the challenge phase: He receives a challenge c_i from the verifier. Then, he picks a random response r'_i and sends it to the verifier. At the same time, he forwards c_i to the prover. The adversary succeeds to pass the round i with the probability $\frac{1}{\mathsf{num}_r}$. So, he passes at least τ-rounds with probability $\mathsf{Tail}(n,\tau,\frac{1}{\mathsf{num}_r})$.

3.3 Optimal Security Bounds in New Structures

Before introducing the new structures, we give some useful lemmas which are used in proving the new versions of DBopt.

Lemma 3.3. *Let* exp *be an experiment,* $\mathcal{V}$ *be a participant and* t_0 *be a time. We consider a simulation* exp_{t_0} *of the experiment in which each participant* U *stops just before time* $t_0+d(V,U)$*. We denote by* $\mathsf{View}_t^{\mathsf{exp}}(U)$ *and* $\mathsf{View}_t^{\mathsf{exp}_{t_0}}(U)$ *the view of participant* U *at a time* t *in* exp *and* exp_{t_0}*, respectively. For any* $t<t_0+d(V,U)$*,*

$$\mathsf{View}_t^{\mathsf{exp}}(U)=\mathsf{View}_t^{\mathsf{exp}_{t_0}}(U)$$

Proof. We prove it by induction on t such that for all $t<t_0+d(V,U)$. Clearly, $\mathsf{View}_0^{\mathsf{exp}}(U)=\mathsf{View}_0^{\mathsf{exp}_{t_0}}(U)$ at the beginning ($t=0$) of the both experiments. Let us assume that for all U and for all $t'<t_0+d(V,U)$, $\mathsf{View}_{t'}^{\mathsf{exp}}(U)=\mathsf{View}_{t'}^{\mathsf{exp}_{t_0}}(U)$ where $t'<t$. Now, we show that $\mathsf{View}_t^{\mathsf{exp}}(U)=\mathsf{View}_t^{\mathsf{exp}_{t_0}}(U)$ with this assumption.

Let participant U be such that $t<t_0+d(V,U)$. We know that $\mathsf{View}_{t'}^{\mathsf{exp}}(U)=\mathsf{View}_{t'}^{\mathsf{exp}_{t_0}}(U)$. Any incoming message m at time t from a participant U' which is in a different location than U was sent at time $t''<t-d(U,U')$. We have $t''<t_0+d(V,U)-d(U,U')\leq t_0+d(V,U')$. Besides, since U' is at a different location than U, we have $t''<t$ so we can apply the induction hypothesis. Therefore, $\mathsf{View}_{t''}^{\mathsf{exp}}(U')=\mathsf{View}_{t''}^{\mathsf{exp}_{t_0}}(U')$ and so the message m is the same in exp and exp_{t_0}. This applies to all instances at the same location as U, since they locally compute the same messages for each other. Hence, $\mathsf{View}_t^{\mathsf{exp}}(U)=\mathsf{View}_t^{\mathsf{exp}_{t_0}}(U)$.

□

Lemma 3.4. *Given an experiment, if a message* c *is randomly selected with fresh coins by a participant* V *at time* t_0*, any* $\hat{c}$ *received by a participant* U *at time* $t_1<t_0+d(U,V)$ *is statistically independent from* c*.*

Proof. We apply Lemma 3.3. c is not selected at all in exp_{t_0} because V stops just before t_0 in exp_{t_0}. As $t_1<t_0+d(U,V)$, $\hat{c}$ is the same in exp and exp_{t_0}. c is randomly chosen with fresh coins, so $\hat{c}$ is statistically independent from c. □

Remark that Lemma 3.4 differs from the fundamental lemma (Lemma 2.7) as Lemma 3.4 is related with the independence of c received by other parties while the fundamental lemma considers the independence of r from c which is received by V.

3.3.1 Sync Structure

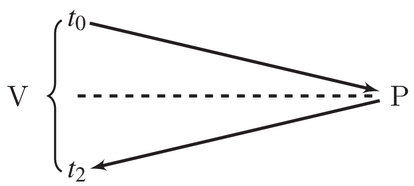

Figure 3.4 – The time check in the common structure is done by measuring the time difference between the curly parenthesis. t shows the time.

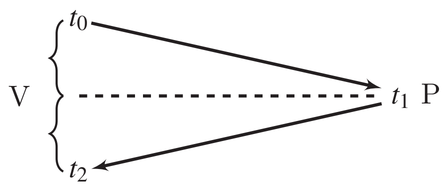

Figure 3.5 – The time check in the sync structure is done by measuring the time difference between the curly parentheses. t shows the time.

Definition 3.5 (Sync Structure)**.** *A DB protocol with the sync structure based on parameters* $(n, \tau, \mathsf{num}_c, \mathsf{num}_r)$ *has an initialization and a verification phase which do not depend on communication times. There is an* n*-round challenge phase between the initialization and the verification phase. The* ***challenge is on time*** *if the elapsed time between sending the challenge (by the verifier) and receiving the challenge (by the prover) (corresponds first part in Figure 3.5) is at most* B*. The* ***response is on time*** *if the elapsed time between sending the response (by the prover) and receiving the response (by the verifier) (corresponds second part in Figure 3.5) is at most* B*. Challenges and responses are in sets of cardinality* num_c *and* num_r*, respectively.*

During the challenge phase, challenges or responses can be corrupted during the transmission. We say that the protocol is τ*-complete when the verifier accepts if and only if at least* τ *rounds have correct and on-time* ***responses and challenges****.*

Differently than the common structure, the arrival time of a challenge is part of the sync structure.

Now, we analyze the optimal security bound of MiM-security in the sync structure.

Theorem 3.6. *Assuming the time when* V *sends his challenge can be predicted by the adversary and* V *and* P *have synchronized clocks, for any PPT adversary playing the MiM game in Definition 2.12 or 2.13 with a* τ*-complete DB protocol (Definition 2.5) following the sync structure with parameters* $(n, \mathsf{num}_c, \mathsf{num}_r)$*, the probability of success is bounded by* $\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c.\mathsf{num}_r})$*.*

Remark that this bound is an improvement compared to Theorem 3.2 in the common structure.

Proof. We consider $\mathcal{V}$, a far-away prover P and a MiM-adversary $\mathcal{A}$ with a noiseless communication. $\mathcal{A}$ relays the messages between $\mathcal{V}$ and P in the initialization and verification phases which are time insensitive. As P is far-away, it cannot just relay the messages. Therefore, it has to guess the challenge and the response before receiving them. Otherwise, it will be too late to make P receive the challenge on time and make

$\mathcal{V}$ receive the response on time thanks to Lemma 3.4. Therefore, it cannot follow either pre-ask or post-ask strategy. We denote that the distance between $\mathcal{V}$ and $\mathcal{A}$ by d_1 and the distance between $\mathcal{A}$ and P by d_2. It can best follow this strategy:

No-Ask Strategy: $\mathcal{A}$ guesses the challenge and the response, and forwards them before seeing them so that they arrive on time. Thanks to our assumption, $\mathcal{A}$ knows the time t_0^i that $\mathcal{V}$ sends a challenge c_i at each round i. In each round i, it picks a challenge c_i' and sends c_i' to P at a time less than equal t_0+B-d_2 so that P receives it at a time $t_1 \leq t_0+B$. It then picks a response r_i' and sends it to $\mathcal{V}$ at a time less than equal t_0+2B-d_1. $\mathcal{V}$ receives r_i' at a time $t_2 \leq t_1+B$. Since $t_1-t_0 \leq B$ and $t_2-t_1 \leq B$, $\mathcal{A}$ succeeds to be "on time" for both the challenge and the response. If r_i' and c_i' are correctly guessed as well then $\mathcal{A}$ passes round i. Hence, the probability that it passes the challenge/response verification for one round is $\frac{1}{\mathsf{num}_c.\mathsf{num}_r}$ and the probability that the $\mathcal{V}$ outputs $\mathsf{Out}_V = 1$ is $\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c.\mathsf{num}_r})$. □

The Problems in the Sync Structure without Synchronization: Remark that in Theorem 3.6, we have the assumption of synchronized clocks between the verifier and the prover. Now, we discuss the reason of this assumption. Let's say that the time difference between the clocks of the verifier and prover is $|\delta|$ [1]. For example, V has time t on its local clock while P has time $T = t+\delta$ on his local clock. V sends the challenge at t_0 according to V's local clock and P receives it at $T_1 \geq t_0+d+\delta$ according to P's local clock where d is the distance of the prover from the verifier. Then, V receives the response at $t_2 \geq t_0+2d$. So, V gets the following result in the verification of timing: $T_1-t_0 \geq \delta+d$ and $t_2-T_1 \geq d-\delta$. If the prover is close, the inequality $|\delta| \leq B-d$ should be satisfied so that P passes the protocol.

In addition, an unsynchronized honest prover and verifier give an advantage to the adversary as pre-ask (for $\delta > 0$) and post-ask (for $\delta < 0$) attacks can be done. Indeed, if the honest prover is far at a distance up to $B+|\delta|$ and at least $\max(B, |\delta|)$, $\mathcal{A}$ passes the protocol with probability $\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$.

Note that $t_2-T_1 \leq B$ and $T_1-t_0 \leq B$ imply that $t_2-t_0 \leq 2B$ which is the verification in the common structure. So, the security results of the common structure apply to the sync structure even if the clocks are not synchronized.

In below attacks, we assume $d = d_1+d_2$ such that d_1 is the distance between the verifier and the adversary, and d_2 is the distance between the prover and the adversary.

- **Pre-Ask:** $\mathcal{A}$ guesses the challenge before it is released and asks for the response to P on time so that it can later on answer. If P and V are synchronized, this strategy never works because $\mathcal{A}$ relays the response from P to V where the distance between them is more than B. However, the following happens if P and V are not synchronized and $\delta > 0$.

  We consider $d = d_1+d_2 \in [\max(B, |\delta|), B+|\delta|]$. V sends the challenge c at t_0. $\mathcal{A}$

[1] If the difference between clocks is not constant it can be still considered as a constant during the protocol as the distance bounding phase takes very short time (order of nanoseconds).

guesses the challenge $\hat{c}$ and sends it to P at t_A which is before receiving the challenge c from V. P receives $\hat{c}$ at $T_1 = t_A + d_2 + \delta$ which is the local time of P. P sends response r and $\mathcal{A}$ relays it to V. V receives r at $t_2 = t_A + 2d_2 + d_1$.

$T_1 - t_0 = t_A + d_2 + \delta - t_0$. By selecting $t_A = t_0 + d_1 - 2\delta$, $T_1 - t_0 = d_1 + d_2 - \delta \in [0, B]$. So the challenge is considered on time.

$t_2 - T_1 = t_A + 2d_2 + d_1 - t_A - d_2 - \delta = d_1 + d_2 - \delta \in [0, B]$. So, the response is considered on time.

Therefore, the pre-ask attack is successful when $\delta > 0$ and the distance between P and V is in between $\mathsf{max}(B, |\delta|)$ and $B + |\delta|$.

- **Post-Ask:** $\mathcal{A}$ guesses the response at the same time that it forwards the challenge to P. If P and V are synchronized, this strategy never works because $\mathcal{A}$ relays the challenge from V to P where the distance between them is more than B. However, the following happens if P and V are not synchronized and $\delta < 0$.

  We consider $d_1 + d_2 \in [-\delta, B - \delta]$. V sends the challenge c, then $\mathcal{A}$ relays c. P receives it at $T_1 = t_0 + d_1 + d_2 + \delta$. Without waiting the response from P, $\mathcal{A}$ guesses the response and sends it at time t_A to V. At the end, V receives it at $t_2 = t_A + d_1$.

  $T_1 - t_0 = t_0 + d_1 + d_2 + \delta - t_0 = d_1 + d_2 + \delta \in [0, B]$. So, the challenge is on time.

  By selecting $t_A = t_0 + d_1 + 2d_2 + 2\delta$, we have $t_2 - T_1 = d_1 + d_2 + \delta \in [0, B]$. So, the response is on time.

  Therefore, the post-ask attack is successful when $\delta < 0$ and the distance between P and V is in between $\mathsf{max}(B, |\delta|)$ and $B + |\delta|$.

  As a result, we have the security bound of Theorem 3.11 if the distance between P and V is more than $B + |\delta|$ even though P and V are not synchronized. However, if P is in the distance between $\mathsf{max}(B, \delta)$ and $B + |\delta|$, we have the lower optimal-security bound as in Theorem 3.2 ($\mathsf{Tail}(n, \tau, \mathsf{max}(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$.

- **The correctness problem:** Beyond security, the other problem in the sync structure with an unsynchronized P and V is correctness as the close-by P cannot pass the protocol, when $d(P, V) \leq B - |\delta|$. Therefore, if the verification fails in the sync structure, V can also do the time verification of the common structure which is checking if $t_2 - t_0 \leq 2B$, but in this case we have a weaker β-security. We stress that this does not require to restart the protocol. We rather obtain a variant of the sync structure which Out_V can take 3 possible values: “reject”, “Common Structure accept”, or “Sync Structure accept”. Applications can decide if a “Common Structure accept” is enough depending on the required security level.

3.3.2 Rand Structure and SyncRand Structure

We think of new structures which are combined with “Common Structure” or “Sync Structure”. In the analysis of “Common Structure” and “Sync Structure”, we assume that the sending time t_0^i of the challenge for each round i in challenge phase is known by the adversary. Now, we suggest a new modification where the verifier randomizes the sending time $t_0^i \in [T, T+\Delta]$ where T and Δ are public and $[T, T+\Delta]$ is uniformly distributed (as real numbers) so that the exact t_0^i cannot be accurately known by the adversary before seeing the challenge.

We note that random delays for the messages (challenges and responses) on both the verifier and the prover side are frequently used for location privacy, as discussed in [RČ08, MOV14]. In our following structures, we use random delays (only on the verifier side) to achieve better security bounds.

Definition 3.7 (Rand Structure)**.** *A DB protocol with the rand structure based on parameters* $(n, \tau, \mathsf{num}_c, \mathsf{num}_r, \Delta)$ *has the same properties with the common structure in Definition 2.3. Additionally, the verifier chooses randomly a sending time in the interval* $[T, T+\Delta]$ *for each challenge in the challenge phase.*

Definition 3.8 (SyncRand Structure)**.** *A DB protocol with the rand structure based on parameters* $(n, \tau, \mathsf{num}_c, \mathsf{num}_r, \Delta)$ *has the same properties with the sync structure in Definition 3.5. Additionally, the verifier chooses randomly a sending time in the interval* $[T, T+\Delta]$ *for each challenge in the challenge phase.*

Theorem 3.9. *For any PPT adversary playing the DF game in Definition 2.8 or 2.9 with a* τ*-complete DB protocol (Definition 2.5) following either the "Rand Structure" or the "SyncRand Structure" with parameters* $(n, \tau, \mathsf{num}_c, \mathsf{num}_r, \Delta)$*, the probability of success is bounded by* $\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}).\frac{2B}{\Delta})$.

Proof. We construct a DF adversary following the early reply strategy: A malicious prover guesses the challenge c_i or the response r_i before it is emitted, and then sends the response at time T_1^i (We use capital T as the prover does not have to be synchronized with the verifier). However, before sending the response, the prover has to guess a proper time T_1^i because the verifier checks the inequalities $t_2^i - t_0^i \leq 2B$ for the "Rand Structure" and $T_1^i - t_0^i \leq B$ and $t_2^i - T_1^i \leq B$ for the "SyncRand Structure". t_2^i is the time that the verifier receives the response and it depends on the sending time T_1^i. It means that $0 \leq t_2^i - t_0^i = T_1^i + d - t_0^i \leq 2B$ where d is the distance between the prover and the verifier. So, we can conclude that if $t_0^i \in [T_1^i + d - 2B, T_1^i + d]$ then P passes i^{th} verification. The probability that it happens is $\frac{2B}{\Delta}$. Once c_i is received, the prover can deduce t_0^i and use $t_1^i = \frac{t_0^i + t_2^i}{2}$ for the verification in the "SyncRand Structure" as the verifier needs to know t_1^i to check if the response and challenge are on time. Therefore, the probability that the prover succeeds the round i is $\max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}).\frac{2B}{\Delta}$ since it also has to guess correctly c_i or r_i. We can conclude that P succeeds at least τ rounds with the probability at least $\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}).\frac{2B}{\Delta})$.

□

Note that there is no change on the optimal MiM-security which is given in Theorem 3.2 in the "Rand Structure". As for the "SyncRand Structure", the new bound is as follows.

Theorem 3.10. *Assuming V and P have synchronized clocks, for any PPT adversary playing the MiM game in Definition 2.12 or 2.13 with a τ-complete DB protocol (Definition 2.5) following the "SyncRand Structure" with parameters $(n, \mathsf{num}_c, \mathsf{num}_r)$, the probability of success is bounded by* $\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c.\mathsf{num}_r}.\frac{B}{\Delta})$.

Proof. We consider $\mathcal{V}$, a far away prover P and MiM adversary $\mathcal{A}$ with a noiseless communication. As showed in Theorem 3.6, $\mathcal{A}$ can use No-ask strategy to pass the protocol. Differently, it also needs to guess a proper time t_A^i to send the guessed challenge to P. P receives the challenge from $\mathcal{A}$ at time t_1^i where $t_1^i = t_A^i + d_2$. If $\mathcal{A}$ passes i^{th} round, the following inequality $0 \leq t_1^i - t_0^i \leq B$ should be satisfied. It means that $0 \leq t_A + d_2 - t_0 \leq B$. If t_A satisfies this inequality then t_0 should be in the interval $[t_A + d_2 - B, t_A + d_2]$. The probability that it happens is $\frac{B}{\Delta}$. Therefore, the probability that prover succeeds the round i is $\frac{1}{\mathsf{num}_c.\mathsf{num}_r}.\frac{B}{\Delta}$ since it also has to guess a correct c_i and r_i. We can conclude that P succeeds at least τ-rounds with the probability at least $\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c.\mathsf{num}_r}.\frac{B}{\Delta})$. □

As a result, among all the structures, "SyncRand Structure" gives the best optimal security bounds for both MiM security and DF security. See Table 3.1 for the review of the optimal bounds for all of the structures.

Structure	**DF**	**MF**
Common	$\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$	$\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$
Sync	$\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$	$^*\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c}.\frac{1}{\mathsf{num}_r})$
Rand	$^*\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}).\frac{2B}{\Delta})$	$\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}))$
SyncRand	$^*\mathsf{Tail}(n, \tau, \max(\frac{1}{\mathsf{num}_c}, \frac{1}{\mathsf{num}_r}).\frac{2B}{\Delta})$	$^*\mathsf{Tail}(n, \tau, \frac{1}{\mathsf{num}_c}.\frac{1}{\mathsf{num}_r}\frac{B}{\Delta})$

Table 3.1 – The review of optimal security bounds in DB structures. The ones with * are different bounds than the bounds in the common structure.

3.4 DBopt in New Structures

We adapt DBopt [BV14] into our new structures 'Sync Structure', 'Rand Structure' and 'SyncRand Structure'. We obtain DBoptSync DBoptRand and DBoptSyncRand, respectively. These versions have some minor differences in the challenge and the verification phase comparing to DBopt.

DBoptSync DBoptRand and DBoptSyncRand are symmetric DB protocols in which P and V share a secret $s \in \mathbb{Z}_2^\ell$ where ℓ is a security parameter. The notations are the following: n is the number of rounds, ℓ_{tag} is the length of the tag, τ is a threshold, $\mathcal{T}$ is the set of all possible time values, q is a prime power.

As in DBopt, we use the function f_s which maps different co-domains depending on the input. $f_s(N_P, N_V, L_\mu, b) \in GF(q)^n$ and $f_s(N_P, N_V, L_\mu, T, b, c) \in GF(q)^{\ell_{\mathsf{tag}}}$. L_μ is a mapping defined from a vector $\mu \in \mathbb{Z}_2^\ell$ where $L_\mu(s) = (\mu(s), \mu(s), \ldots, \mu(s))$ and $\mu(s) = \mathsf{map}(\mu.s)$ such that $\mathsf{map} : \mathbb{Z}_2 \to GF(q)$ is an injection. Here $N_P, N_V \in \{0,1\}^{\ell_{\mathsf{nonce}}}$, $L_\mu \in \mathcal{L}$ where $\mathcal{L}$ includes all possible L_μ mappings, $b, c \in GF(q)^n$ and $T \in \mathcal{T}^n$.

Verifier		**Prover**
secret: s		secret: s
	initialization phase	
pick $L_\mu \in_U \mathcal{L}, N_V \in_U \{0,1\}^{\ell_{\mathsf{nonce}}}$	$\xleftarrow{N_P}$	pick $N_P \in_U \{0,1\}^{\ell_{\mathsf{nonce}}}$
select $b \in \mathsf{GF}(q)^n$		
$a = f_s(N_P, N_V, L_\mu, b)$	$\xrightarrow{N_V, L_\mu, b}$	$a = f_s(N_P, N_V, L_\mu, b)$
$s' = L_\mu(s)$		$s' = L_\mu(s)$
	challenge phase	
	for $i = 1$ to n	
pick $c_i \in_U \mathsf{GF}(q)$		
start time t_0^i	$\xrightarrow{c_i}$	receive c_i', save time t_1^i
receive r_i, stop time t_2^i	$\xleftarrow{r_i'}$	$r_i' = \phi_{c_i'}(a_i, s_i', b_i)$
	verification phase	
receive c'', check $\mathsf{tag} = f_s(N_P, N_V, L_\mu, T, b, c'')$	$\xleftarrow{c', T, \mathsf{tag}}$	$\mathsf{tag} = f_s(N_P, N_V, L_\mu, T, b, c')$
check $\#\{i; c_i = c_i'', r_i \text{ and } t_2^i, t_1^i \text{ correct}\} \geq \tau$	$\xrightarrow{\mathsf{Out}_V}$	

Figure 3.6 – DBoptSync

The *initialization phases* of DBoptSync DBoptRand and DBoptSyncRand are the same as in the DBopt protocol [BV14] (See Figure 3.6). The *challenge phases* are as follows:

DBoptSync: P saves the time t_1^i of receiving the challenge c_i' from V at round i and V saves the times t_0^i and t_2^i which are the time of sending the challenge c_i and receiving the response r_i'[2], respectively.

DBoptRand: It is as the challenge phase of DBopt except that V randomizes the sending time $t_0^i \in [T, T+\Delta]$ where T and Δ are public and $[T, T+\Delta]$ is uniformly distributed (as real numbers) for each round i in the challenge phase. V saves the sending time $t_0^i \in [T, T+\Delta]$ of the challenge c_i and saves the receiving time t_2^i the response r_i' from P in each round i.

DBoptSyncRand: V randomizes the sending time $t_0^i \in [T, T+\Delta]$ as in DBoptRand. Then, as in DBoptSync, P saves the time t_1^i of receiving the challenge c_i' from V at round i and V saves the times t_0^i and t_2^i which are the time of sending the challenge c_i and

[2] We use the notations c_i' and r_i' instead of c_i and r_i as received messages because when a message arrives it may change on the way because of the noise.

receiving the response r'_i.

The verification phase of DBoptRand is the same as DBopt. The *verification phase* of DBoptSync and DBoptSyncRand is as follows: P sets $T = (t_1^1, t_1^2, ..., t_1^n)$ and $c' = (c'_1, c'_2, ..., c'_n)$ and calculates the tag $f_s(N_P, N_V, L_\mu, T, b, c')$. Then, P sends the tag and V does the following:

- V first checks if the tag and (c', T) are compatible, which means the tag it received is equal to $f_x(N_P, N_V, L_\mu, T, b, c')$. If it is compatible, it continues with the next step. Otherwise, it rejects P.
- V counts the number of correct rounds. A round is correct if $c'_i = c_i$ and $r'_i = r_i$. If the number of correct rounds are less then τ, it rejects P and outputs $\mathsf{Out}_V = 0$. Otherwise, it continues with the next step.
- V checks challenges and responses arrived on time for each correct round i. A challenge and a response is on time if $t_0^i \leq t_1^i \leq t_2^i$, $t_1^i - t_0^i \leq B$ and $t_2^i - t_1^i \leq B$, respectively. If the number of on time and correct rounds is at least τ, then V accepts P and outputs $\mathsf{Out}_V = 1$. Otherwise, it rejects.

We note that the on time condition of DBoptSync and DBoptSyncRand implies $t_2^i - t_0^i \leq 2B$, which is the only timing verification in DBopt [BV14]. Therefore, the DBoptSync's timing condition is more restrictive.

In Section 3.5, we consider $\Delta = 100B$ for DBoptRand and DBoptSyncRand. For instance, $\Delta = 1\mu s$ (microseconds) and $B = 10ns$ (this corresponds to 3 m according to speed of light). n rounds take n μs which is reasonable.

The responses are computed depending on the concrete instance of b and ϕ_{c_i}. There are three protocols defined in [BV14] whose instances are given in Table 3.2. Hence, DBoptSync, DBoptRand and DBoptSyncRand have the same instances as well.

Protocol	q	map	b	ϕ_{c_i}
DB1	$q > 2$	$\mathsf{map}(u) \neq 0$	no b used	$\phi_{c_i}(a, x'_i, b_i) = a_i + c_i x'_i$
DB2	$q = 2$	$\mathsf{map}(u) = u$	Hamming weight $\frac{n}{2}$	$\phi_{c_i}(a, x'_i, b_i) = a_i + c_i x'_i + c_i b_i$
DB3	$q \geq 2$	no map used	Hamming weight n	$\phi_{c_i}(a, x'_i, b_i) = a_i + c_i b_i$

Table 3.2 – Classification of the protocols according the selection of b and ϕ in DBoptSync DBoptRand and DBoptSyncRand [BV14]

Theorem 3.11 (MiM Security)**.** *Assuming that V and P are synchronized, the DBoptSync protocol with the selection of b and ϕ as in Table 3.2 is MiM-secure with the success probabilities*

- *(DB1 and DB2)* $\mathsf{Tail}(n, \tau, \frac{1}{q^2}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + (r+1)\mathsf{Adv}(C\text{-}PRF) + r 2^{-\ell_{\mathsf{tag}}}$ *when f is a circular PRF (Definition 2.16),*
- *(DB3)* $\mathsf{Tail}(n, \tau, \frac{1}{q^2}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF) + 2^{-\ell_{\mathsf{tag}}}$ *when f is a PRF (Definition 2.15).*

Here, r is the number of honest instances.

If $\mathsf{Adv}(C\text{-}PRF)$, $\mathsf{Adv}(PRF)$, $2^{-\ell_{\mathsf{nonce}}}$ and $2^{-\ell_{\mathsf{tag}}}$ are negligible, DB1, DB2 and DB3 are **optimal** for the security according to Theorem 3.6.

Proof. The proof is the same with [BV14] until Γ_3.

Γ_0 : We consider a distinguished experiment $\mathsf{exp}(\mathcal{V})$ with no close-by prover and $\mathcal{V}$ accepts with probability p_0. We consider a game Γ_0 where we simulate $\mathsf{exp}(\mathcal{V})$. The success probability of this game is p_0. We reduce Γ_1, Γ_2 and Γ_3 as in [BV14].

Γ_1 : We reduce Γ_0 to Γ_1 whose success additionally requires that for every (N_P, N_V, L_μ) triplet, there is no more than one instance $P(s)$ and one instance $V(s)$ using this triplet. As $P(s)$ is honest and $P(s)$ and $V(s)$ are selecting N_P and N_V at random, respectively, the success probability of Γ_1 is at least $p_0 - \frac{r^2}{2}2^{-\ell_{\mathsf{nonce}}}$.

The following games are for $DB1$ and $DB2$.

Γ_2 : We reduce Γ_1 to Γ_2 where $\mathcal{V}$ never accepts forged tag. f_x satisfies the circular PRF assumptions (See Definition 2.16) as shown in [BV14]. It means that the tag can be forged with probability $\mathsf{Adv}(C\text{-}PRF) + 2^{-\ell_{\mathsf{tag}}}$. Therefore, the success probability of G_2 is at least $p_0 - \frac{r^2}{2}2^{-\ell_{\mathsf{nonce}}} - r\varepsilon - r2^{-\ell_{\mathsf{tag}}}$ (See [BV14] for the full proof of this step).

Γ_3 : In Γ_3, we replace the oracle $f_s(.)$ by $O_{\tilde{s},F}$ and obtain a simplified game Γ_3. Γ_3's requirements for the success is the same with Γ_2. So, we have $p_3 \geq p_0 - \frac{r^2}{2}2^{-\ell_{\mathsf{nonce}}} - (r+1)\mathsf{Adv}(C\text{-}PRF) - r2^{-\ell_{\mathsf{tag}}}$.

We now detail the analysis of Γ_3 which differs from [BV14]. In Γ_3, P and $\mathcal{V}$ never repeat the nonces and use a random function F to select a. So, the distinguished $\mathcal{V}$ has a single matching P and these two instances pick a at random. Furthermore, acceptance implies that both instances have seen the same L_μ, T, b, c. The acceptance message of $\mathcal{V}$ (Out_V) also depends on the correct and on time responses and challenges. In the case that $\mathcal{V}$ accepts P, P has to receive the challenge c on time and $\mathcal{V}$ has to receive the corresponding response r on time for at least τ rounds. Let's denote t_0^i the time when $\mathcal{V}$ sends c_i, t_1^i the time when P receives c_i' and t_2^i the time when $\mathcal{V}$ receives r_i. Thanks to Lemma 3.4, in order to have on time responses and challenges, the challenge that $P(s)$ receives should be independent from the challenge that is sent by $V(s)$. As the challenge c is randomly selected by $V(s)$, the message that $P(s)$ received matches with probability $\frac{1}{q}$.

Similarly, if we exchange the roles of P and $\mathcal{V}$ in Lemma 3.4 and replace t_0 with t_1^i and t_1^i with t_2^i, we can conclude that r that $\mathcal{V}(s)$ receives is independent from the response r_i' that is sent by $P(s)$ as well. The response functions on DB1, DB2 in each round i depends on challenge, a_i and s_i'. In Γ_3, a_i is random in $GF(q)^n$. As $\phi_{c_i'}(a_i, s_i', b_i) = a_i + g(c_i', s_i', b_i)$ where g is a function (See Table 3.2 for the details of g) we can assume that a_i is randomly selected in $GF(q)$ just when r_i' is computed. Equivalently, r_i is uniformly selected in $GF(q)$ just before being sent. So, $r_i = r_i'$ with probability $\frac{1}{q}$.

As a result, we have $p_0 \leq \mathsf{Tail}(n, \tau, \frac{1}{q^2}) + \frac{r^2}{2}2^{-\ell_{\mathsf{nonce}}} + (r+1)\mathsf{Adv}(C\text{-}PRF) + r2^{-\ell_{\mathsf{tag}}}$ for DB1 and DB2 .

For DB3's analysis, we use a random oracle for PRF in Γ_2 and obtain $p_0 \leq \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF)$. Similarly, we define a game Γ_3 where the tag is never forged and we obtain $p_0 \leq \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF) + 2^{-\ell_{\mathsf{tag}}}$ with a reduction from Γ_2 to Γ_3. Then, we can make the same analysis as Γ_3 above and obtain $p_3 \leq \mathsf{Tail}(n, \tau, \frac{1}{q^2})$ because of Lemma 3.3. In the end, we have $p_0 \leq \mathsf{Tail}(n, \tau, \frac{1}{q^2}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF) + 2^{-\ell_{\mathsf{tag}}}$ for DB3.

□

Theorem 3.12 (MiM security)**.** *Assuming that $\mathcal{V}$ and P are synchronized, the sending time of the challenge is randomized and the time interval $[T, T+\Delta]$ to send the challenge is public. Then the DBoptSyncRand protocol is MiM-secure with the success probabilities*

- *(b and ϕ as in DB1 and DB2 [BV14])* $\mathsf{Tail}(n, \tau, \frac{1}{q^2} . \frac{B}{\Delta}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + (r+1)\mathsf{Adv}(C\text{-}PRF) + r 2^{-\ell_{\mathsf{tag}}}$ *when f is a circular PRF [BV14],*
- *(b and ϕ as in DB3 [BV14])* $\mathsf{Tail}(n, \tau, \frac{1}{q^2} . \frac{B}{\Delta}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF) + 2^{-\ell_{\mathsf{tag}}}$ *when f is a PRF.*

Here, r is the number of honest instances of the prover and K is a complexity bound on the experiment and ϕ is response function. β is negligible for $\frac{\tau}{n} \geq \frac{1}{q^2} + cte$ and r and K polynomially bounded and ε is negligible.

If $\mathsf{Adv}(PRF)$, $\mathsf{Adv}(C\text{-}PRF)$, $2^{-\ell_{\mathsf{nonce}}}$ and $2^{-\ell_{\mathsf{tag}}}$ are negligible, DB1, DB2 and DB3 are **optimal** for the security according to Theorem 3.10.

Proof. The proof is the same as Theorem 3.11 until game Γ_3. The success of Γ_3 depends on the correct and on time responses and challenges. Lemma 3.4 shows that the challenge and the response have to be independent in each round so that they arrive on time. These independent responses and challenges can be correct with probability $\frac{1}{q^2}$ (See the proof of Theorem 3.11). Additionally, they can be on time with probability $\frac{B}{\Delta}$ as showed in Theorem 3.10. Therefore, the probability of one successful round is $\frac{1}{q^2} . \frac{B}{\Delta}$.

Consequently, success probability Γ_0 is at least $\mathsf{Tail}(n, \tau, \frac{1}{q^2} . \frac{B}{\Delta}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + (r+1)\mathsf{Adv}(C\text{-}PRF) + r 2^{-\ell_{\mathsf{tag}}}$ for DB1 and DB2. For DB3, it is at least $\mathsf{Tail}(n, \tau, \frac{1}{q^2} . \frac{B}{\Delta}) + \frac{r^2}{2} 2^{-\ell_{\mathsf{nonce}}} + \mathsf{Adv}(PRF) + 2^{-\ell_{\mathsf{tag}}}$.

□

Theorem 3.13 (DF security)**.** *The DBoptSyncRand and DBoptRand protocols are DF secure with the success probabilities*

- *(DB1 and DB3)* $\mathsf{Tail}(n, \tau, \frac{1}{q} . \frac{2B}{\Delta})$,
- *(DB2)* $\sum\limits_{\substack{i+j \geq \tau \\ i,j \leq n/2}}^{n} \binom{n/2}{i} (\frac{2B}{\Delta})^i (1 - \frac{2B}{\Delta})^{\frac{n}{2}-i} \binom{n/2}{j} (\frac{B}{\Delta})^j (1 - \frac{B}{\Delta})^{\frac{n}{2}-j}$

DB1 and and DB3 are **optimal** for the DF-resistance according to Theorem 3.9, while DB2 cannot reach the optimal bounds for DF.

Proof. We consider distinguished experiment $\mathsf{exp}(V)$ with no close-by participant. Due to the Fundamental Lemma (Lemma 2.7), the response r_i is independent from c_i. For DB1 and DB2, r_i is correct with probability $\frac{1}{q}$. As r_i has to be arrived on time, the proper time has to be chosen. As stated in Theorem 3.9 the sending time is chosen correctly with probability $\frac{2B}{\Delta}$. So, the probability of success in one round i is $\frac{1}{q}.\frac{2B}{\Delta}$.

In DB2, half of the rounds where $x' = b_i$ are correct because of the hamming weight of b. Therefore, the only necessity in these rounds is sending the response in a correct time which can be chosen well with probability $\frac{2B}{\Delta}$. For the remaining rounds ($\frac{n}{2}$ rounds), at least $\tau - \frac{n}{2}$ rounds should pass correctly. The correct response is chosen with the probability $\frac{1}{2}$ and correct time with the probability $\frac{2B}{\Delta}$. □

3.5 Performance

Three adaptations DBoptSync DBoptSyncRand and DBoptRand of DBopt have different success probabilities for DF and MiM security. DBoptSync and DBoptSyncRand have better bound against mafia fraud compared to DBopt while DBoptRand has the same security bound against MiM adversary with DBopt. In addition, DBoptRand and DBoptSyncRand have the same and better success probability for distance fraud compared to DBopt but DBoptSync is the same with DBopt.

Assuming a noise level of $p_{\mathsf{noise}} = 0.05$ and $\frac{B}{\Delta} = 0.01$, we get results in Table 3.3 and 3.4. We find τ in terms of rounds n such that $\mathsf{Tail}(n, \tau, 1 - p_{\mathsf{noise}}) \approx 99\%$ for τ-completeness. Table 3.3 shows the required number of rounds for the DF security. Table 3.4 shows the number of rounds required for the MiM security. We used Theorem 3.11, 3.12, 3.13 and theorems in [BV14] to compute the required number of rounds to achieve security level.

	$\ell = 2^{-10}$				$\ell = 2^{-20}$			
	DB1 ($q = 3$)	DB1 ($q = 4$)	DB2	DB3	DB1 ($q = 3$)	DB1 ($q = 4$)	DB2	DB3
DBoptSync	14	12	69	24	24	20	123	43
DBoptSyncRand	3	3	2	3	6	6	2	6
DBoptRand	3	3	2	3	6	6	2	6
DBopt	14	12	69	24	24	20	123	43

Table 3.3 – Number of required rounds to be secure against a distance fraud where ℓ is the security level in DB protocols. The bold protocols improve DBopt

As we can see in Table 3.3 and Table 3.4, we can use DB2 with 5 rounds (instead of 123) in DBoptSyncRand and reach a pretty good security. If synchronized clocks are not realistic, we can see that we have a much better DF-security with DBoptRand with the same number of rounds.

	$\ell = 2^{-10}$			$\ell = 2^{-20}$		
	DB1 ($q = 3$)	DB1 ($q = 4$)	DB2-DB3	DB1 ($q = 3$)	DB1 ($q = 4$)	DB2-DB3
DBoptSync	7	6	12	12	8	20
DBoptSyncRand	3	1	3	5	5	5
DBoptRand	14	12	24	24	20	43
DBopt	14	12	24	24	20	43

Table 3.4 – Number of required rounds to be secure against a MiM adversary where ℓ is the security level in DB protocols. The bold protocols improve DBopt

3.6 Conclusion

We define new structures for DB protocols. The first structure is the “Sync Structure” where the prover measures the time as well as the verifier. We modify the DBopt [BV14] according to sync structure and we get DBoptSync which has better security against MiM adversary. Then, we add a new modification which is randomizing the sending challenge time to both “Common Structure” and “Sync Structure” and obtain the second and third structures “Rand Structure” and “SyncRand Structure”, respectively. Similarly, we modify the DBopt and DBoptSync protocols based on these structures and get better security bounds against distance fraud for the DBoptSyncRand and DBoptRand protocols and MiM adversary for DBoptSyncRand protocol. We give the optimal security bounds against distance fraud and MiM adversary for all DB protocols that follows these new structures.

Chapter 4

Efficient Public-key Distance Bounding

In some applications such as payment systems, using public-key distance bounding protocols is practical as no pre-shared secret is necessary between the payer and the payee. In general, such applications use powerless devices with RFID and NFC technologies. Therefore, they may suffer from energy constraints because of very limited computation resources. On the other hand, the public-key cryptography requires much more computations than symmetric-key cryptography. So, the inefficiency may cause problems on these powerless devices when they need to do the public-key cryptography related computations.

In this chapter, we focus on the efficiency problem in public-key distance bounding protocols and the formal security proofs of them. We construct two protocols Eff-pkDB and Eff-pkDB^p (the former without privacy, the latter with) which require fewer computations on the prover side compared to the existing protocols, while keeping the highest security level.

The content of this chapter was published in ASIACRYPT16 [KV16].

Related Works:

Table 4.1 shows the security and the efficiency properties of existing public-key protocols and our protocols (See Appendix A for the details and analysis of the procols in Table 4.1). We can see that most of the previous public-key DB protocols [BC93, BB05, GOR14a, Vau15d, Vau15c, Vau15a, ABG$^+$17] do not concentrate on this efficiency problem, except HPO [HPO13]. So far, HPO is the most efficient one among them as it requires only 4 elliptic curve (EC) multiplications on the prover side, but it is not strong private [Vau15b] and it is not secure against DH (See Appendix A.2, Figure A.3) and TF. In addition to this, its security is based on several ad-hoc assumptions [HPO13] which are not so well studied: “OMDL”, “Conjecture 1”, “extended ODH” and “XL”.

GOR [GOR14a] (Appendix A.3) is constructed to have strong privacy and anonymity against verifier, but it has been shown [Vau15b] that it is neither strong private nor

Protocol	**MiM**	**DF**	**DH**	**TF**	**Privacy**	**Strong Privacy**
Brands-Chaum [BC93]	✓	✓	×	×	×	×
HPO [HPO13]	✓	✓	×	×	✓	×
GOR [GOR14a]	✓	✓	×	×	×	×
PaySafe [CGDR+15]	✓*	×	×	×	×	×
PrivDB [Vau15c]	✓	✓	✓	×	✓	✓
ProProx [Vau15d]	✓	✓	✓	✓	×	×
eProProx [Vau15a]	✓	✓	✓	✓	✓	✓
TREAD [ABG+17]	✓	✓	✓	×	✓	✓
Simp-pkDB [KV16]	✓	✓	×	×	×	×
Eff-pkDB [KV16]	✓	✓	✓	×	×	×
Eff-pkDB^p [KV16]	✓	✓	✓	×	✓	✓

Table 4.1 – The review of the existing public-key DB protocols. ✓ means that it is secure for corresponding threat model and × means it is not. ✓* means that it is secure against the adversaries that cannot relay the messages close to the speed of light.

private.

ProProx [Vau15d] (Appendix A.4) provides MiM, DF and DH security and extractor based TF-security [BV14] but it is not private. Its version eProProx [Vau15a] is a extension with strong privacy. However, both ProProx and eProProx suffer from heavy cryptographic operations such as zero-knowledge (ZK) proofs in order to achieve extractor based TF-security [BV14]. These are the only extractor based TF-secure protocols, but we can see that their cost is unreasonable.

PrivDB [Vau15c] (Appendix A.5) and our new protocol Eff-pkDB^p have the same security properties. However, PrivDB is a bit less efficient on the prover side than Eff-pkDB^p and it has no light privacy-less variant, contrarily to Eff-pkDB^p.

TREAD [ABG+17] (Appendix A.6) is a very efficient public-key DB compared to its security level. It is MiM, DF and DH secure and strong private. It is claimed that TREAD is simulator based TF-secure [DFKO11] (SimTF) but we realize that the proof of SimTF security is not correct (See Appendix A.6). Thus, we do not consider the SimTF security of TREAD in our comparisons. We remark that the TF-security (extractor based) of and ProProx and eProProx [Vau15a] is stronger than claimed SimTF security of TREAD. The extractor based TF-security is stronger because it guarantees that the malicious prover cannot get help from an adversary to pass the protocol without leaking its secret key. TREAD is very similar to PrivDB [Vau15c]. Differently, it has a small trick in order to achieve the claimed SimTF-security. In this trick, the information given to the adversary to pass the protocol lets the adversary replay. The same trick can be applied to Eff-pkDB and Eff-pkDB^p with preserving the efficiency.

PaySafe [CGDR+15] is a very efficient protocol designed for contactless payment, but we do not compare it with the other protocols because it assumes a weaker adversarial model. It is only secure against MiM. It is not secure against DF, DH and TF because the response of the prover in the challenge phase does not depend on any message of the verifier. It also does not protect the privacy of the prover.

Adding privacy in public-key DB protocols is yet another challenge. Strong privacy cannot be achieved so easily as shown in Section 4.4.2. HPO and GOR failed in this.

4.1 Our Contribution

Our contributions are as follows:

- We design two public-key DB protocols Eff-pkDB and Eff-pkDB^p. The first protocol is secure against **DF, MF and DH** but it is not private. It uses **only one public key related operation** on the prover side. Basically, this protocol can be used in applications not requiring privacy in a very efficient way. Then, we modify this protocol by adding a public-key encryption to make it **strong private**. Both protocols are **quite efficient compared with the previous protocols**. Our constructions are generic based on a key agreement protocol, a weakly-secure symmetric DB protocols, and a cryptosystem. We formally prove the security following the BMV model [BMV13a, BMV15, BMV13b] (See Section 2.2.1) which was adapted to public-key DB in Vaudenay [Vau15c].
- We define a new key agreement (KA) security game (D-AKA). In literature, the extended Canetti-Krawczyk (eCK) security model [LLM07] is widely accepted for KA. However, **a weaker security model (D-AKA) is sufficient** for the security of our new public-key DB protocols as we care both the efficiency and the security. Finally, we design a D-AKA secure key agreement protocol (Nonce-DH) based on the hardness of the GDH problem and a random oracle. The Nonce-DH key agreement protocol can be used in our DB constructions.
- We construct another reasonable protocol Simp-pkDB which was our first attempt to construct an efficient and a secure protocol. Although this protocol is quite efficient and **does not require any public-key of a verifier**, it fails in DH-security.
- We compare the efficiency and security level of our protocols and we see that our lighter protocol Eff-pkDB and our first attempt Simp-pkDB are the most efficient public-key DB protocols. We give a detailed analysis and comparison between existing public-key DB protocols in Section 4.5.

Structure of the Chapter: In Section 4.2 and in Section 4.3, we introduce our new key-agreement security model and more results about one time security of DB [Vau15c] that we give in Section 2.2.1. The security model and the results are the basis of our constructions Eff-pkDB and Eff-pkDB^p. Then, in Section 4.4, we give our constructions Eff-pkDB and Eff-pkDB^p together with a variant and Simp-pkDB. We conclude this chapter with Section 4.5.

4.2 Authenticated Key Agreement (AKA) Protocols

4.2.1 One-Pass AKA Model

In this section, we show our new KA security model and some preliminaries about the AKA protocols. The security models in this section are used *to construct secure and private public-key DB protocols* Eff-pkDB and Eff-pkDB^p in Section 4.4.

We note that Eff-pkDB and Eff-pkDB^p in Section 4.4 can employ any eCK-secure [LLM07] key agreement protocol to have the same security properties. However, eCK-security is stronger than what we need in our protocols. Therefore, we define a weaker notion **to have simpler, more efficient and secure public-key DB.**

Definition 4.1 (Authenticated Key Agreement (AKA) in One-Pass)**.** *A one-pass AKA protocol (See Figure 4.1) is a tuple* $(\mathsf{Gen}_A,\ \mathsf{Gen}_B, D, A, B)$ *of PPT algorithms. Let A and B be the two parties. A and B generate secret/public key pairs* $(\mathsf{sk}_A, \mathsf{pk}_A)$ *and* $(\mathsf{sk}_B, \mathsf{pk}_B)$ *with the algorithms* $\mathsf{Gen}_A(1^\ell)$ *and* $\mathsf{Gen}_B(1^\ell)$*, respectively where* ℓ *is the security parameter. B picks N from the sampling algorithm D and runs* $B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N)$ *which outputs the session key s. Then, it sends N and finally, A gets the session key s by running* $A(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B, N)$*. We say that AKA is correct, if A and B obtain the same s at the end of the protocol for all N and random coins.*

$$\begin{array}{lcr}
\underline{\mathbf{A}}(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B) & & \underline{\mathbf{B}}(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A) \\
 & & N \leftarrow D(1^\ell) \\
A(\mathsf{sk_A}, \mathsf{pk_A}, \mathsf{pk_B}, \mathsf{N}) \to s & \xleftarrow{\quad \mathsf{N} \quad} & B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N) \to s
\end{array}$$

Figure 4.1 – The structure of an authenticated key-agreement (AKA) protocols in one pass.

We now give the security definition of one-pass AKA protocol.

Definition 4.2 (Decisional Authenticated Key-Agreement (D-AKA) Security)**.** *We define set up of two oracles with* $\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{sk}_B, \mathsf{pk}_B$.

$$\begin{array}{ll}
\underline{O_{\boldsymbol{A}}(.,.):} & \underline{O_{\boldsymbol{B}}(.):} \\
\quad \textbf{\textit{return}}\ A(\mathsf{sk}_A, \mathsf{pk}_A, ., .) & \quad N' \leftarrow D(1^\ell) \\
 & \quad s' \leftarrow B(\mathsf{sk}_B, \mathsf{pk}_B, ., N') \\
 & \quad \textbf{\textit{return}}\ s', N'
\end{array}$$

Given $b \in \{0,1\}$ *and the oracles* $O_A(.,.), O_B(.)$*, the game* $\mathsf{KA}_{b,\mathcal{A}}^{O_A(.,.),O_B(.)}(\ell)$ *is as follows:*

1. *Challenger executes* $\mathsf{Gen}_A(1^\ell) \to (\mathsf{sk}_A, \mathsf{pk}_A), \mathsf{Gen}_B(1^\ell) \to (\mathsf{sk}_B, \mathsf{pk}_B)$*, sets up the oracles, calls* $O_B(\mathsf{pk}_A) \to (s_0, N)$ *and randomly picks* s_1*. Then, it sends* $s_b, N, \mathsf{pk}_A, \mathsf{pk}_B$ *to the adversary* $\mathcal{A}$*.*

2. $\mathcal{A}$ *has access to the oracle* $O_B(.)$ *and* $O_A(.,.)$ *under the condition of not querying the input* (pk_B, N) *to the oracle* O_A*. Eventually,* $\mathcal{A}$ *outputs* b'*.*

We define the advantage of this game as:

$$\mathsf{Adv}(\mathsf{KA}_{b,\mathcal{A}}^{O_A(.,.),O_B(.)}(\ell)) = |\Pr[\mathsf{KA}_{0,\mathcal{A}}^{O_A(.,.),O_B(.)}(\ell)=1] - \Pr[\mathsf{KA}_{1,\mathcal{A}}^{O_A(.,.),O_B(.)}(\ell)=1]|.$$

A one-pass AKA protocol with $(\mathsf{Gen}_A(1^\ell), \mathsf{Gen}_B(1^\ell), D, A, B)$ *is D-AKA secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}(\mathsf{KA}_{b,\mathcal{A}}^{O_A(.,.),O_B(.)}(\ell))$ *is negligible.*

We show that eCK-security implies D-AKA security in Theorem B.1 in Appendix B. It means that Eff-pkDB and Eff-pkDB^p can employ eCK-secure key agreement protocols as well.

We show in the following lemma that the probability that the same N is picked by the oracle B is negligible when we have a D-AKA security.

Lemma 4.3. *Assume that we have a key agreement protocol with* $(\mathsf{Gen}_A, \mathsf{Gen}_B, D, A, B)$*. We define the random variables* $(\mathsf{sk}_A, \mathsf{pk}_A)$*,* $(\mathsf{sk}_B, \mathsf{pk}_B)$ *generated with* $\mathsf{Gen}_A(1^\ell)$ *and* $\mathsf{Gen}_B(1^\ell)$ *respectively, and* (s,N) *and* (s',N') *generated by* $O_B(\mathsf{pk}_A)$*. If the key agreement protocol is D-AKA secure, then* $\Pr[N=N']$ *is negligible in* ℓ*. Furthermore, for all values* u *which could depend on* $\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{sk}_B, \mathsf{pk}_B$*,* $\Pr[N=u]$ *is negligible.*

Proof. We define an adversary $\mathcal{A}$ playing the D-AKA game as follows:

$\underline{\mathcal{A}}$
receive $s_b, N, \mathsf{pk}_B, \mathsf{pk}_A$
$(s',N') \leftarrow O_B(\mathsf{pk}_A)$
if $N' = N$
    if $s' = s_b$: **output** 0
    else: output 1
else:
    output $b' \leftarrow \{0,1\}$

In this strategy, $\mathcal{A}$ wins if $N=N'$ (except $s_1 = s_0$ and $b=1$). Otherwise, he wins with the probability $\frac{1}{2}$.

$$\begin{aligned}\Pr[\mathcal{A}\,\mathsf{win}] &= \frac{1}{2}(1-\Pr[N=N']) + \Pr[N=N'] - \Pr[N=N', s_1=s_0, b=1]\\ &= \frac{1}{2} + \frac{1}{2}\Pr[N=N'] - \Pr[N=N', s_1=s_0, b=1]\end{aligned}$$

We know from the D-AKA security that $\Pr[\mathcal{A}\,\mathsf{win}] - \frac{1}{2}$ is negligible. $\Pr[s_1 = s_0] = 2^{-\ell}$ is negligible as well. So, $\Pr[N=N']$ is negligible. Now, we need to show that it holds for all values u in the distribution D.

Let v be the most probable value for N. We have

$$\begin{aligned}\Pr[N=N'] &= \sum_w \Pr[N=N'=w]\\ &= \sum_w \Pr[N=w]^2\\ &\geq \Pr[N=v]^2\end{aligned}$$

So, we have the following inequality in the end:

$$\Pr[N=u] \leq \Pr[N=v] \leq \sqrt{\Pr[N=N']}$$

We know that $\Pr[N=N']$ is negligible so $\Pr[N=u]$ is negligible.

□

We also give a privacy definition for one-pass AKA. This definition is for the privacy of the party which runs the algorithm B.

Definition 4.4 (D-AKA^p Privacy)**.** *Given $b \in \{0,1\}$ and the oracle $O_A(.,.)$ (as defined in Definition 4.2), the game $\mathsf{pKA}_{b,\mathcal{A}}^{O_A(.,.)}(\ell)$ is follows:*

1. *Challenger runs $\mathsf{Gen}_A(1^\ell) \to (\mathsf{sk}_A, \mathsf{pk}_A)$ and $\mathsf{Gen}_B(1^\ell) \to (\mathsf{sk}_{B_1}, \mathsf{pk}_{B_1})$, sets up the oracle and gives $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ and sk_{B_1} to $\mathcal{A}$.*
2. *$\mathcal{A}$ generates $(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0})$ with $\mathsf{Gen}_B(1^\ell)$ and sends $(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0})$ to the challenger.*
3. *Challenger runs $D(1^\ell) \to N$ and then $B(\mathsf{sk}_{B_b}, \mathsf{pk}_{B_b}, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N) \to s$. Then, it sends s to the adversary $\mathcal{A}$.*
4. *$\mathcal{A}$ has access to the oracle O_A without any constraint. Eventually, $\mathcal{A}$ outputs b'. (Remark that $\mathcal{A}$ does not know N.)*
5. *The advantage of the game is*

$$\mathsf{Adv}(\mathsf{pKA}_{b,\mathcal{A}}^{O_A(.,.)}(\ell)) = \Pr[\mathsf{pKA}_{0,\mathcal{A}}^{O_A(.,.)}(\ell) = 1] - \Pr[\mathsf{pKA}_{1,\mathcal{A}}^{O_A(.,.)}(\ell) = 1].$$

An AKA protocol $(\mathsf{Gen}_A(1^\ell), \mathsf{Gen}_B(1^\ell), D, A, B)$ is D-AKA^p private if for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}(\mathsf{pKA}_{b,\mathcal{A}}^{O_A(.,.)}(\ell))$ is negligible.

Basically, in D-AKA^p privacy, we want to make sure that an adversary which may corrupt a party who runs B cannot easily decide who generated a session key s.

4.2.2 A One-Pass AKA Protocol (Nonce-DH)

$\underline{\mathbf{A}}(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B)$		$\underline{\mathbf{B}}(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A)$
		$\texttt{pick}\, N \in \{0,1\}^n$,
$H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_B^{\mathsf{sk}_A}, N) \to s$	$\xleftarrow{\quad N \quad}$	$H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_B}, N) \to s$

Figure 4.2 – The Nonce-DH key agreement protocol.

We construct a D-AKA secure protocol (Nonce-DH) based on the Diffie-Hellman (DH) problem [DH76] as in Figure 4.2. Here, g is a generator of cyclic group $\mathbb{G}$ of prime order

q. g and q depend on the security parameter ℓ. The parties know each others' public keys beforehand, where $\mathsf{pk}_A = g^{\mathsf{sk}_A}$ and $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ and sk_A and sk_B are the corresponding secret keys which are uniformly picked in $\mathbb{Z}_q$.

The party B has the input $(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A)$. He randomly picks N from $\{0,1\}^n$ and computes $B(\mathsf{sk}_B, \mathsf{pk}_B, \mathsf{pk}_A, N) = H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_B}, N)$ to get s. The party A computes $A(\mathsf{sk}_A, \mathsf{pk}_A, \mathsf{pk}_B, N) = H(g, \mathsf{pk}_B, \mathsf{pk}_A, \mathsf{pk}_B^{\mathsf{sk}_A}, N)$ and gets s. Here, H is a deterministic function.

Clearly, Nonce-DH is correct as H is deterministic.

Theorem 4.5. *Assuming that the GDH problem is hard in $\mathbb{G}$ (See Definition 2.17) and $n = \Omega(\ell)$, Nonce-DH is D-AKA secure in the random oracle model.*

Proof. Γ_0: The game is the D-AKA game. The challenger works as follows: He picks q and g as described in Nonce-DH. He randomly picks $\mathsf{sk}_A, \mathsf{sk}_B \in \mathbb{Z}_q$, and computes $\mathsf{pk}_A = g^{\mathsf{sk}_A}$, $\mathsf{pk}_B = g^{\mathsf{sk}_B}$. He picks randomly $s_1 \in \{0,1\}^\ell$ and then he gets (s_0, N) from $\mathcal{O}_B(\mathsf{pk}_A)$ as defined below. Then, he gives $g, q, \mathsf{pk}_A, \mathsf{pk}_B, N, s_b$ to the adversary $\mathcal{A}$. $\mathcal{A}$ has an access to the random oracle H, $\mathcal{O}_A(.,.)$ (with the restriction not asking for pk_B, N) and $\mathcal{O}_B(.)$ defined below.

$\underline{\mathcal{O}_A(.,.)}$	$\underline{\mathcal{O}_B(.)}$	$\underline{H(.)}$
Input: pk'_B, N'	**Input:** pk'_A	**Input:** U
if (pk'_B, N') **equals** (pk_B, N)	**pick** $N' \in \{0,1\}^n$	**if** $(U,.) \in T$
return $\perp$	$H(g, \mathsf{pk}_B, \mathsf{pk}'_A, \mathsf{pk}'^{\mathsf{sk}_B}_A, N') \to s$	**return** V where $(U,V) \in T$
else:	**return** (s, N')	**else:**
$H(g, \mathsf{pk}'_B, \mathsf{pk}_A, \mathsf{pk}'^{\mathsf{sk}_A}_B, N') \to s$		**pick** $V \in \{0,1\}^\ell$
return s		**store** (U,V) to T
		return V

We let $\perp$ be a special symbol which is unavailable to $\mathcal{A}$. The success probability of $\mathcal{A}$ in Γ_0 is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where the oracle $\mathcal{O}_B$ never selects again the nonce N (which is obtained by the first call to $\mathcal{O}_B$). As a nonce in Γ_0 is equal to N with the probability $\frac{1}{2^n}$, $|p_1 - p_0| \leq \frac{q_B}{2^n}$ where q_B is the number of queries to $\mathcal{O}_B$. Due to $n = \Omega(\ell)$, $p_1 - p_0$ is negligible.

Γ_2 : We reduce Γ_1 to Γ_2 where we replace H with H'. H' is defined with an access to a DDH oracle (as Definition 2.17) as follows:

$\underline{H'(.)}$
Input: $U = (w, x, y, z, N')$
if $w = g$ **and** $\mathsf{DDH}(g, x, y, z) \to 1$:
  $z \leftarrow \perp$
return $H(w, x, y, z, N')$

As there is one-to-one mapping in the transformation of (g, x, y, z, N') and $\perp$ cannot be used by $\mathcal{A}$ in queries to H', the success probability of Γ_2 remains the same which means $p_2 = p_1$.

Γ_3 : We define another game Γ_3 where the only difference from Γ_2 is that we replace the oracle $\mathcal{O}_B$ with the oracle $\mathcal{O}'_B$.

$\underline{O'_B(.)}$
Input: pk'_A
pick $N' \in \{0,1\}^\ell$
$H(g, \mathsf{pk}_B, \mathsf{pk}'_A, \perp, N') \to s$
send (s, N')

Note that O'_B queries H instead of H' and $N' \neq N$ due to the reduction to Γ_1. Γ_3 is exactly same with Γ_2 so the success probabilities p_3 and p_2 are the same.

Now in Γ_3, sk_B is used only by the DDH oracle.

Γ_4 : We reduce Γ_3 to Γ_4 where $\mathcal{A}$ does not make the query $(g, \mathsf{pk}_B, \mathsf{pk}_A, z, N)$ with $z = \mathsf{pk}_A^{\mathsf{sk}_B}$ to H'. Indeed, any such query can be filtered using the DDH oracle and stopped to solve the GDH problem. As the GDH problem is hard, $\mathcal{A}$ in Γ_3 selects $z = \mathsf{pk}_A^{\mathsf{sk}_B}$ given $(\mathsf{pk}_A, \mathsf{pk}_B)$ with negligible probability. Therefore, $p_4 - p_3$ is negligible.

In Γ_4, $(g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N)$ is queried only once to H and this query is only done by the challenger.

Γ_5 : We reduce Γ_4 to Γ_5 where the challenger picks a random s_0 instead of getting s_0 from H.

Γ_4 and Γ_5 are the same because if $(g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N)$ is never being queried again, it is not necessary that H stores $((g, \mathsf{pk}_B, \mathsf{pk}_A, \perp, N), s_0)$ in T. So, $p_4 = p_5$.

In Γ_5, s_0 and s_1 play a symmetric role and could be erased with b from the game after s_b is released. So, the state of the game after erasure of b, s_0 and s_1 are independent from b. Hence, $p_5 = \frac{1}{2}$ leading to $p_0 - \frac{1}{2}$ is negligible.

□

Theorem 4.6. *Assuming that $n = \Omega(\ell)$, Nonce-DH is D-AKA^p private in the random oracle model.*

Proof. Γ_0 : The game Γ_0 is D-AKA^p game. The challenger works as follows: He picks q and g as described in Nonce-DH. He selects $\mathsf{sk}_A, \mathsf{sk}_{B_1} \in \mathbb{Z}_q$, and computes $\mathsf{pk}_A = g^{\mathsf{sk}_A}$ and $\mathsf{pk}_{B_1} = g^{\mathsf{sk}_{B_1}}$. Then, he sends $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ and sk_{B_1} to $\mathcal{A}$. $\mathcal{A}$ selects sk_{B_0} and pk_{B_0} and sends them to the challenger. Next, the challenger picks $b \in \{0,1\}$, $N \in \{0,1\}^n$, queries $(g, \mathsf{pk}_{B_b}, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N)$ to H and receives s. He sends s to $\mathcal{A}$. $\mathcal{A}$ has an access to the oracle H and to the oracle $O_A(.,.)$ as defined in the proof of Theorem 4.5.

Γ_1 : We reduce Γ_0 to Γ_1 where $\mathcal{A}$ never selects the nonce N in the query of the oracle H or O_A. The probability that he selects N again is $\frac{1}{2^n}$ so $p_2 - p_1$ is negligible.

Γ_2 : We reduce Γ_1 to Γ_2 where O_B picks s at random instead of a response from H. As $(g, \mathsf{pk}_{B_b}, \mathsf{pk}_A, \mathsf{pk}_A^{\mathsf{sk}_{B_b}}, N)$ is queried only one time by the challenger, we have $p_1 = p_2$. Now, b is never used in Γ_2. It means that s is independent from b, so $p_2 = \frac{1}{2}$. Therefore, $p_0 - \frac{1}{2}$ is negligible.

□

Table B.1 in Appendix B shows that Nonce-DH which is secure in our weaker model is more efficient than the previous KA protocols.

4.3 More Security Results on OTDB

In this section, we define new security notions related to one-time secure DB. We define them to be able to have the result in Theorem 4.8 which helps us to prove the security of our constructions.

Definition 4.7 (Multi-verifier OT-MiM:). *The OT-MiM game with more than one verifier instance is called as multi-verifier OT-MiM-security.*

Remark that in OT-MiM definition (Section 2.2.1), we have only one verifier instance. Now, we complement the known security results of OTDB.

Theorem 4.8. *OTDB [Vau15c] (See Section 2.2) is multi-verifier OT-MiM secure.*

Proof. Γ_0: In this game, an adversary $\mathcal{A}$ plays multi-verifier OT-MiM game. Here, we have a distinguished verifier instance $\mathcal{V}$ with other instances $\{V_1, ..., V_k\}$ and one prover instance P. The success probability of Γ_0 is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where at most one verifier instance outputs 1. Let's say E is an event in Γ_0 where at least two verifier instances output 1 ($\mathsf{Out}_V = 1$). To reduce Γ_0 to Γ_1, we show that $\Pr[E]$ is negligible.

First, we define hybrid games $\Gamma_{i,j}$'s to analyze $\Pr[E]$. $\Gamma_{i,j}$ is similar to Γ_0 except the game stops right after V_i and V_j have sent their final outputs and all Out_V is replaced by 0 except V_i and V_j. The adversary wins the game if $\mathsf{Out}_{V_i} = \mathsf{Out}_{V_j} = 1$.

In $\Gamma_{i,j}$, we define three arrays for the challenges. The first array C_{V_i} includes the challenges sent by V_i, the second array C_{V_j} includes the challenges sent by V_j and the third array C_P includes the challenges seen by P. The bits in C_{V_i} and C_{V_j} are independent. We also define a response function $\mathsf{resp}_k(c) = a_{2k+c-1}$ for each round k. As the bits of the secret s are independent, the bits of $\{\mathsf{resp}_k(0)||\mathsf{resp}_k(1)\}_{k=1}^n$ are independent as well. If $C_{V_i}[k] \neq C_{V_j}[k]$, then the adversary could have taken $C_P[k] = c$ where c is equal either $C_{V_i}[k]$ or $C_{V_j}[k]$ and learn $\mathsf{resp}_k(c)$. So, he responds correctly to either V_i or V_j for sure, but to the other instance with probability $\frac{1}{2}$. We define an event $E_{ij,k}$ where the responses are correct for V_i and V_j in round k. Clearly, all events $\{E_{ij,k}\}_{k=1}^n$ are independent. So, $\Gamma_{i,j} = \prod_k \Pr[E_{ij,k}]$. Hence,

$$\Pr[E_{ij,k}] \leq \Pr[C_{V_i}[k] = C_{V_j}[k]] + \Pr[E_{ij,k}|C_{V_i}[k] \neq C_{V_j}[k]] \times \Pr[C_{V_i}[k] \neq C_{V_j}[k]] \leq \frac{3}{4}$$

So, the adversary wins $\Gamma_{i,j}$ with the probability $(\frac{3}{4})^n$ which is negligible.

Now, we can analyze E.

$$\Pr[E] \leq \sum_{i,j} \Pr[\Gamma_{i,j}] = \mathsf{negl}(n)$$

As E happens with the negligible probability, we can reduce Γ_0 to Γ_1 and conclude $p_1 - p_0$ is negligible. For Γ_1 to succeed, only $\mathcal{V}$ must produce $\mathsf{Out}_V = 1$.

Γ_2 : We reduce Γ_1 to Γ_2 where we simulate all verifier instances except $\mathcal{V}$. We can do this simulation because the messages but Out_V sent by a verifier does not depend on the secret. As $\mathsf{Out_V} = 0$ for all verifier instance except $\mathcal{V}$ in the winning case (only $\mathcal{V}$ can output 1), $p_1 \leq p_2$.

Now in Γ_2, we are in OT-MiM game where there is only one verifier instance $\mathcal{V}$ and one prover instance P. By using the OT-MiM-security result of OTDB [Vau15c], we deduce p_2 is negligible so p_0 is negligible.

□

Definition 4.9 (Multi-verifier Impersonation Fraud (IF)). *The game begins by running the key setup algorithm $\mathcal{K}$ of a symmetric DB protocol $DB = (\mathcal{K}, P, V, B)$ which outputs s. It consists of verifier instances running $V(s)$ and an adversary. The adversary wins if any verifier instance outputs $\mathsf{Out}_V = 1$. A distance bounding protocol is multi-verifier IF-secure, if for any such game, the probability of an adversary to win is negligible.*

Note that MiM-security implies multi-verifier IF-security. In Theorem 4.10, we prove that OT-MiM-security also implies multi-verifier IF-security for a DB protocol following the canonical structure (Definition 2.4). This result will be used to prove DH-security of our constructions.

Theorem 4.10. *If a (symmetric) DB protocol following the canonical structure is OT-MiM secure, then it is multi-verifier IF-secure.*

Proof. We take an adversary $\mathcal{M}$ playing the multi-verifier IF game. $\mathcal{M}$ interacts with polynomially many verifier instances V_j's. We define adversaries $\mathcal{A}_i$'s playing the OT-MiM game. $\mathcal{A}_i$ simulates $\mathcal{M}$ and takes the verifier instance V_i as $\mathcal{V}$ in the OT-MiM game. Concretely, we number the V_j's by their order of appearance during the simulation of $\mathcal{M}$. When $\mathcal{M}$ queries $V_1, ..., V_{i-1}, V_{i+1}, ..., V_k$ (where k is the total number of verifier instances), $\mathcal{A}_i$ just simulates them (this is possible as the protocol follows the canonical structure. So, no message from the verifier except Out_V depends on s). If Out_V needs to be returned to $\mathcal{M}$, $\mathcal{A}_i$ returns 0. When $\mathcal{M}$ queries V_i, $\mathcal{A}_i$ relays it to $\mathcal{V}$ and sends the response of $\mathcal{V}$ to $\mathcal{M}$.

Let E_i be the event in the multi-verifier IF game which is $\mathsf{Out}_{V_i} = 1$ and all previously released Out_V are equal to 0. Clearly, we have $\Pr[\mathcal{M}\,\mathsf{wins}] = \sum_{i \geq 1} \Pr[\mathcal{M}\,\mathsf{wins} \wedge E_i]$. On the other hand, $\Pr[\mathcal{M}\,\mathsf{wins} \wedge E_i] \leq \Pr[\mathcal{A}_i\,\mathsf{wins}]$ because for all coins making $\mathcal{M}$ win the multi-verifier IF-game and E_i occur at the same time, we have $\mathsf{Out}_{V_j} = 0$ for all $j < i$ and $\mathsf{Out}_{V_i} = 1$ so the same coins make $\mathcal{A}_i$ win the OT-MiM game. So, $\Pr[\mathcal{M}\,\mathsf{wins}] \leq \sum_{i \geq 1} \Pr[\mathcal{A}_i\,\mathsf{wins}]$. Due to OT-MiM security, $\Pr[\mathcal{A}_i\,\mathsf{wins}]$ is negligible for every i. So, $\Pr[\mathcal{M}\,\mathsf{wins}]$ is negligible. So, we have multi-verifier IF-security.

□

Thanks to Theorem 4.10, OTDB [Vau15c] in Figure 2.1 is multi-verifier IF-secure.

4.4 Our Constructions

In this section, we first introduce our new protocol Eff-pkDB which is secure against DF, MF and DH and then Eff-pkDB^p which is a variant of it preserving the strong privacy as well. We also present two new protocols related to Eff-pkDB which have some advantages and disadvantages against Eff-pkDB and Eff-pkDB^p. Finally, we introduce another efficient public-key DB protocol Simp-pkDB which does not require any key setup for the verifier.

4.4.1 Eff-pkDB

$$
\begin{array}{lcr}
\underline{V}(\mathsf{sk}_V,\mathsf{pk}_V) & & \underline{P}(\mathsf{sk}_P,\mathsf{pk}_P,\mathsf{pk}_V) \\
 & & N \leftarrow D(1^\ell) \\
A(\mathsf{sk}_V,\mathsf{pk}_V,\mathsf{pk}_P,N) \to s & \xleftarrow{\mathsf{N,pk_P}} & B(\mathsf{sk}_P,\mathsf{pk}_P,\mathsf{pk}_V,N) \to s \\
 & \xleftrightarrow{\mathsf{symDB}(s)} & \\
 & \xrightarrow{\mathsf{Out}_V} &
\end{array}
$$

Figure 4.3 – Eff-pkDB

Eff-pkDB is constructed with a one-pass AKA and a symmetric DB protocol. P and V first agree on a secret key s using an AKA protocol $(\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D)$. Then, they together run a symmetric key DB protocol symDB by using s as a secret key.

Using OTDB in Figure 2.1 as symDB and using Nonce-DH in Section 4.2.2 as an AKA protocol appear to be enough for its security as shown in the following theorems.

Theorem 4.11. *If symDB is DF-secure, then Eff-pkDB is DF-secure.*

Proof sketch: The malicious and far away prover with its instances plays the DF game. We can easily reduce it to a game where a distinguished verifier $\mathcal{V}$ and the malicious prover receive the same s' from outside (even if maliciously selected). As symDB is DF-secure, the prover passes the protocol with the negligible probability. □

Theorem 4.12. *If symDB is multi-verifier OT-MiM-secure and the one-pass AKA* $(\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D)$ *is D-AKA secure, then Eff-pkDB is MiM-secure.*

Proof. Γ_0 : The adversary plays the MiM game in Eff-pkDB with the distinguished verifier $\mathcal{V}$, instances of the verifier and instances of the prover. $\mathcal{V}$ receives pk_P and a given N. We call "matching instance" the instance who sends this N.

Γ_1 : We reduce Γ_0 to Γ_1 where no nonce produced by any prover instance is duplicated or equal to any nonce received by any verifier instance before. Thanks to Lemma 4.3 which says that repeating a nonce picked using D is negligible, $p_1 - p_0$ is negligible. So, the matching instance (if any) is unique and sets N before it is sent to $\mathcal{V}$.

Γ_2 : We simulate the prover instances and $\mathcal{V}$ as below in this game. Basically, in Γ_2, the prover and the verifier do not use the secret generated by the oracles O_B and O_A, respectively.

$\underline{P(.)}$ (in Γ_2)	$\underline{V(.)}$ (in Γ_2)
run $O_B(\mathsf{pk}_V) \to (s_0, N')$	**receive** N', pk_P
send N', pk_P	**if** $(N', ., \mathsf{pk}_P) \in T$
pick s_1	**retrieve** s from T **where** $(N', s, \mathsf{pk}_P) \in T$
store (N', s_1, pk_P) **in** T	**else:**
run $\mathsf{symDB}(s_1)$	$O_A(\mathsf{pk}_P, N') \to s$
	run $\mathsf{symDB}(s)$

With the reduction from Γ_1 to Γ_2, we show that the secret generated by A and B are indistinguishable from the randomly picked secret. The reduction is showed below using the D-AKA security of AKA:

We define the hybrid games $\Gamma_{2,t}$ to show $p_2 - p_1$ is negligible. Here, $t \in \{0, 1, 2, ..., k\}$ and k is the number of prover instances bounded by a polynomial.

$\Gamma_{2,i}$: $\mathcal{V}$ is simulated as in Γ_2. The j^{th} instance of P is simulated as in Γ_1 for all $j \leq i$ and as in Γ_2 for all $j > i$. Clearly, $\Gamma_{2,0} = \Gamma_2$ and $\Gamma_{2,k} = \Gamma_1$.

First, we show that $\Gamma_{2,i}$ and $\Gamma_{2,i+1}$ are indistinguishable. For this, we use an adversary $\mathcal{B}$ that plays the D-AKA game. $\mathcal{B}$ receives $\mathsf{pk}_A, \mathsf{pk}_B, s_b, N$ from the D-AKA challenger and simulates either $\Gamma_{2,i}$ or $\Gamma_{2,i+1}$ against the adversary $\mathcal{A}$ which distinguishes $\Gamma_{2,i}$ and $\Gamma_{2,i+1}$. $\mathcal{B}$ lets $\mathsf{pk}_V = \mathsf{pk}_A$ and $\mathsf{pk}_P = \mathsf{pk}_B$ in his simulation. $\mathcal{B}$ simulates each prover instance P_j (j^{th} prover instance) as described below.

$\underline{P_j(.)}$
if $j < i+1$
  $O_B(\mathsf{pk}_V) \to (s', N')$ (Using the D-AKA game)
else if $j > i+1$
  pick s'
  store (N', s', pk_P) **to** T
else: $(j = i+1)$
  $s' \leftarrow s_b$ **and** $N' \leftarrow N$ (s_b and N were received from the D-AKA-game as a challenge)
  store (N', s', pk_P) **to** T
send N', pk_P
run $\mathsf{symDB}(s')$

Note that if $b = 0$ in the D-AKA game which means s_b is generated by the oracle O_B then $\mathcal{B}$ simulates the game $\Gamma_{2,i+1}$. Otherwise, he simulates $\Gamma_{2,i}$.

For the verifier simulation, $\mathcal{B}$ first checks, if $(N', ., \mathsf{pk}_P)$ is stored by himself as $\mathcal{V}$ in Γ_2. Otherwise, he sends (pk_P, N') to the oracle O_A and receives s'. As (N, s_b, pk_P) is always stored in T, (pk_P, N) is not queried to O_A oracle by $\mathcal{B}$. At the end of the game, $\mathcal{A}$ sends his decision. If $\mathcal{A}$ outputs i, then $\mathcal{B}$ outputs 1. If $\mathcal{A}$ outputs $i+1$, then $\mathcal{B}$ outputs 0. Clearly, the advantage of $\mathcal{B}$ is $p_{2,i} - p_{2,i+1}$. Due to the D-AKA security, we obtain that $p_{2,i} - p_{2,i+1}$ is negligible. From the hybrid theorem, we can conclude that $p_{2,0} - p_{2,k}$ is negligible where $p_{2,0} = p_2$ and $p_{2,k} = p_1$.

Γ_3 : We simulate the prover instances as below so that they do not run the oracle O_B to have N. The only change in this game is the generation of the nonce. As the prover in Γ_3 picks the nonce from the same distribution that O_B picks, $p_3 = p_2$. This game shows that the prover generates N' (and also s_1) independently from O_B.

$\underline{P(.)}$ (in Γ_3)
   pick $N' \in D(1^n)$
   send N', pk_P
   pick s_1
   store (N', s_1, pk_P) **to** T
   run symDB(s_1)

Γ_4 : We reduce Γ_3 to the multi-verifier OT-MiM-security game Γ_4 where there is only matching instance and the other instances are simulated. With this final reduction, we show that the adversary has to break the multi-verifier OT-MiM-security of symDB in order to break the MiM-security of Eff-pkDB.

The reduction is the following: $\mathcal{A}^3$ plays Γ_3. We construct an adversary $\mathcal{A}_i^4$ in Γ_4. $\mathcal{A}_i^4$ receives N from the matching prover in Γ_4. $\mathcal{A}_i^4$ takes P_i as a matching prover in Γ_3 where $i \in \{1, ..., k\}$. $\mathcal{A}_i^4$ simulates all of the provers except P_i against $\mathcal{A}^3$. For P_i, $\mathcal{A}_i^4$ just sends (pk_P, N). In the end, if P_i is the matching instance in Γ_3 and $\mathcal{A}^3$ wins then $\mathcal{A}_i^4$ wins. Therefore $p_3 \leq \sum_i p_{4,i}$ where $p_{4,i}$ is the probability that $\mathcal{A}_i^4$ wins. Due to multi-verifier OT-MiM-security, all $p_{4,i}$'s are negligible. So, p_3 is negligible. Hence, p_0 is negligible. □

Theorem 4.13. *If symDB is OT-MiM-secure, OT-DH-secure and follows the canonical structure, and if the one-pass AKA* $(\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D)$ *is D-AKA secure, then Eff-pkDB is DH-secure.*

Proof. Γ_i is a game and p_i denotes the probability that Γ_i succeeds.

Γ_0 : The adversary P with its instances plays the DH-security game in Eff-pkDB with the distinguished verifier $\mathcal{V}$, other instances of the verifier and an honest prover P'. The probability that the adversary succeeds in Γ_0 is p_0.

Γ_1 and Γ_2 : These games are like in the proof of Theorem 4.12 except that P_j is replaced by P'_j. The reduction from Γ_0 to Γ_1 and Γ_1 to Γ_2 is similar to the proof of Theorem 4.12. So we can conclude that $p_2 - p_0$ is negligible.

We let N be the nonce produced by the instance of P' and s_1 be its key which is playing a role during the challenge phase of $\mathcal{V}$ in the DH game.

Γ_3 : We reduce Γ_2 to a game Γ_3 in which all Out_V from a verifier instance who receives pk_P and N is replaced by 0 during the initialization phase. Intuitively, in this case, Out_V cannot be equal 1 because if it is 1, it means P' impersonates P. The reduction is as follows: During the initialization game, P' sends messages which do not depend on s_1 because of the canonical structure, and which can be simulated. So, we can reduce this phase to the multi-verifier IF game and use Theorem 4.10 to show that $p_3 - p_2$ is negligible. This reduction shows that the DH-adversary P cannot win the game with sending pk_P and N generated by P'.

Γ_4 : We reduce Γ_3 to Γ_4 where the game stops after the challenge phase for $\mathcal{V}$. As the verification phase which is after the challenge phase is non-interactive and Out_V is determined at the end of the challenge phase, $p_4 = p_3$.

Γ_5 : We reduce Γ_4 to Γ_5 which is OT-DH game. In Γ_4, s_1 has never been used so s (the key of $\mathcal{V}$ which is given by the adversary) is independent from s_1. In this case, P' and $\mathcal{V}$ run symDB with independent secrets. So, $p_5 = p_4$. Because of the OT-DH security of symDB, p_5 is negligible. □

4.4.2 Eff-pkDB^p

Eff-pkDB is not strong private as the public key of the prover is sent in clear. Adding one encryption operation to Eff-pkDB is enough to have strong privacy.

Eff-pkDB^p in Figure 4.4 is the following: The prover and the verifier generate their secret/public key pairs by running the algorithms $\mathsf{Gen}_P(1^\ell)$ and $\mathsf{Gen}_V(1^\ell)$, respectively. We denote $(\mathsf{sk}_P, \mathsf{pk}_P)$ for the secret/public key pair of the prover and $(\mathsf{sk}_V, \mathsf{pk}_V)$ for the secret/public key pair of the verifier where $\mathsf{sk}_V = (\mathsf{sk}_{V_1}, \mathsf{sk}_{V_2})$ and $\mathsf{pk}_V = (\mathsf{pk}_{V_1}, \mathsf{pk}_{V_2})$. The first key of the verifier is used for the encryption and the second key is used for the AKA protocol. The prover picks N from the sampling algorithm D and generates s with the algorithm $B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N)$. Then, he encrypts pk_P and N with pk_{V_1}. After, he sends the ciphertext e to the verifier. The verifier decrypts e with sk_{V_1} and learns N and pk_P which helps him to understand who is interacting with him. Next, the verifier runs $A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N)$ and gets s. Finally, the prover and verifier run a symmetric DB protocol symDB protocol with s.

$$
\begin{array}{lcr}
\underline{V}(\mathsf{sk}_V, \mathsf{pk}_V) & & \underline{P}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V) \\
 & & N \leftarrow D(1^\ell) \\
 & & B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N) \rightarrow s \\
\mathsf{pk}_P, N = \mathsf{Dec}_{\mathsf{sk}_{V_1}}(e) & \xleftarrow{\quad e \quad} & e = \mathsf{Enc}_{\mathsf{pk}_{V_1}}(\mathsf{pk}_P, N) \\
A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N) \rightarrow s & & \\
 & \xleftrightarrow{\mathsf{symDB}(s)} & \\
 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure 4.4 – Eff-pkDB^p: private variant of Eff-pkDB

We can easily show that Eff-pkDB^p is DF, MiM, DH-secure from Theorem 4.11, 4.12, 4.13 with the same assumptions, respectively. To prove this, we start from an adversary playing the DF, MiM or DH-security game against Eff-pkDB^p. We construct an adversary playing the same game against Eff-pkDB to whom we give sk_{V_1}. The simulation is straightforward. Now, we show the strong privacy of Eff-pkDB^p.

Theorem 4.14. *Assuming the one-pass AKA* $(\mathsf{Gen}_A, \mathsf{Gen}_B, A, B, D)$ *is D-AKA^p secure and the cryptosystem is IND-CCA secure, then Eff-pkDB^p is strong private in the HPVP model (Definition 2.14).*

Proof. Γ_0 : The adversary $\mathcal{A}$ plays the HPVP privacy game.

Γ_1 : The verifiers skip the decryption when they receive a ciphertext produced by any prover and continue with the values encrypted by the prover. Because of the correctness of the encryption scheme $p_1 = p_0$.

Γ_2 : This game is the same as Γ_1 the except the provers encrypt a random string instead of pk_P, N. The verifier retrieves e and s from the table T so that it does not decrypt any ciphertext that comes from a prover as in Γ_1. Thanks to the IND-CCA security (Verifiers are simulated using a decryption oracle due to our Γ_1 reduction. The use of this oracle is valid in IND-CCA game), $p_2 - p_1$ is negligible. So, provers and the verifier works as follows:

$\underline{P(.)}$ (in Γ_2)
pick $N \in D(1^\ell)$
$B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N) \rightarrow s$
pick r
$e \leftarrow \mathsf{Enc}_{\mathsf{pk}_{V_1}}(r)$
store (e, s) **to** T
send e
run $\mathsf{symDB}(s)$

$\underline{V(.)}$ (in Γ_2)
receive e
if $(e, .) \in T$
  retrieve s from T
  where $(e, s) \in T$
else:
  $(\mathsf{pk}', N) \leftarrow \mathsf{Dec}_{\mathsf{sk}_{V_1}}(e)$
  $A(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}', N) \rightarrow s$
run $\mathsf{symDB}(s)$

This reduction shows that the adversary cannot retrieve pk_P and N from the encryption.

Γ_3 : It is the same as Γ_2 except that we simulate the prover as below. In this game, s is generated independently from sk_P and pk_P.

$\underline{P(.)}$ (in Γ_3)
$\mathsf{Gen}_B(1^\ell) \rightarrow (\mathsf{sk}, \mathsf{pk})$
pick $N \in D(1^\ell)$
run $B(\mathsf{sk}, \mathsf{pk}, \mathsf{pk}_{\mathsf{V}_2}, \mathsf{N}) \rightarrow s$
pick r
$e \leftarrow \mathsf{Enc}_{\mathsf{pk}_{V_1}}(r)$
store (e, s) **to** T
send e
run $\mathsf{symDB}(s)$

We defined hybrid games $\Gamma_{3,t}$ to show $p_3 - p_2$ is negligible. Here, $t \in \{0, 1, 2, ..., k\}$ and k is the number of prover instances bounded by a polynomial.

$\Gamma_{3,i}$: $\mathcal{V}$ is simulated as in Γ_3. The j^{th} instance of P is simulated as in Γ_2 if $j \leq i$ and as in Γ_3 if $j > i$.

First, we show that $\Gamma_{3,i}$ and $\Gamma_{3,i+1}$ are indistinguishable. For this, we use an adversary $\mathcal{B}$ that plays D-AKA^p game. $\mathcal{B}$ receives $\mathsf{pk}_A, \mathsf{pk}_{B_1}$ and sk_{B_1} from the D-AKA^p challenger, generates $(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0})$ by running $\mathsf{Gen}_B(1^\ell)$ and sends $(\mathsf{sk}_{B_0}, \mathsf{pk}_{B_0})$ to the D-AKA^p game. Finally, $\mathcal{B}$ receives s. After, he begins simulating either $\Gamma_{3,i}$ or $\Gamma_{3,i+1}$ against the adversary $\mathcal{A}$ that wants to distinguish $\Gamma_{3,i}$ and $\Gamma_{3,i+1}$. In the simulation, $\mathcal{B}$ lets $\mathsf{pk}_V = \mathsf{pk}_A$ and $\mathsf{pk}_P = \mathsf{pk}_{B_1}$. It can simulate the `Corrupt` oracle called by $\mathcal{A}$ since sk_{B_1} is available. For all of the prover instances P_j where $j \neq i+1$, it simulates normally and for P_{i+1}, it simulates as follows:

$\underline{P_{i+1}(.)}$
pick r
$e \leftarrow \mathsf{Enc}_{\mathsf{pk}_V}(r)$
store (e,s) **to** T
send e
run $\mathsf{symDB}(s)$

Note that if s is generated by running $B(\mathsf{sk}_{B_0},\mathsf{pk}_{B_0},\mathsf{pk}_V,N)$ then $\mathcal{B}$ simulates $\Gamma_{3,i}$ and if it is generated from $B(\mathsf{sk}_{B_1},\mathsf{pk}_{B_1},\mathsf{pk}_V,N)$ then $\mathcal{B}$ simulates $\Gamma_{3,i+1}$.

For the verifier simulation, $\mathcal{B}$ first checks if $(e,.)$ is stored by himself as $\mathcal{V}$ in Γ_3. Otherwise, he decrypts e and sends (pk_{P_j},N) to the oracle $\mathcal{O}_A(\mathsf{pk}_P,N)$ and receives s. At the end of the game, $\mathcal{A}$ sends his decision. If $\mathcal{A}$ outputs i, then $\mathcal{B}$ outputs 0. If $\mathcal{A}$ outputs $i+1$, then $\mathcal{B}$ outputs 1. Clearly, the advantage of $\mathcal{B}$ is $p_{3,i}-p_{3,i+1}$ which is negligible because of the D-AKA^p assumption. From the hybrid theorem, we can conclude that $p_{3,0}$ and $p_{3,k}$ is negligible where $p_{3,0}=p_2$ and $p_{3,k}=p_3$.

Now, in Γ_3, no identity is used by the provers. Hence, $\mathcal{A}$ does not have any advantage to guess the prover which means $p_3=\frac{1}{2}$. As a result of it, $p_0-\frac{1}{2}$ is negligible.

□

Consequently, if we use D-AKA secure and D-AKA^p private key agreement protocol in Eff-pkDB^p, then we have DF, MF, DH secure and strong private public-key DB protocol. For instance, Nonce-DH key agreement protocol is a good candidate for Eff-pkDB^p.

Difficulties of having strong privacy: The strong privacy is the hardest privacy notion to achieve in DB protocols. Sending all messages of provers with an IND-CCA secure encryption is not always enough to have a strong privacy. We exemplify our argument as follows: Clearly, Eff-pkDB protocol is still DF-MiM and DH-secure, if we replace the nonce selection by a counter. So, we can make a new version of Eff-pkDB^p based on the counter version of Eff-pkDB where the prover encrypts his public key and the counter by an IND-CCA encryption. However, it does not give strong privacy because when an adversary calls `Corrupt` oracle, he learns the counter of two drawn provers. Since the adversary knows the corresponding secret keys for both of them, he can easily differentiate the drawn provers based on the counter. This attack is not possible in Eff-pkDB^p which uses a nonce instead of a counter because the nonce is in the volatile memory. So, the adversary does not learn it with the `Corrupt` oracle.

4.4.3 Another variant of Eff-pkDB: Eff-pkDB+1

In this section, we give a variant of Eff-pkDB (the same can apply for Eff-pkDB^p). In this variant, by adding one more pass to Eff-pkDB, we can replace the assumption of multi-verifier OT-MiM security in Theorem 4.12 with the assumption OT-MiM-security.

In this way, we require less security on the symmetric DB protocol by having one-more message exchange.

Eff-pkDB+1 is very similar to Eff-pkDB. Differently, V picks and sends a nonce N_V to P before starting the one-pass AKA protocol. In the AKA phase, V runs $A_{N_V}(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N)$ and P runs $B_{N_V}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N)$ to obtain the secret key s. Here, A_{N_V} and B_{N_V} are the algorithms of the AKA protocol but their outputs depend on N_V. The rest is the same as Eff-pkDB.

$$
\begin{array}{lcr}
\underline{V}(\mathsf{sk}_V, \mathsf{pk}_V) & & \underline{P}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V) \\
 & & \\
N_V \leftarrow \{0,1\}^\ell & \xrightarrow{\quad N_V \quad} & N \leftarrow D(1^\ell) \\
A_{N_V}(\mathsf{sk}_{V_2}, \mathsf{pk}_{V_2}, \mathsf{pk}_P, N) \to s & \xleftarrow{\quad \mathsf{pk}_P, N \quad} & B_{N_V}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_{V_2}, N) \to s \\
 & \xleftrightarrow{\quad \mathsf{symDB}(s) \quad} & \\
 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure 4.5 – Eff-pkDB+1

Theorem 4.15. *Assuming the one-pass AKA* $(\mathsf{Gen}_V, \mathsf{Gen}_P, A_{N_V}, B_{N_V}, D)$ *is D-AKA secure for all fixed* $N_V \in \{0,1\}^s$ *and* symDB *is* ***one time MiM-secure*** *then Eff-pkDB+1 is MiM-secure.*

Proof. Γ_0 : The adversary plays MiM-game of Eff-pkDB+1 with the prover instances and the verifier instances where one of them is the distinguished verifier $\mathcal{V}$.

Γ_1 : We reduce Γ_0 to the game Γ_1 where at most one prover instance and $\mathcal{V}$ see the same pair (N_P, N_V). Because of the D-AKA security, $D(1^\ell)$ guarantees that the repetition of N_P is negligible, and N_V is picked randomly. So, $p_1 - p_0$ is negligible.

We can reduce Γ_1 to the game Γ_3 in Theorem 4.12 with using the similar reductions.

Now, we have at most one distinguished pair $\mathcal{V}$ and P which see (N_V, N_P) and they share a random secret s. Therefore, we are in OT-MiM game. As symDB is OT-MiM secure, p_1 is negligible.

We deduce that Eff-pkDB+1 is MiM-secure because p_0 is negligible. □

We give this variant separately because this version have one more round and a computation on the prover side related to N_V which depends on the AKA algorithm. On the other hand, this version can use less secure symmetric DB protocol to have the same level of security with Eff-pkDB.

4.4.4 Simp-pkDB

We construct another public-key DB protocol Simp-pkDB in Figure 4.6 which is as efficient as Eff-pkDB and does not require key setup for the verifier algorithm. The key setup algorithm $\mathcal{K}_P$ is the key generation algorithm of an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$.

In Simp-pkDB, the prover P randomly selects a nonce $N \in \{0,1\}^\ell$ and sends it to the verifier together with pk_P. Then, V selects a secret $s \in \{0,1\}^\ell$, encrypts it and N with the public key pk_P and sends the encryption e to P. After receiving e, P decrypts it with its secret key sk_P and obtains s,N. If N is the nonce of P, then they run a symmetric DB symDB with using s as a secret key.

$$
\begin{array}{lcr}
\underline{V} & & \underline{P(\mathsf{pk_P},\mathsf{sk_P})} \\
\text{pick } s \in \{0,1\}^\ell & \xleftarrow{\quad N,\mathsf{pk}_P \quad} & \text{pick } N \in \{0,1\}^\ell \\
e = \mathsf{Enc}_{\mathsf{pk}_P}(s||N) & \xrightarrow{\quad e \quad} & s,N = \mathsf{Dec}_{\mathsf{sk}_P}(e) \\
 & & \mathsf{Verify}(N) \\
 & \xleftrightarrow{\quad \mathsf{symDB}(s) \quad} & \\
 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure 4.6 – Simp-pkDB

We show that this protocol is MiM-secure but not DH-secure. P in Simp-pkDB requires only one operation which is IND-CCA decryption.

Theorem 4.16. *If symDB is DF-secure then Simp-pkDB is DF-secure.*

We can easily reduce the DF-game of Simp-pkDB to the DF game of symDB.

Theorem 4.17. *If symDB is one-time MiM-secure and the encryption scheme is IND-CCA secure then Simp-pkDB is MiM-secure.*

Proof. Γ_0 : Adversary plays the MiM game with the verifier instances and the prover instances. Let's assume that the number of prover instances is t where t is polynomially bounded.

Let s, pk_P, N and e be the values seen by the distinguished instance $\mathcal{V}$ of the verifier. Here, $e = \mathsf{Enc}_{\mathsf{pk_P}}(s||N)$. We group the prover's instances as follows:

1. The provers seeing N and e,
2. The provers seeing e but another nonce N'.
3. The provers not seeing e (see a ciphertext e' which is not e).

   The probability that an adversary succeeds in Γ_0 is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where the first group has up to one prover instance P (matching prover). The probability that more than one prover picks the same N is bounded by $\binom{k}{2}2^{-\ell}$ which is negligible. So, $p_1 - p_0$ is negligible.

Γ_2 : We reduce Γ_1 to Γ_2 where the matching P receives e after $\mathcal{V}$ has released e. This means that e which is the encryption of $s||N$ is only sent by the verifier. In Γ_1, the probability that $\mathcal{V}$ selects s after P has received e so that $\mathsf{Dec}_{\mathsf{sk}}(e) = s,N$ is $\frac{1}{2^\ell}$ which means that $p_2 - p_1$ is negligible.

Γ_3 : We reduce Γ_2 to Γ_3 where the prover instances are simulated as below:

The prover instance P, after receiving e, runs $\mathsf{symDB}(s)$ without decrypting e. As e was released before, the value of s is already defined. The prover instances in the second group, abort the protocol after receiving e. The prover instances in the third group, call decryption oracle $\mathsf{Dec}_{\mathsf{sk}}(.)$ after receiving e' and check if the nonce is the same nonce that was chosen by them. Then, they run $\mathsf{symDB}(s')$ with s' obtained from the decryption oracle.

The simulation gives the identical result so the success probabilities in Γ_3 and Γ_2 are the same.

Γ_4 : We reduce Γ_3 to Γ_4. We simulate $\mathcal{V}$ in Γ_4. The simulation of $\mathcal{V}$, after selecting s, encrypts a random message instead of $s||N$.

Γ_3 and Γ_4 are indistinguishable because of the IND-CCA security of the encryption scheme. We construct an adversary $\mathcal{B}$ playing IND-CCA game and simulating MiM game against the adversary $\mathcal{A}$.

$\mathcal{B}$ receives pk_P from the IND-CCA game challenger and then $\mathcal{B}$ forwards it to $\mathcal{A}$. Firstly, $\mathcal{B}$ picks $N,s \in \{0,1\}^\ell \times \{0,1\}^\ell$ and $r \in \{0,1\}^{2\ell}$ and lets $m_0 = s||N, m_1 = r$. Then, he sends m_0 and m_1 to IND-CCA game challenger and receives the response e_b where $e_b = \mathsf{Enc}_{\mathsf{pk_P}}(m_0)$ or $\mathsf{Enc}_{\mathsf{pk_P}}(m_1)$. If $\mathcal{A}$ interacts with $\mathcal{V}$ then $\mathcal{B}$ sends e_b, if $\mathcal{A}$ interacts with P, then $\mathcal{B}$ sends N. For the simulation of other prover instances, $\mathcal{B}$ sends the encryptions e' to IND-CCA game challenger and receives decryption of e'. In the end, if $\mathcal{A}$ succeeds then $\mathcal{B}$ outputs 0, otherwise, he outputs 1. If $\mathcal{A}$ succeeds given $b = 0$, then it means that he succeeds Γ_3 and if $\mathcal{A}$ succeeds given $b = 1$ then it means that he succeeds Γ_4. Therefore, we have the following success probability of $\mathcal{B}$.

$$\mathsf{Adv}(\mathcal{B}) = \mathsf{Pr}[\mathcal{B} \to 1|b = 0] + \mathsf{Pr}[\mathcal{B} \to 1|b = 1] = p_3 - p_4$$

As we know that the advantage of $\mathcal{B}$ is negligible, we can deduce that $p_3 - p_4$ is negligible.

Γ_5 : In Γ_5, we have at most one prover and verifier instance and they both run $\mathsf{symDB}(s)$ with the same and fresh random s. Γ_4 and Γ_5 work the same. So. $p_4 = p_5$. The success probability p_5 of Γ_5 is negligible because of the OT-MiM security of symDB. As as result, we can conclude that p_0 is negligible.

□

DH-Security: Simp-pkDB is not secure against DH because of the attack in Figure 4.7. In this attack, the malicious and far away prover P uses honest and close prover P' so that in the end V accepts P.

The attack starts after seeing the nonce N that is picked by P'. P sends (pk_P, N) to V where $\mathsf{pk_P}$ is the public key of P. Then, V encrypts $s||N$ with $\mathsf{pk_P}$ and sends it to P. P decrypts e with his own secret key $\mathsf{sk_P}$ and behaves as if he is the verifier and sends the encryption $e' = \mathsf{Enc}_{\mathsf{pk_{P'}}}(s||N)$ where $\mathsf{pk_{P'}}$ is the public key of P'. As e' is valid encryption for P', it continues by executing $\mathsf{symDB}(s)$ with V. At the end of the protocol, V accepts P. P' is used by P only to be able to pass the challenge phase of $\mathsf{symDB}(s)$ protocol.

$$
\begin{array}{lcccr}
\underline{V(\mathsf{pk_P})} & & \underline{\mathsf{P}(\mathsf{pk_P},\mathsf{sk_P})} & & \underline{\mathsf{P}'(\mathsf{pk_{P'}},\mathsf{sk_{P'}})} \\
\text{pick } s & & & \xleftarrow{N,\mathsf{pk_{P'}}} & \text{pick } N \\
e = \mathsf{Enc}_{\mathsf{pk_P}}(s||N) & \xleftarrow{N,\mathsf{pk}_P} & & & \\
& \xrightarrow{e} & s,N = \mathsf{Dec}_{\mathsf{sk_P}}(e) & & \\
& & e' = \mathsf{Enc}_{\mathsf{pk_{P'}}}(s||N) & & \\
& & & \xrightarrow{e'} & s,N = \mathsf{Dec}_{\mathsf{sk_{P'}}}(e') \\
& & & & \mathsf{Verify}(N) \\
& \multicolumn{3}{c}{\xleftrightarrow{\qquad\qquad\mathsf{symDB}(s)\qquad\qquad}} & \\
& \xrightarrow{\mathsf{Out}_V} & & &
\end{array}
$$

Figure 4.7 – DH attack on Simp-pkDB.

We can have weak private variant of Simp-pkDB. In this variant (Simp-pkDB^p), V has secret/public key pair $(\mathsf{sk}_V,\mathsf{pk}_V)$ which is the key pair of another encryption scheme $(\mathsf{Enc}',\mathsf{Dec}')$. P sends pk_P,N as in Simp-pkDB but differently by encrypting them with pk_V. The rest is the same.

Theorem 4.18 (Weak privacy of Simp-pkDB^p)**.** *Assuming that the encryption scheme with* $(\mathsf{Enc}',\mathsf{Dec}')$ *is IND-CCA secure and the encryption scheme with* $(\mathsf{Enc},\mathsf{Dec})$ *is IND-CCA and IK-CPA [BBDP01] secure (Definition 2.19), then Simp-pkDB^p is weak private (Definition 2.14).*

Proof. Γ_0 : The adversary $\mathcal{A}$ plays the weak-privacy game. The success probability of $\mathcal{A}$ is p_0.

Γ_1 : We reduce Γ_0 to Γ_1 where the verifiers do not decrypt (with Dec') any encryptions sent by the provers and the provers do not decrypt (with Dec) the encryptions generated by the verifiers. Instead, they directly use the values inside the encryption. Because of the correctness of both encryption schemes $p_1 = p_0$.

Γ_2 : We reduce Γ_1 to Γ_2 where all provers encrypt (with Enc') a random value instead of pk_P,N and all verifiers encrypt (with Enc) a random value instead of $(s||N)$. Note that the change on the encryption is indistinguishable by an adversary as it does not know sk_P (we prove here weak privacy). Thanks to the IND-CCA security of the encryption schemes $p_1 - p_2$ is negligible.

Γ_3 : We reduce Γ_2 to Γ_3 where the prover does not decrypt (with Dec) the encryptions e generated by the adversaries and it aborts. As N has never been used, the probability that $\mathcal{A}$ sends a valid encryption of N is negligible. Therefore, $p_3 - p_2$ is negligible. Remark that in Γ_3, $\mathsf{Dec}_{\mathsf{sk}_P}$ has never been used.

Γ_4 : We reduce Γ_3 to Γ_4 where the prover replaces pk_P by a freshly generated public-key. (that V uses if the encryption that P sends is correctly forwarded). The only visible change from Γ_3 is that now e (sent by the verifier) is encrypted using a new key. Because of IK-CPA security of the encryption scheme (with $\mathsf{Enc}, \mathsf{Dec}$), $p_4 - p_3$ is negligible.

Now, in Γ_4, no identity is used by the verifiers and the provers, so adversary succeeds Γ_4 with $\frac{1}{2}$ probability. Therefore, $p_0 - \frac{1}{2}$ is negligible. □

Simp-pkDB^p is not strong private due to the following attack: assume that an adversary corrupts a prover P and learns sk_P. Later, he can decrypt all encryptions (e) sent by the verifier with sk_P. If e is sent to P, then it means the adversary learns the challenges and responses. When these challenges and responses become known during symDB, the adversary can identify P.

Simp-pkDB^p is not as good as Eff-pkDB^p, privDB [Vau15c] which have higher security level. Its only advantage is that it does not require a key setup for the verifier. We give Simp-pkDB^p to show that we can obtain some level of privacy and security which can be converted into higher privacy and security level in a different model as we discuss in Chapter 5.

4.5 Conclusion

Our main purpose in this chapter was to design an efficient and a secure public-key DB protocol. Therefore, we began by designing a new AKA security model which we can use to construct a public-key DB. The level of security of AKA we defined is less strong than existing ones [LLM07]. However, as our main purpose is efficiency, we define the least security level that we want from an AKA protocol to achieve a good security (MiM, DF, DH- security) on our public-key DB protocols. Finally, we designed a public-key DB protocol Eff-pkDB which is secure against DF, MiM and DH with using an AKA protocol and a symmetric-DB protocol. We did not consider privacy in this one because privacy is not the main concern of some applications. Eff-pkDB is one of the most efficient public key DB protocols compared to the previous ones (See Table 4.2). Besides, we added strong privacy to the Eff-pkDB protocol and obtained Eff-pkDB^p. We achieved this by adding one public-key IND-CCA secure encryption. In this case, the protocol is not as efficient as before but still one of the most efficient ones with the same security and privacy properties.

We also discussed about a variant of Eff-pkDB and Eff-pkDB^p: the one (Eff-pkDB+1) that requires less security on the symmetric DB. We suggested another public-key DB protocol Simp-pkDB which is quite efficient and do not require any key setup for verifiers. It is DF and MiM-secure but not DH-secure.

Now, we prove our claim about the prover efficiency of Eff-pkDB, Eff-pkDB and Simp-pkDB.

Comparison: In Table 4.2, we give the security properties of existing public-key DB protocols along with the number of computations done on the prover side. We use the number of elliptic curve multiplications and hashing as a metric in our efficiency

Protocol	Security	Privacy	PK operations	Number of Computations
Brands-Chaum [BC93]	MiM, DF	No Privacy	1 commitment, 1 signature	1 EC multiplication, 2 hashings, 1 mapping, 1 modular inversion, 1 random string selection
HPO [HPO13]	MiM, DF	Weak Private		4 EC multiplications, 2 random string selections, 2 mappings
PrivDB [Vau15c]	MiM, DF, DH	Strong Private	1 signature, 1 IND-CCA encryption	3 EC multiplications, 2 hashings, 2 random string selection, 1 modular inversion, 1 mapping, 1 symmetric key encryption, 1 MAC
TREAD [ABG$^+$17]	MiM,DF,DH	Strong Private	1 signature, 1 IND-CCA encryption	3 EC multiplications, 2 hashings, 2 random string selection, 1 modular inversion, 1 mapping, 1 symmetric key encryption, 1 MAC
Simp-pkDB	MiM, DF	No Privacy	1 decryption	1 EC multiplication, 1 hashing, 1 symmetric key decryption, MAC
Eff-pkDB	MiM, DF, DH	No privacy	1 D-AKA secure KA protocol	1 EC multiplication, 1 hashing, 1 random string selection
Eff-pkDB^p	MiM, DF, DH	Strong Private	1 IND-CCA Encryption, 1 D-AKA secure KA protocol	3 EC multiplications, 2 hashings, 2 random string selections, 1 symmetric key encryption, 1 MAC

Table 4.2 – The efficiency and security of existing public-key DB protocols.

analysis. We exclude GOR, ProProx and eProProx (in Appendix A.3 and A.4) as they clearly require a lot more computation than the other public-key DB protocols.

In our counting for the number of computations in Table 4.2, 1 commitment is counted as 1 hashing operation. For the signature, we prefer an efficient and existentially unforgeable under chosen-message attacks resistant signature scheme ECDSA [JMV01]. ECDSA requires 1 EC multiplication, 1 mapping, 1 hashing, 1 modular inversion and 1 random string selection. For the IND-CCA encryption scheme, we use ECIES [Sho01] which requires 2 EC multiplications, 1 KDF, 1 symmetric key encryption, 1 MAC and 1 random string selection. For the D-AKA secure key agreement protocol, we use Nonce-DH which requires 1 EC multiplication, 1 hashing and 1 random string selection.

We first compare the protocols considering the security and the efficiency trade-off. Eff-pkDB and Simp-pkDB are the most efficient ones. However, Simp-pkDB is secure only against MiM and DF. After Eff-pkDB, the second most efficient protocol is Brands-Chaum protocol [BC93] (Appendix A.1) but this protocol is only secure against MiM and DF while Eff-pkDB is secure against DH as well.

Now, we compare the protocols considering security, privacy and efficiency trade-off. In this case, HPO requires 4 EC multiplications while PrivDB, TREAD and Eff-pkDB^p require 3 EC multiplications and 1 hashing. Hashing is more efficient than elliptic curve multiplication so it looks like PrivDB and Eff-pkDB^p are more efficient. However, HPO has an advantage in efficiency if it is used in a dedicated hardware allowing only EC operations. On the other hand, Eff-pkDB^p and PrivDB are secure against MiM, DF, DH and strong private while HPO is only MiM and DF secure and only private.

Eff-pkDB^p, TREAD and PrivDB have the same security and privacy properties and almost the same efficiency level. However, if we analyze the efficiency with more metrics, we see that PrivDB and TREAD require 1 extra modular inversion and 1 mapping. So, Eff-pkDB^p is slightly more efficient. More importantly, Eff-pkDB^p has lighter version Eff-pkDB which can be used efficiently in the applications which do not need privacy.

Chapter 5

Formal Analysis of Distance Bounding with Secure Hardware

Until this chapter, we mainly covered MiM, DF and DH secure constructions. However, we did not fully concentrate on TF-security because of the problems on achieving it. We analyze it in this chapter. Formally, the TF-security prevents against malicious and far-away provers which try to make the verifier accept the access of himself with the help of an adversary who may be closer. Clearly, the strongest security notion in the DB world is the resistance to TF. So, if we can construct a DB protocol that is secure against TF, then the DB protocol will be secure against MiM, DF and DH. However, it is not possible to achieve the TF-security because of a trivial attack: the malicious prover gives his secret (key) to a close adversary, and the adversary authenticates on behalf of the malicious prover by running the protocol. To achieve the TF-security, the trivial attack is artificially excluded from the TF model in the literature by assuming that malicious provers would never share their keys (in this chapter, we call this weaker version "TF'-security"). Beyond being weak, we cannot adapt TF'-security as an all-in-one security notion because no connection between TF'-security and MiM, DF or DH security can be established. Because of this disconnection, all DB protocols require separate security analysis for each of them.

The content of this chapter was published in ACNS18 [KV18a].

Related Works:

The only public-key DB protocols that are secure against all of them (MiM, DF, DH, TF') are ProProx [Vau15d], its variant eProProx [Vau15a] and TREAD [ABG+17]. There are also a few symmetric key DB protocols [BMV13a, BMV15, BMV13b, FO13, Vau13, BV14] that are secure against all (MiM, DF, DH, TF'). Some important distance bounding protocols [BC93, BB05, ČBH03, Han05, RNTS07, SP07, KV16] are all vulnerable to TF'. The protocol by Bultel et al. [BGG+16] is TF'-secure thanks to a 'cheat option' (as explained below) but it is not DH-secure since it aims for anonymity against verifier.

Moreover, the formal definition of TF'-security is controversial. The TF'-security definition of Dürholz et al. [DFKO11] allows treatment of the partial disclosure of the secret key. Essentially, the TF' security in this definition implies that any information forwarded to a close-by adversary would allow another adversary to later pass, without a help of the prover, with the same probability. Fischlin and Onete [FO13] adapted the Swiss-Knife protocol [KAK+08] to have this definition. However, it was proven that this technique weakens Swiss-Knife for MiM-security [Vau13]. Clearly, it is not reasonable to weaken the most relevant security to protect it against the least relevant one. There are also extractor based TF'-security definitions [BV14, Vau15d, Vau13] stronger than the definition of Dürholz et al. model [DFKO11]. However, all TF'-security definitions are constructed with the assumption that the malicious prover do not reveal any secret key related information. This assumption can be considered **weak and not realistic**. Recently, Ahmadi and Safavi-Naini [ASN17] showed that some existing TF'-secure protocols become insecure when the malicious prover and the verifier use a directional antenna. In short, none of the models in the literature fully covers TF.

Apparently, there is no way of achieving TF-security without hiding the secret key from the prover. This intuitive idea has been noticed [SP05, BR04, ABK+09, ABK+11], but never formally defined. A natural question to ask here is whether this idea really prevents TF. The answer is "yes and no" because hiding the key is necessary but not sufficient. We can give an example protocol where the prover never learns the key but the protocol is still vulnerable to TF-attacks.

In a nutshell, state-of-the-art DB results say that TF-security is not possible in the existing models of DB and it could be possible by hiding the key but this is not enough. However, it is still not formally noted how it can be achievable. Therefore, in this chapter, we define a new formal model where constructing TF-secure protocols is possible.

5.1 Our Contribution

Our formal model for DB, which we call secure hardware model (SHM), provides a solution to all DB related problems that we mention. We denote the two-algorithm (Prover and Verifier) DB corresponding to the classical DB in the literature as "plain model" (PM) [BC93, DFKO11, BMV13a, BV14, Vau15c] that we cover in Section 2.2.1. In the SHM, we have another entity called "Hardware" that is always honest and only communicates with its holder (the prover). Mainly, this hardware runs some part of the prover algorithm honestly and neither a malicious prover nor an adversary can corrupt it. In the real world, we can realize our new entity as e.g. tamper-resistant modules in smart-cards. In more detail, *our contribution* in this chapter is the following:

- We define a new type of DB with three algorithms (V, P, H): verifier, prover, hardware. Then, we design a communication and adversarial model for three-algorithm DB which we call secure hardware model (SHM). In SHM, it is possible to have

TF-secure DB protocols without excluding trivial attacks. We give a new security definition in SHM for a three-algorithm DB. In this security definition, achieving TF-security means achieving MiM, DF and DH-security. So, we obtain an all-in-one definition.

- We obtain a convincing model for TF based on SHM. We show that the TF-security of (V,P,H) in SHM is equivalent to the MiM-security of (V,H) in PM where H in PM corresponds to the prover algorithm. This result implies that P **plays no role in security but only in the correctness of the protocol to have TF-security**.

- We establish security relations between PM and SHM. We show that the MiM-security in SHM and the MiM-security in PM are equivalent when we take as a prover algorithm in PM the union P^H of the prover P and the hardware H in SHM. Additionally, we show that a MiM-secure DB protocol in PM can be converted into a fully-secure DB protocol in SHM. This result shows that if we have only a MiM-secure DB protocol in PM, **we can easily construct an efficient DB protocol secure against all threats in SHM**.

- We define a strong privacy notion of DB in SHM.

- We construct a symmetric DB protocol **MiM-symDB which is the most efficient optimally secure MiM-secure protocol in PM (in terms of computation and number of rounds) among the protocols with binary challenges and responses.** Then, we convert it into a DB protocol in SHM (Full-symDB^H) and obtain the most efficient symmetric DB protocol secure against all threats and achieving optimal security bounds.

We underline that the **only assumption on the secure hardware is that it is honest** which means that it runs the specified algorithm only. By doing so, we give a model here where the TF-security is achievable. Avoine et al. [ABK$^+$09, ABK$^+$11] considered a similar model called “black box model”. In their model, they assume that the prover cannot observe or tamper with the execution of the algorithm. Differently, in SHM, we consider two algorithms a prover and a hardware algorithm and only the hardware algorithm cannot be tampered. The black box model can be seen as a variant of our model where the prover algorithm is dummy which only relays the messages between the hardware and the verifier. In addition, in the black box model, Avoine et al. [ABK$^+$09, ABK$^+$11] only consider TF’-security instead of TF-security.

One may argue that our assumption on secure hardware is too strong for the real world applications. For example, in the real world, if the secure hardware is implemented using a tamper-resistant hardware, it is always possible that a side-channel attack will break our assumption. However, we believe that relying on our assumption is more reasonable than relying on some adversarial intention (e.g., that the adversary never shares his secret). We can never prevent a TF-adversary to share his secret-key, but we can construct a strong tamper-resistant hardware which requires very expensive equipments to be tampered. Besides, MiM-security would be preserved even if the tamper resistance assumption is broken.

Structure of the Chapter: In Section 5.2, we give the formal definitions of SHM and its security and privacy definitions. We also prove some security implications in SHM. In Section 5.3, we introduce the MiM-symDB protocol in PM and adapt it to SHM. In Section 5.4, we show how we can achieve full security in public-key DB in SHM with Eff-pkDB^p and Simp-pkDB^p. We conclude the chapter in Section 5.5.

5.2 Secure Hardware Model

We first give the formal definitions of SHM and security in this model. Then, we provide some security relations related to PM and SHM.

5.2.1 Definitions

Parties of a DB protocol are a prover and a verifier [BC93] as detailed in previous chapters. However, we define a new version of it called three-algorithm (symmetric or public-key) DB where the algorithms are prover, verifier, and hardware.

Definition 5.1 (Three-Algorithm Symmetric DB)**.** *Three-algorithm symmetric DB is a probabilistic polynomial-time (PPT) protocol. It consists of a tuple* $(\mathcal{K}, V, P, B, H)$ *where* $\mathcal{K}$ *is the key generation algorithm,* P *is the proving algorithm,* H *is the hardware algorithm,* V *is the verifying algorithm and* B *is the distance bound. The input of* V *and* H *is* K *generated by* $\mathcal{K}$*.* P *interacts with* $H(K)$ *and* $V(K)$*. At the end of the protocol,* $V(K)$ *outputs a final message* $\mathsf{Out}_V \in \{0,1\}$*. If* $\mathsf{Out}_V = 1$*, then* V *accepts. If* $\mathsf{Out}_V = 0$*, then* V *rejects.*

In symmetric DB, V knows that it needs to use K (possibly resulting from a prior identification protocol).

Definition 5.2 (Three-Algorithm Public-key DB)**.** *Three-algorithm public key distance bounding is a PPT protocol. It consists of a tuple* $(\mathcal{K}_P, \mathcal{K}_V, V, P, B, H)$ *where* $(\mathcal{K}_P, \mathcal{K}_V)$ *are the key generation algorithms of* P *and* V*, respectively. The output of* $\mathcal{K}_P$ *is a secret/public key pair* $(\mathsf{sk}_P, \mathsf{pk}_P)$ *and the output of* $\mathcal{K}_V$ *is a secret/public key pair* $(\mathsf{sk}_V, \mathsf{pk}_V)$*.* V *is the verifying algorithm with the input* $(\mathsf{sk}_V, \mathsf{pk}_V)$*,* P *is the proving algorithm with the input* $(\mathsf{pk}_P, \mathsf{pk}_V)$ *and* H *is the hardware algorithm with the input* $(\mathsf{sk}_P, \mathsf{pk}_P)$*.* B *is the distance bound.* P *interacts with* $H(\mathsf{sk}_P, \mathsf{pk}_P)$ *and* $V(\mathsf{sk}_V, \mathsf{pk}_V)$*. At the end of the protocol,* $V(\mathsf{sk}_V, \mathsf{pk}_V)$ *outputs a final message* $\mathsf{Out}_V \in \{0,1\}$ *and has* pk_P *as a private output. If* $\mathsf{Out}_V = 1$*, then* V *accepts. If* $\mathsf{Out}_V = 0$*, then* V *rejects.*

This definition assumes a priori identification of pk_V for P.

Definition 5.3 (Correctness of DB). *A public-key (resp. symmetric) DB protocol is correct if and only if under an honest execution, whenever the distance between P and V is at most B, V always outputs $\mathsf{Out}_V = 1$ and pk_P (resp. $\emptyset$).*

In all the definitions below, verifiers, provers, and hardware are the parties running the algorithms V, P and H, respectively. The parties can move and run their algorithms multiple times. Each new execution of a party's algorithm is an **instance** of this party.

Classical DB in literature is very similar to three-algorithm DB with the following differences: no H algorithm exists and the input of P in public-key and symmetric DB is $(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ and K, respectively. The plain model is the model corresponding to the classical DB (See Section 2.2.1 for details).

The secure hardware model is the model corresponding to three-algorithm DB: P, V and H.

Secure Hardware Model (SHM): Parties of SHM are provers, secure hardware, verifiers and other actors. SHM includes all the characteristics of PM given in Section 2.2.1 and the additional ones:

- Secure hardware are honest parties.
- Each prover possesses its own secure hardware.
- The secure hardware of an honest prover can only communicate with its prover and they are both at the same location.

In the rest of the chapter, whenever we say "a distance bounding protocol in SHM", it refers to the three-algorithm DB.

Remark that as secure hardware are honest parties, they always run their assigned algorithms even if malicious provers hold them. They should be taken as a subroutine of a prover algorithm running on a secure enclave where the prover can never change or interfere it.

Now, we give our security definition for a DB protocol in SHM. The definition covers distance fraud, mafia fraud (MiM), distance hijacking and terrorist fraud which are the threat models in PM.

Definition 5.4 (Security in SHM). *Consider a public-key DB. The game consists of a verifier and provers $P_1, P_2, ..., P_t$ with their corresponding hardware $H_1, H_2, ..., H_t$. It begins by running the key setup algorithm $\mathcal{K}_V$ outputting $(\mathsf{sk}_V, \mathsf{pk}_V)$ for V and $\mathcal{K}_P$ outputting $(\mathsf{sk}_{P_i}, \mathsf{pk}_{P_i})$ for H_i. The game consists of instances of the verifier, provers, hardware and actors. $\mathcal{V}$ is a distinguished instance of the verifier. One prover (let's denote P) is the target prover. The winning condition of the game is $\mathcal{V}$ outputs $\mathsf{Out}_V = 1$ and privately pk_P (public key of P) if no close instance of P's hardware exists during the execution of $\mathcal{V}$.*

- *The DB protocol is MiM-secure if the winning probability is always negligible whenever P is honest*[1]*.*
- *The DB protocol is DF-secure if the winning probability is always negligible whenever there is no instance of any party close to $\mathcal{V}$.*
- *The DB protocol is DH-secure if the winning probability is always negligible whenever all close instances are honest provers other than P and their hardware.*
- *The DB protocol is TF-secure if the winning probability is always negligible.*

The same security definition holds for a symmetric DB where we replace $\mathcal{K}_V$ and $\mathcal{K}_P$ with $\mathcal{K}$ and $\mathsf{sk}_{P_i}/\mathsf{pk}_{P_i}$ with K_i.

Without loss of generality, we can consider all other actors as adversaries.

It is clear that TF-security implies DF-security, MiM-security, and DH-security. So, we have an all-in-one security notion in SHM. Hence, we say **"secure" instead of "TF-secure"** in SHM.

Security in PM: In PM, there is always a trivial TF-attack in which a malicious prover can give his secret key to another malicious party so that the party authenticates the prover while it is far-away. So, TF-security is not possible in PM. Clearly, this trivial attack is preventable in SHM if we can assure that H never leaks K.

Note that we do not consider the weaker version of TF-security [DFKO11, KAK$^+$08, Vau13] (TF'-security) which artificially excludes trivial attack. So, when we refer to TF-security in PM, we indeed refer to an impossible-to-achieve notion.

Notations:

$\boldsymbol{P_{dum}}$ is a dummy prover algorithm in SHM which only relays the messages between the outside world and H without even using any of its input. Remark that if the prover who should run P_{dum} is malicious, then it can still play with its hardware or other parties maliciously.

$\boldsymbol{P^H}$ is the algorithm which is constructed from joining P and H in SHM. More precisely, P^H runs P and instead of interacting with H, it executes the same computation that H would do if P had interacted. Therefore, $\boldsymbol{P^H_{dum}}$ is actually the hardware algorithm H.

5.2.2 Security Results

We give some security relations between a DB protocol in PM and SHM.

Theorem 5.5 (MiM in SHM $\Rightarrow$ MiM in PM)**.** *Let $DB = (\mathcal{K}, V, P, B, H)$ be a symmetric-key DB protocol in SHM. We define a DB protocol $DB' = (\mathcal{K}, V, P^H, B)$ in PM. If DB is MiM-secure then DB' is MiM-secure.*

The same holds with public-key DB.

[1] Recall that it implies that H communicates with P only and that they are at the same location.

The proof is trivial by adding a hardware to every honest prover at the same location: A MiM-game against DB' becomes a MiM-game against DB.

Theorem 5.6 (MiM-security in PM with P^H_{dum} $\Leftrightarrow$ Security in SHM)**.** *Let $DB = (\mathcal{K}, V, P, B, H)$ be a symmetric DB in SHM and and $DB' = (\mathcal{K}, V, P^H_{dum}, B)$ be a symmetric-key DB in PM where superscript of P^H_{dum} in DB' corresponds H of DB. DB' is MiM secure in PM if and only if DB is TF-secure in SHM.*

Here, the prover algorithm of DB' is just H because $P^H_{dum} \equiv H$.

Surprisingly, Theorem 5.6 does not depend on the prover algorithm P of DB. Note that DB' in Theorem 5.6 is not a correct DB protocol in general if $P \neq P_{dum}$ as the algorithm P disappeared. However, we can still consider MiM-security for DB' without correctness.

Proof. ($\Rightarrow$) Consider a TF-game against DB in SHM. We run this game against DB' in PM by simulating the secure hardware H of DB with the prover P^H_{dum} of DB' and simulating the prover P in SHM with a malicious actor in PM (it is possible because P in SHM, does not have any secret key as an input). Then, we obtain MiM-game of DB' with the same probability of success.

($\Leftarrow$) If $\mathcal{A}$ wins the MiM-game of DB', then a TF adversary runs $\mathcal{A}$ and wins the TF-game for DB.

□

Remark that it is not possible to prove "MiM-security of $DB' = (\mathcal{K}, V, P^H, B) \Leftrightarrow$ security of $DB = (\mathcal{K}, V, P, B, H)$" where P in DB' is not necessarily P_{dum} because we could not simulate H and P in "$\Rightarrow$" case of the proof in Theorem 5.6. A counterexample which elucidates this can be seen in Section 5.3. In this counterexample, we have a MiM-secure protocol in PM and a conversion of it in SHM without P_{dum}. Its conversion is not secure in SHM even though P does not learn any key related information.

Clearly, having a secure hardware running whole algorithm without its prover's effect on the security is a trivial solution to have a TF-security. However, we show here that **it does not always work when the prover is active.** This result does not mean that prover should not do any computation to have TF-security. Actually, in our TF-secure protocols in Section 5.4, the prover algorithm in SHM still executes some part of the algorithm P^H in PM but it does not have any effect on the security of the protocol (as it can be seen in their security proofs Theorem 5.12 and Theorem 5.14).

Some more results of Theorem 5.6:

- We can conclude if $DB' = (\mathcal{K}, V, P^H_{dum}, B)$ is **MiM-secure** and correct DB protocol, then we can construct a **secure** DB protocol $DB = (\mathcal{K}, V, P, B, H)$ in SHM for any algorithm P. DB is further correct when $P = P_{dum}$.

- In order to prove security of $DB = (\mathcal{K}, V, P, B, H)$ in SHM, it is enough to prove MiM-security of $DB' = (\mathcal{K}, V, P^H_{dum}, B)$ in PM.

- **MiM security and security of a DB protocol $\boldsymbol{DB} = (\mathcal{K}, \boldsymbol{V}, \boldsymbol{P}, \boldsymbol{B}, H)$ in SHM are equivalent if $\boldsymbol{P} = \boldsymbol{P}_{dum}$** due to Theorem 1 and Theorem 2. Note that this result may not hold without P_{dum} .

In Figure 5.1, we give the security (non)-implications in SHM and PM. The proof of these (non)-implications are in Appendix C. In Figure 5.2, we give the same for SHM when the prover is P_{dum}. In this case, **the full security is equivalent to MiM-security**. The rest of the (non)-implications in Figure 5.2 can be proven as in Appendix C.

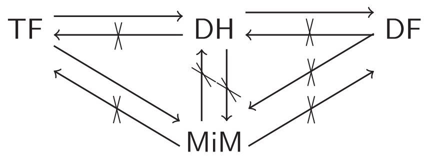

Figure 5.1 – Security implications of DB protocols in PM and SHM. TF-security implies all of them, DH-security implies DF security and no relation exists between MiM and DH (also DF).

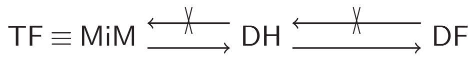

Figure 5.2 – Security implications in SHM with the prover P_{dum}. TF-security and MiM security are equivalent in SHM with P_{dum}. The relations between DF, DH and MiM are the same as in Figure 5.1.

5.2.3 Privacy

In strong-privacy definition of PM, the adversary can corrupt the provers and learn the secrets. However, the hardware in SHM is honest by nature. So, it cannot be corrupted. Hence, we define semi-strong privacy with no such corruption. Achieving semi-strong privacy in a DB protocol is good enough assuming that the hardware is tamper-resistant. Nevertheless, we also allow corruption of hardware in order to define the strong privacy notion.

Definition 5.7 (Privacy in SHM)**.** *The privacy game in SHM for a public-key distance bounding $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B, H)$ with a bit $b \in \{0,1\}$ is the following: The game runs the key setup algorithms $\mathcal{K}_P(1^\ell)$ for a number t provers and $\mathcal{K}_V(1^\ell)$ for the verifier. Then, it lets the adversary $\mathcal{A}$ play the game $\mathsf{shm}-\mathsf{Priv}^{\mathcal{O}}_{b,\mathcal{A}}(\ell)$ with the oracles in Definition 2.14 with a change in the corrupt oracle and the oracle* `SendH`*:*

- `SendH`$(vtag, m)$*: It sends the message m to the drawn prover's hardware and returns the response m' of the hardware. If $vtag$ was not drawn or was released, nothing happens.*

- `Corrupt`(P_i) *: It returns the current state of P_i and its hardware H_i. Current state includes all values in P_i's and H_i's current memories. It does not include volatile memory.*

In the end, the adversary outputs b'. If $b' = b$, the adversary wins. Otherwise, it loses.

We say a DB protocol in SHM is ***strong private*** *if the advantage of the adversary in this game is bounded by a negligible probability. We say a DB protocol in SHM is* ***semi-strong private*** *if the advantage of the adversary in a version of this game, where the corruption only lets the adversary communicate with the hardware non-anonymously, is bounded by a negligible probability.*

In semi-strong privacy, even though we do not allow corruption of hardware, we let semi-strong corruption occur by allowing interaction with the secure hardware. In SHM, we stress that when P interacts with its secure hardware, this interaction remains private.

Hermans et al. [HPVP11] (See Definition 2.14) defined a similar game for the strong privacy of DB in PM. In that game, no hardware exists, so the definition of semi-strong privacy is not considered. Instead, the weak privacy notion exists where no corruption on provers are allowed.

Note that we obtain a notion of strong privacy of $DB = (K,V,P,B,H)$ in SHM which is fully equivalent to the strong privacy of $DB' = (K,V,P^H,B)$ in PM.

5.3 Optimal symmetric DB protocol in SHM

In this section, we show our new protocol MiM-symDB in PM which is only MiM-secure (not DF, DH or TF-secure). We construct a DB at this level of security because having MiM-security in PM is enough to achieve (full) security in SHM as a result of Theorem 5.6. The security bounds of MiM-symDB is very close to optimal security bounds. Its conversion into SHM reaches the same bound as well. As we see in Chapter 3, it is proved [BV14] that an optimal security bound in PM for a MiM-adversary is $(\frac{1}{2})^n$ given that challenges and responses are bits and the challenge phase consists of n rounds. The same bound applies in SHM as well.

We note that using other optimally MiM-secure DB protocols such as DB1, DB2, DB3 (variants of DBopt) [BV14] is reasonable as well to have fully secure DB protocols in SHM. However, these protocols are also secure against DF or TF' in PM which is an overkill as we need only MiM-security. By constructing an optimal MiM-only secure DB in PM, we can save some computations and rounds to make an optimal protocol in SHM.

Notation: When we use H as a superscript in the name of a protocol, it shows that it is in SHM.

MiM-OTDB: First, we describe our MiM-OTDB protocol which is MiM-secure when it is executed *only once*. The prover P and the verifier V share a secret key $s = C||R$. Here, the bits of C correspond to the challenges and the bits of R correspond to the responses. In the challenge phase, in each round i, V sends the challenge $c_i = C[i]$ to P and P sends the response $r_i = R[i]$ to V. If P receives a challenge which is different from $C[i]$, then P does not continue to the protocol. In the verification phase, V checks if the responses are correct and on time. (See Figure 5.3.)

$$
\begin{array}{lcr}
\underline{V}(s) & & \underline{P}(s) \\
s = C||R & & s = C||R \\
 & \textbf{challenge phase} & \\
 & \text{for } i = 0 \text{ to } n & \\
c_i = C[i], \text{ start } \mathsf{timer}_i & \xrightarrow{\quad c_i \quad} & \text{if } c_i \neq C[i], \text{ abort} \\
\text{stop } \mathsf{timer}_i & \xleftarrow{\quad r_i \quad} & \text{otherwise, } r_i = R[i] \\
 & \textbf{verification phase} & \\
\text{check } \mathsf{timer}_i \leq 2B, r_i = R[i] & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure 5.3 – MiM-OTDB

MiM-symDB: Now, we describe MiM-symDB which is constructed on top of MiM-OTDB and optimally MiM secure. The prover P and the verifier V share a secret key s. They use a pseudo random function (PRF) f returning strings of $2n$ bits. P and V exchange the nonces $N_P, N_V \in \{0,1\}^\ell$, respectively, where ℓ is a security parameter. Then, P and V compute $f_s(N_P, N_V)$ which outputs $C||R$. Finally, V and P run MiM-OTDB with using $C||R$ as a key. (See Figure 5.4.)

$$
\begin{array}{lcr}
\underline{V}(s) & & \underline{P}(s) \\
\text{pick } N_V \in \{0,1\}^s & \xleftarrow{\quad N_P \quad} & \text{pick } N_P \in \{0,1\}^s \\
C||R = f_s(N_P, N_V) & \xrightarrow{\quad N_V \quad} & C||R = f_s(N_P, N_V) \\
 & \xleftrightarrow{\mathsf{MiM\text{-}OTDB(C||R)}} &
\end{array}
$$

Figure 5.4 – MiM-symDB

Theorem 5.8 (MiM-security of MiM-symDB)**.** *If f is a secure PRF, then the winning probability of a probabilistic polynomial time (PPT) adversary in a MiM-game of MiM-symDB in PM is at most $\frac{3}{2^{n+1}} + \frac{q^2}{2^{\ell+1}} + \frac{q'^2}{2^{\ell+1}} + \mathsf{Adv}_{PRF}((\varepsilon, K))$. For a PPT game, this is negligible.*

Proof. Γ_0 : It is a MiM-game where P's instances and V's instances with the distinguished instance $\mathcal{V}$ play in PM. The winning probability in Γ_0 is p.

Γ_1 : We reduce Γ_0 to Γ_1 where the nonces of the prover instances and the nonces of the verifier instances do not repeat. The probability that a prover (resp. verifier) instance selects the same nonce with the one of the other prover (resp. verifier) instances is bounded by $\frac{q^2}{2}\frac{1}{2^\ell}$ (resp. $\frac{q'^2}{2}\frac{1}{2^\ell}$). So, the winning probability of Γ_1 is at least $p - \frac{q^2}{2^{s+1}} - \frac{q'^2}{2^{\ell+1}}$.

Γ_2 : We reduce Γ_1 to Γ_2 where $\mathcal{V}$ and the prover's instances replace $f_s(.,.)$ by a random function. Clearly, the winning probability in Γ_2 is at least $p - \frac{q^2}{2^{\ell+1}} - \frac{q'^2}{2^{\ell+1}} - \frac{1}{2^\ell} - \mathsf{Adv}(PRF)$.

In Γ_2, we have a game where at most one prover instance P seeing (N_P, N_V) pair with $\mathcal{V}$ and $C||R$ is completely random meaning that it is independent from N_P and N_V. If P exists, it has to be far from $\mathcal{V}$ because of the winning condition of MiM-game. Assuming that $\mathcal{V}$ and P see the same (N_P, N_V), we look each round i for the case where r_i arrived on time. If r_i arrived on time, thanks to Lemma 3.3, the response sent by P is independent from r_i or the challenge that P received is independent from c_i sent by $\mathcal{V}$. In any case, the adversary's probability to pass each round is $\frac{1}{2}$ because the response r_i has to be correct and on time: the adversary guesses either r_i or c_i (post-ask or pre-ask attack as shown in Section 3.2). There may also be one round where the pre-ask strategy is done for a constant number of rounds until it makes P abort. After abort, there is an additional opportunity (in the last of these rounds) for the adversary to pass the round by guessing the response. Therefore,

$$p = \frac{3}{2^{n+1}} + \frac{q^2}{2^{\ell+1}} + \frac{q'^2}{2^{\ell+1}} + \frac{1}{2^\ell} + \mathsf{Adv}(PRF).$$

□

Theorem 5.9 (OT-MiM security of MiM-OTDB)**.** *MiM-OTDB is OT-MiM-secure (one time MiM-secure).*

Proof. Using the last game in the proof of Theorem 5.8, we can show that MiM-OTDB is OT-MiM-secure. □

Assuming that $\frac{q^2}{2^{\ell+1}} + \frac{q'^2}{2^{\ell+1}} + \frac{1}{2^\ell} + \mathsf{Adv}(PRF)$ is negligible, the success probability of a MiM-adversary is $\frac{3}{2^{n+1}}$ very close to the optimal security $\frac{1}{2^n}$.

MiM-symDB is more efficient than the existing optimally MiM-secure protocols DB1, DB2, DB3 [BV14]. The provers in DB1, DB2, DB3 compute a PRF function two times and compute some other mappings too. So, with parameter $n_c = n_r = 2$ in common structure, for a given target security, we construct a nearly optimal protocol, both in terms of number of round complexity and computation complexity.

Adaptation of MiM-symDB to SHM (Full-symDB^H): We define Full-symDB^H with the tuple $(\mathcal{K}, V, P_{dum}, B, H)$ where B, V and $\mathcal{K}$ are as in MiM-symDB, H is the same with P in MiM-symDB.

Theorem 5.10 (Security of Full-symDB^H)**.** *If f is a secure PRF, Full-symDB^H is secure in SHM.*

Proof. The conversion of Full-symDB^H in PM is $(\mathcal{K}, V, P^H_{dum}, B)$ which is equal to MiM-symDB. We know that MiM-symDB is MiM-secure as f is a secure PRF. Hence, Full-symDB^H with $(\mathcal{K}, V, P_{dum}, B.H)$ is secure thanks to Theorem 5.6. The security bound of Full-symDB^H is the same with the MiM-security bound of MiM-symDB.

□

Full-symDB^H is the first protocol that reaches the optimal secure bounds for MiM, DH, DF and TF secure.

The following counterexample shows why we need more than hiding the key from the prover to achieve TF-security and why Theorem 5.6 holds with P_{dum}.

Why is not Theorem 5.6 correct without P_{dum}?: Let us define Full-symDB$^{H'}$ with the tuple $(\mathcal{K}, V, P', B, H')$ where B, V and $\mathcal{K}$ are the same with those defined in MiM-symDB. P' and H' are as follows:

$\underline{P'}$	$\underline{H'(s)}$
`pick` $N_P \in \{0,1\}^s$	`receive` N_P, N_V
`send` N_P to V	`compute` $C\|\|R = f_s(N_V, N_P)$
`receive` N_V from V	`run` MiM-OTDB($C\|\|R$)
`send` N_V, N_P to H'	
`relay` between V and H'	

Letting P' pick N_P is the only difference between the prover of Full-symDB^H and the prover of Full-symDB$^{H'}$. However, Full-symDB$^{H'}$ is not DF-secure because of the following attack: Malicious and far away P' sends N_P, N_V to H'. After, in MiM-OTDB execution, P' sends a guessed challenge c' to H' as if V sends. If H' accepts c', then P' learns the corresponding response r'. If c' is not the valid challenge, then P' restarts with inputs N_P, N_V to H' again. This time, P' sends $1-c'$ as a challenge and learns the corresponding response. P' follows the same strategy until it learns all the challenges and the responses. For sure, after $2n$ trials, it determines $C||R$. After learning $C||R$, P' just sends $C||R$ to a close adversary. Finally, the adversary runs MiM-OTDB($C||R$) with V picked N_V and passes the protocol.

Clearly, $(\mathcal{K}, V, P'^{H'}, B)$ is MiM-symDB which is a conversion of Full-symDB$^{H'}$ into PM. So, the conversion is MiM-secure. However, Full-symDB$^{H'}$ is not secure. **This shows that Theorem 5.6 may not hold without P_{dum}.**

Actually, this insecurity result is not surprising because picking N_P has an influence on the security of the protocol as it can be seen in the proof of Theorem 5.8. We repeat the result of Theorem 5.6: the prover has to be as passive as P_{dum} on a security of the DB protocol in SHM to have the full security.

Remark that in Full-symDB$^{H'}$, even though P' does not learn any information about s, it is able to break the security. Therefore, we can see that the intuitive idea [SP05, BR04] of sealing secret keys in a secure hardware is not enough, in general, to always protect against TF. This shows the necessity of a formal model.

5.4 Optimal Public-key DB Protocols in SHM

In this section, we give two public key DB protocols in SHM: Simp-pkDB^H and Eff-pkDB^H which is correct, private and secure. The first one is derived from Simp-pkDB^p in PM (Section 4.4.4). The second one is derived from the version Eff-pkDB^p with OT-MiM security (Section 4.4.3) in PM. We use these protocols because of their efficiency in PM.

Simp-pkDB^H: This protocol is derived from Simp-pkDB^p in Section 4.4.4. It is the same as Simp-pkDB^p except that P and H in Simp-pkDB^H shared the computation done by P in Simp-pkDB^p. In Simp-pkDB^H, P encrypts the nonce N picked by H along with pk_P. H decrypts the encryption sent by V. The algorithm V is unchanged. As a symmetric DB, V and H runs MiM-OTDB. The protocol is depicted in Figure 5.5.

$$
\begin{array}{lcccr}
\underline{V(\mathsf{sk}_V,\mathsf{pk}_V)} & & \underline{P(\mathsf{pk}_P,\mathsf{pk}_V)} & & \underline{H(\mathsf{sk}_P,\mathsf{pk}_P)} \\
\mathsf{pk}_P, N = \mathsf{Dec}'_{\mathsf{sk}_V}(e_P) & \xleftarrow{\quad e_P \quad} & e_P = \mathsf{Enc}'_{\mathsf{pk}_V}(\mathsf{pk}_P, N) & \xLeftarrow{\mathsf{pk}_P, N} & N \leftarrow \{0,1\}^s \\
\text{pick } C||R \in \{0,1\}^{2n} & & & & \\
e_V = \mathsf{Enc}_{\mathsf{pk}_P}(C||R||N) & \xrightarrow{\quad e_V \quad} & & \xRightarrow{e_V} & C||R||N = \mathsf{Dec}_{\mathsf{sk}_P}(e_V) \\
 & & & & \text{Verify } N \\
 & & \xleftrightarrow{\qquad\qquad \mathsf{MiM\text{-}OTDB(C||R)} \qquad\qquad} & & \\
\mathsf{POut}_V = \mathsf{pk}_P & & & &
\end{array}
$$

Figure 5.5 – Simp-pkDB^H. The double arrow shows the communication between P and H

Theorem 5.11 (Security of Simp-pkDB^H)**.** *If the encryption scheme (*$(\mathsf{Enc},\mathsf{Dec})$*) is IND-CCA secure, Simp-pkDB^H is secure in SHM.*

Proof. Consider $DB = (\mathcal{K}_V, \mathcal{K}_P, V, P^H_{dum}, B)$ with V and H from Simp-pkDB^H. Actually, $DB =$ Simp-pkDB^p. Using Theorem 5.6, Simp-pkDB^H is secure because $DB =$ Simp-pkDB^p is MiM-secure (Theorem 4.17) assuming that $(\mathsf{Enc},\mathsf{Dec})$ is IND-CCA secure and MiM-OTDB is OT-MiM-secure.

□

Simp-pkDB^H achieves almost optimal security bounds because MiM-security of Simp-pkDB is reduced to MiM-security of MiM-OTDB as shown in the proof of Theorem 4.17.

We see that Simp-pkDB^H is still secure without Enc' (only Enc needs security). Actually, this encryption is only used for achieving privacy. So, if privacy is not a concern, we can use Simp-pkDB^H without the encryption and decryption. In this case, the verifier has no secret/public key pair. This can be useful in practical applications.

Now, we prove that Simp-pkDB^H is semi-strong private in SHM.

Theorem 5.12 (Semi-strong privacy of Simp-pkDB^H)**.** *Assuming that the encryption scheme with* $(\mathsf{Enc}',\mathsf{Dec}')$ *is IND-CCA secure and the encryption scheme with* $(\mathsf{Enc},\mathsf{Dec})$ *is IND-CCA and IK-CCA [BBDP01] secure, then Simp-pkDB^H is semi-strong private in SHM.*

Proof. The proof works like in Theorem 4.18. We only let non-anonymous hardware decrypt e_V from the adversary with the right key through a CCA query in the IK-CCA game.

□

Eff-pkDB^H: One of the assumptions in MiM-security of Eff-pkDB^p is that the symmetric DB is "one-time multi-verifier MiM-secure" defined in Definition 4.7. It is not possible to use MiM-OTDB on Eff-pkDB^p as a symmetric DB because MiM-OTDB does not fulfill the assumption. Hence, we use Eff-pkDB+1 (Section 4.4.3) when contruction Eff-pkDB^H. In this way, we are able to use MiM-OTDB as a symmetric DB which does not require any computation.

Eff-pkDB^H is the same as Eff-pkDB+1 (Figure 4.5) except that P and H share some computations done by P in Eff-pkDB^p. P computes the encryption and H selects the nonce and runs B. As a symmetric DB, V and H runs MiM-OTDB. The protocol is depicted in Figure 5.6.

$\underline{V(\mathsf{sk}_V,\mathsf{pk}_V)}$		$\underline{P(\mathsf{pk}_P,\mathsf{pk}_V)}$		$\underline{H(\mathsf{sk}_P,\mathsf{pk}_P)}$
$N_V \leftarrow \{0,1\}^s$	$\xrightarrow{N_V}$		$\xRightarrow{N_V,\mathsf{pk}_V}$	$N_P \leftarrow D(1^s)$,
$\mathsf{pk}_P, N_P = \mathsf{Dec}_{\mathsf{sk}_{V_1}}(e)$,	$\xleftarrow{e}$	$e = \mathsf{Enc}_{\mathsf{pk}_{V_1}}(\mathsf{pk}_P, N_P)$	$\xLeftarrow{N_P,\mathsf{pk}_P}$	
$C\|\|R =$ $A_{N_V}(\mathsf{sk}_{V_2},\mathsf{pk}_{V_2},\mathsf{pk}_P,N_P)$				$C\|\|R =$ $B_{N_V}(\mathsf{sk}_P,\mathsf{pk}_P,\mathsf{pk}_{V_2},N_P)$
		$\xleftrightarrow{\text{MiM-OTDB(C\|\|R)}}$		
$\mathsf{POut}_V = \mathsf{pk}_P$				

Figure 5.6 – Eff-pkDB^H. Double arrow shows the communication with H.

Theorem 5.13 (Security of Eff-pkDB^H)**.** *If the key agreement protocol* $(\mathsf{Gen}_V,\mathsf{Gen}_P,A_{N_V},B_{N_V},D)$ *is D-AKA secure (Definition 4.2) for all fixed* $N_V \in \{0,1\}^s$*, then Eff-pkDB^H is secure in SHM.*

Proof. Consider that $DB = (\mathcal{K}_V,\mathcal{K}_P,V,P^H_{dum},B)$ with V and H from Eff-pkDB^H is MiM-secure in PM. Actually, DB is Eff-pkDB+1. Using Theorem 5.6, Eff-pkDB^H is secure because Eff-pkDB+1 is MiM-secure (Theorem 4.15) assuming that the key agreement protocol $(\mathsf{Gen}_V,\mathsf{Gen}_P,A_{N_V},B_{N_V},D)$ is D-AKA secure for all fixed $N_V \in \{0,1\}^s$ and MiM-OTDB is **one time MiM-secure**.

□

We see that Eff-pkDB^H is secure without encryption. Actually, the encryption is used for achieving privacy. So, if privacy is not a concern, we can use Eff-pkDB^H without the encryption and decryption.

Theorem 5.14 (Strong privacy of Eff-pkDB^H)**.** *Assuming that the key-agreement protocol* $(\mathsf{Gen}_V,\mathsf{Gen}_P,A_{N_V},B_{N_V},D)$ *is D-AKA^p secure (Definition 4.2) for all fixed* $N_V \in \{0,1\}^n$ *and the encryption scheme* $(\mathsf{Enc},\mathsf{Dec})$ *is IND-CCA secure, Eff-pkDB^H is strong private in SHM.*

Proof. We first show that the version of Eff-pkDB^p with OT-MiM is strong private in PM. Actually, the strong privacy proof of our variant of Eff-pkDB^p is the same with the proof of Eff-pkDB^p (Theorem 4.14) where first it reduces the privacy game to the game where all the encryptions are random (the reduction showed by using IND-CCA security) and then reduces to the game where the provers use a random secret and public key pair with B_{N_V} (the reduction showed by using D-AKA^p). Because of the equivalence of strong privacy of a DB in SHM and its conversion in PM, we can conclude that Eff-pkDB^H is strong private.

□

The prover algorithms of Simp-pkDB^H and Eff-pkDB^H are not P_{dum}, but it can be easily seen from the proofs of Theorem 5.11 and Theorem 5.13 that the computations in these algorithms do not have any effect on the security (i.e., the security of Simp-pkDB^H and Simp-pkDB^H do not need any security assumptions on the encryption scheme with $(\mathsf{Enc}', \mathsf{Dec}')$ which is used by P.)

5.5 Conclusion

In this chapter, we defined a new DB with three algorithms and designed its adversarial and communication model of SHM. According to our new model, we propose a new security definition. We showed that the trivial attack of TF is preventable in our definition. By showing implications between different threat models, we deduced that if a DB protocol achieves TF-security in SHM, then it is secure against all other security notions. This result cannot be applied in PM because TF-security is not possible. We also gave some security relations between PM and SHM. One of the relations shows that we can construct a DB protocol that is secure against all the threat models including TF in SHM if its conversion into PM is MiM-secure. This result is significant because it shows that many MiM-secure DB protocols in the literature [BV14, Vau13, BMV13a, BC93, HPO13, KV16, Vau15c] can be used to achieve higher security level in our model.

We also gave two constructions which are converted from Eff-pkDB^p and Simp-pkDB^p.

Compared to the previous models [ABK$^+$09, ABK$^+$11, DFKO11, BV14] which do not have any practical and secure solution against all the threats, SHM lets us construct more efficient protocols while achieving the **highest security**.

Part II

Integrated Distance Bounding

Chapter 6

Contactless Access Control

Contactless access control (AC) systems are critical for security but often vulnerable to relay attacks. In this chapter, we define an integrated security and privacy model for access control using distance bounding (DB) which is the most robust solution to prevent relay attacks. We show how a secure DB protocol can be converted to a secure contactless access control protocol. Regarding privacy (i.e., keeping anonymity in strong sense to an active adversary), we show that the conversion does not always preserve privacy but it is possible to study it on a case by case basis. Finally, we provide two example protocols and prove their security and privacy according to our new models.

The content of this chapter is published in ISC17 [KV17].

Related Works

A report from Smart Card Alliance [All03] lists the main components of an access control system (tags, readers, controllers, database) and their security requirements which are however informal. Wongsen et al. [WKC$^+$12] proposed an access control protocol between doors and mobile units (e.g. smartphone), but the protocol lacks any security proof. Some access control systems such as OPACITY [All13] and PLAIN [gDoHSD10] mutually authenticate and establish a shared key between the terminal and card. The security analysis of PLAIN in [gDoHSD10] is far from being formal. OPACITY [All13] was partly analyzed by Dagdelen et al. [DFG$^+$13] where their security model is based on the key agreement security model of Bellare and Rogaway [BR93]. Hence, most of the previous works do not have a comprehensive security analysis. Moreover, **none of them consider relay attacks in their security analysis**. Unfortunately, these types of attacks are easily implementable [Han05, Han06, FDČ11, RLS13, FHMM10, MFHM12], so they violate access control.

The other problem in contactless AC is to address privacy. Informally, if an AC protocol is private then it is hard for an outside observer to identify or recognize a party who wants to access a system. Some previous works [gDoHSD10, DFF$^+$14, DFG$^+$13] touched on privacy. PLAID [gDoHSD10] claims to be private (with an informal definition) but

Degabriele et al. [DFF$^+$14] show that it is weaker than what it claims. Dagdelen et al. [DFG$^+$13] give two privacy related definitions: identity hiding and untraceability. The problem in their privacy model is that it only considers the interaction between the card and the reader. In reality, this may not be enough because the other interactions or outputs of the other components (i.e., controller, database) of an AC system can violate the privacy.

As a result, a formal security model which covers relay attacks has not been designed for AC. In addition to this, a formal privacy model which considers whole AC system is missing.

6.1 Our Contribution

By considering these critical issues, we design the first security and privacy model of an access control system which encompasses the propagation time of communication. Intuitively, in our definitions, we mix DB and access control based on a database of privileges. However, mixing both is not so straightforward when it comes to prove the security in a generic composition. Current AC protocols [All13, gDoHSD10] do not consider malicious users in their security models while DB considers malicious users. Therefore, the natural composition of them does not necessarily achieve the security level we need for AC protocols[1]. In addition, we can show that an AC protocol which is constructed based on a private DB protocol does not achieve privacy in AC. All these reasons obviously show the need for complete security and privacy models in AC. Our contributions in this chapter are as follows:

- We first define **an integrated security model for AC** including identification, access control, and distance bounding by using the same components as defined in Smart Card Alliance [All03].
- We define **a new privacy model for AC** which includes the time of the communication. To the best of our knowledge, the time of the communication has not been considered for defining a privacy model before. Our new model covers all the previously defined privacy related definitions for access control such as identity hiding and untraceability.
- We give **a framework that clarifies how to use a secure DB** to construct a secure AC in our new security model. Basically, we show how to transform a man-in-the-middle (MiM), distance-fraud (DF) and distance-hijacking (DH) secure DB protocol into a secure AC scheme with proximity check. We also formally prove the security of this transformation.
- We show that the same framework can be used to achieve privacy in AC with restrictions on the database of the AC system: The framework achieves privacy if

[1] A malicious user can behave maliciously in an AC protocol and retrieve some information which may help him to attack the DB protocol which is composed with this AC protocol.

the database is trivial meaning that it is empty, or it includes all possible relations. We give a counterexample protocol that clearly shows why the framework does not work for non-trivial databases. This shows that **privacy in distance bounding is not always preserved when transformed into an access control system** which unfolds the need for a new model.

- We construct a specific AC scheme by using Eff-pkDB^p [KV16] (Section 4.4) and prove its security and privacy with any type of database.

Structure of the Chapter: In Section 6.2, we introduce our new security and privacy model for AC. Then, in Section 6.3, we show how we can achieve secure and private AC protocols by using secure and private DB protocols. We conclude this chapter with Section 6.4.

6.2 Security and Privacy Model of AC

We first introduce the components of an access control system. In our definitions, for simplicity, we do not consider the user who may give a PIN code or a biometric data to authenticate himself (this would be a parallel protocol). The components of an access control system are tag, reader, database and controller. Controller and database are in the secure area of AC system. where it is not possible to tamper or access.

Tags (Access Cards): They hold personalized data which is used for identification and authentication. In an AC system, each tag T generates a secret/public key pair $(\mathsf{sk}_T, \mathsf{pk}_T)$. They also store the public key of the controllers that are responsible for the doors[2] that T can access.

Reader: A reader is an interface between a tag and a door. We can consider them as transmitters. They communicate with the tags. Each reader R has a location loc_R which is important as the tag can be granted if the tag proves that it is close enough to the reader.

Database: It contains information about tags and their rights. It stores a list of $(\mathsf{pk}_T, loc_R, req)$ triplets meaning that the tag with pk_T is allowed to make the service request req on a reader which is at location loc_R. For instance, a service request can be "opening of a door". The database is in the secure area.

The database is not necessarily a list of triplets. It can also be a predicate deciding if a triplet belongs to it or not. A database is ***trivial*** if it is empty or if it contains all possible triplets.

For simplicity, we consider that the content of the database is static in what follows.

[2] Door is a representation of the system or service that a user desires to access.

Controller: It controls access authentication. All controllers can be connected with multiple readers. Depending on the data which is received from one of the connected readers and the database, they give the final decision for the authorization.

More generally, the access control is relative to a service (such as opening a door) in a given location. The tag T of public key pk_T requests a service req to a reader at location loc_R and its corresponding controller checks if the privilege $(\mathsf{pk}_T, loc_R, req)$ exists in the database. T stores req and it can change req later on. All controllers stay in the secure area.

Definition 6.1 (Access Control (AC)). *AC consists of a distance bound B, a database $DataB$, a controller C, a reader R, and a tag T, the key generation algorithms: Gen_C generating $(\mathsf{sk}_C, \mathsf{pk}_C)$ for a controller C and Gen_T generating $(\mathsf{sk}_T, \mathsf{pk}_T)$ for a tag T. C, R, and T run the algorithms $\mathcal{C}(\mathsf{sk}_C, \mathsf{pk}_C, DataB, B), \mathcal{R}(loc_R)$ and $\mathcal{T}(\mathsf{sk}_T, \mathsf{pk}_T, \mathsf{pk}_C, req)$, respectively. In the end of the protocol, C outputs either $\mathsf{Out}_C = 1$ and private output $\mathsf{POut}_C = (\mathsf{pk}_T, loc_R, req)$ if the authentication succeeds or $\mathsf{Out}_C = 0$ if it fails. R also publicly outputs $\mathsf{Out}_R = \mathsf{Out}_C$.*

Definition 6.2 (Correctness of AC). *We say that an AC is correct, when for all loc_R, req and for all sets of keys generated by Gen_C and Gen_T, if*

- *T requests service req to R at location loc_R,*
- *T is within a distance at most B from loc_R and*
- *$(\mathsf{pk}_T, loc_R, req)$ is in $DataB$,*

then

$$\mathsf{Pr}[\mathsf{Out}_C = 1 \wedge \mathsf{POut}_C = (\mathsf{pk}_T, loc_R, req)] = 1$$

The probability is over the randomness in algorithms $\mathcal{C}$, $\mathcal{R}$ and $\mathcal{T}$.

6.2.1 Security of AC

In this section, we give the formal security model for an access control system.

Adversarial and Communication Model: Each party (readers, controllers, tags, adversaries) has polynomially many instances. An instance of a party corresponds to a protocol execution with this party at a given location and time. Each instance of our model is as follows:

- We have the communication model of distance bounding as described in Section 2.2.1.
- Readers are all honest. They are connected to their corresponding controllers with a secure and an authenticated channel.
- Controllers are all honest. They are the only components of the AC which can access the database.

- **Tags are all honest**. However, they can receive special signals as defined in Section 2.2.1 by replacing P with tags. We also change the input of $\mathsf{Activate}$ as follows: the special signal $\mathsf{Activate}(T, req)$ activates the only activatable instance of T with a specified input req[3].
- **Adversaries create the database**. So, they can generate fake relations $(\tilde{\mathsf{pk}}_T, ., .)$ where $\tilde{\mathsf{pk}}_T$ and its corresponding secret key $\tilde{\mathsf{sk}}_T$ are generated by an adversary. Instances which could hold some $\tilde{\mathsf{sk}}_T$ are called **fake tags**.

  Except for the communication between readers and controllers, the adversary instances see all communication.

Definition 6.3 (AC-Security)**.** *The game begins by setting up the components of the AC system. The security game is as follows given the security parameter ℓ:*

- *Run $\mathsf{Gen}_C(1^\ell) \to (\mathsf{sk}_C, \mathsf{pk}_C)$ for the controller and run $\mathsf{Gen}_T(1^\ell) \to (\mathsf{sk}_{T_i}, \mathsf{pk}_{T_i})$ for each tag T_i and give the public key pk_C and pk_{T_i}'s to the adversary.*
- *The adversary creates instances of T_i at chosen locations. Each instance can start after activation and run $\mathcal{T}(\mathsf{sk}_{T_i}, \mathsf{pk}_{T_i}, \mathsf{pk}_C, req)$ only once.*
- *The adversary creates instances of readers at chosen locations loc_{R_k}. They run $\mathcal{R}(loc_{R_k})$ once activated by an incoming message. They communicate with an instance of C over a secure channel [4]. There is a distinguished instance of a reader R. We denote by loc_R its location.*
- *The adversary sets $DataB$.*
- *The adversary creates instances of himself (fake tags). These instances run independently and communicate.*

All messages follow our communication model. The game ends when the distinguished instance R (and its corresponding instance C) outputs some value Out_R. An AC protocol is secure, if for any such game, the adversary wins with a negligible probability. $\mathcal{A}$ wins the game if $\mathsf{Out}_R = 1$ and $\mathsf{POut}_C = (\mathsf{pk}_T, loc_R, req)$ for some pk_T and req satisfying at least one of the following conditions:

1. $(\mathsf{pk}_T, loc_R, req) \notin DataB$,
2. $\mathsf{pk}_T \in \{\mathsf{pk}_{T_i}\}_{i=1}^t$ *and no active instance of the honest tag holding pk_T is close to loc_R during the execution of the AC protocol with C and R,*
3. $\mathsf{pk}_T \notin \{\mathsf{pk}_{T_i}\}_{i=1}^t$ *and no fake tag is close to loc_R during the execution with C and R.*

where t is the number of public keys generated by Gen_T in setup.

[3] This can also correspond to a user who is the owner of T to input whatever requests he wants into his tag.

[4] For simplicity, we assume that the instance C of the controller is at the same location as R_k but the time of communication between R_k and C should have no influence on the result. The difference between C and R_k only makes sense for practical reasons.

Remarks:

- In the third condition, we need that no fake tag is close to loc_R to prevent the trivial attacks where a far away fake tag can give its secret key to a close by fake tag. Without this condition, the adversary would always win. This would however exclude all TF-attacks as well.
- The third condition is to prevent DF or DH attacks.
- The second condition is to have security against MiM attacks including impersonation attacks and relay attacks.

In practice, the controllers are connected to multiple readers. So, it is not practical for them to check if a tag is close. Therefore, readers are the components that can give this decision.

Before proceeding the next part, we show that the natural composition of access control and distance bounding does not always achieve the security in Definition 6.3. Assume that we have a MiM, DF and DH secure symmetric DB protocol $DB = (\mathcal{K}, P, V, B)$. As an AC protocol, we have an AC protocol OPACITY [All13] [5]. In the natural composition, first, the parties run OPACITY with a minor change and then DB (the reader runs V, the tag runs P with the secret key K). The change in OPACITY is as follows: the reader sends K at the end of the OPACITY protocol. Clearly, the modified version of OPACITY is still secure AC in the security model of Dagdelen et al. [DFG$^+$13] as K is completely independent parameter. Unfortunately, this composition is not secure in Definition 6.3 as an adversary can win AC-game with satisfying the second condition. However, when we look at the modified OPACITY and DB separately in their own security models, they are secure. Therefore, the generic composition of AC and DB is not straightforward.

6.2.2 Privacy of AC

Privacy is also an important property to be achieved in access control protocols. The definition of privacy we provide uses the same adversarial and communication model that we use for security. It also covers the identity hiding and untraceability with the corruption of tags. Informally, identity hiding means given an execution of a protocol the adversary should not output the public key of the tag and untraceability means the adversary should not decide if two executions belong to the same tag or not.

We adapt Definition 2.14 in a setting where the transmission time is important.

Definition 6.4 (AC-Privacy)**.** *The privacy game has the same setting as the game in Definition 6.3. We first decide to play the right r or the left l game. Differently than the security model, each active tag instance can be paired with an another tag instance by an adversary. The pairing happens with the signal* $\mathsf{Draw}(T_i, T_j, k)$ *which pairs T_i and T_j by giving an index k, if the conditions below are satisfied:*

[5] OPACITY is basically a key agreement protocol where the authentication of a tag is done with this key.

- *T_i and T_j are at the same location,*
- *T_i and T_j have the same access privileges,*
- *neither T_i nor T_j is already paired and*
- *k is greater than the index of previous* Draw *signal to both T_i and T_j.*

A tag instance can be paired to itself as well. The adversary lets $\mathsf{vtag} = (T_i, T_j, k)$ be a virtual tag. All messages (and special signals) can only have a virtual tag as a destinator. If we are in game l, then vtag simulates T_i and if we are in game r, vtag simulates T_j. The signal $\mathsf{Free}(T_i, T_j, k)$ *breaks the pair if it exists. The adversary can corrupt a tag T_i (and actually all tags) by receiving* sk_{T_i} *during the setup.*

In the end, the adversary decides if vtag simulates game r or game l. If the decision of the adversary is correct, then the adversary wins.

If an AC protocol is private, the advantage of a polynomial time adversary in this game is bounded by a negligible probability.

The most important distinction of our definition is that we consider "communication time which leaks the proximity of a party" contrarily of previous work related to privacy [Vau07, HPVP11]. To the best of our knowledge, it has not been taken into account before for a privacy model. It is reasonable to consider the location of a user as a privacy leakage for the protocols where the communication time influences the output such as DB.

As Mitrokotsa et al. [MOV14] showed that location privacy is nearly impossible to achieve, we cannot prevent this leakage. So, our privacy game has the condition of being at the same location which is necessary to avoid the adversary to trivially distinguish the left or right game by checking the communication time.

Besides, the condition of having the same access privileges is necessary to prevent the adversary to determine the left or right game by seeing the accepting or the rejecting message by a controller.

6.3 Distance Bounding in Access Control

In this section, instead of designing a new AC protocol, we give a conceivable framework that converts a DB protocol into an AC protocol. We prove in Theorem 6.5 that, after conversion, the AC protocol achieves AC-security (in Definition 6.3) assuming that the DB protocol is MiM and DH secure. However, we show that we cannot always achieve AC-Privacy with this framework, even though the DB protocol is (strong) private according to Definition 2.14. Therefore, we prove in Theorem 6.6 that the AC protocol which is converted from a private DB achieves privacy, if ***DataB*** is trivial. The details are in the following subsections.

$\mathcal{C}(\mathsf{sk}_V, \mathsf{pk}_V, \mathit{DataB}, B)$		$\mathcal{R}(loc_R)$		$\mathcal{T}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, req)$
	$\xleftarrow{req, loc_R}$		$\xleftarrow{req}$	
run $V(\mathsf{sk}_V, \mathsf{pk}_V)$		$\xleftrightarrow{\text{run } DB=(\mathcal{K}_P, \mathcal{K}_V, P, V, B)}$		run $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$
output Out and pk				
if $(\mathsf{pk}, loc_R, req) \in \mathit{DataB}$				
$\quad \mathsf{Out}_C = \mathsf{Out}$				
$\quad$ **if** $\mathsf{Out} = 1$				
$\quad\quad \mathsf{POut} = (\mathsf{pk}, loc_R, req)$				
else: $\mathsf{Out}_C = 0$				
	$\xrightarrow{\mathsf{Out}_C}$		$\xrightarrow{\mathsf{Out}_C}$	

Figure 6.1 – The framework to convert a DB protocol to an AC protocol

6.3.1 Secure AC with a Secure DB

If we have a public-key DB protocol $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$, we can construct an AC protocol with $(\mathsf{Gen}_C, \mathsf{Gen}_T, \mathcal{C}, \mathcal{T}, \mathit{DataB}, B)$ with the framework below:

- We match the key generation algorithms: $\mathsf{Gen}_C = \mathcal{K}_V$, $\mathsf{Gen}_T = \mathcal{K}_P$. So, $(\mathsf{sk}_C, \mathsf{pk}_C) = (\mathsf{sk}_V, \mathsf{pk}_V)$ and $(\mathsf{sk}_T, \mathsf{pk}_T) = (\mathsf{sk}_P, \mathsf{pk}_P)$.

- We create *DataB* according to the access privileges of tags using the keys.

- $\mathcal{T}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, req)$ uses $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ as a subroutine. $\mathcal{T}$ outputs req and then run $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$.

- Whenever $\mathcal{R}(loc_R)$ is activated with req, it sends req and loc_R to $\mathcal{C}$.

- $\mathcal{C}(\mathsf{sk}_V, \mathsf{pk}_V, \mathit{DataB}, B)$ runs $V(\mathsf{sk}_V, \mathsf{pk}_V)$ as a subroutine jointly with $\mathcal{R}(loc_R)$. When V reaches the part where challenge/response is necessary to determine the distance to loc_R, $\mathcal{R}$ steps in to check if the responses arrive on time and are correct.

  Here, $\mathcal{C}$ may give all necessary input(s) to $\mathcal{R}$ so that $\mathcal{R}$ can check the responses. Alternatively, $\mathcal{C}$ may only give the challenges, and $\mathcal{R}$ only determines if the responses arrive on time. Then, if they arrive on time, $\mathcal{R}$ can send the responses to $\mathcal{C}$ so that $\mathcal{C}$ can check if the responses are correct. The only restriction is that $\mathcal{R}$ **has to decide if the responses arrive on time.**

- When $V(\mathsf{sk}_V, \mathsf{pk}_V)$ outputs Out and the private output pk_P: If $(\mathsf{pk}_P, loc_R, req) \in \mathit{DataB}$ and $\mathsf{Out} = 1$, it publicly outputs $\mathsf{Out}_C = 1$ and privately outputs $\mathsf{POut}_C = (\mathsf{pk}_P, loc_R, req)$. Otherwise, it outputs $\mathsf{Out}_C = 0$. In both cases, $\mathcal{R}$ outputs $\mathsf{Out}_R = \mathsf{Out}_C$. The framework is in Figure 6.1.

An example protocol in Figure 6.2 is constructed using this framework. Before, we prove that the framework achieves AC security if DB is MiM and DH secure.

Theorem 6.5. *Assuming that a DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$ is MiM-secure (Definition 2.12) and DH-secure (Definition 2.10), then an AC protocol using this DB protocol with the framework as described in Figure 6.1 is secure according to Definition 6.3.*

Proof. Assume that there exists an adversary $\mathcal{A}$ which wins the game in Definition 6.3 where the output of the game is $\mathsf{Out}_R = 1$ and $\mathsf{POut}_C = (\mathsf{pk}_{T_i}, loc_R, req)$, then we can construct an adversary which wins MiM-game or DH-game.

Apparently, $\mathcal{A}$ can win the AC-game with either second or third condition because C outputs $\mathsf{Out}_C = 0$ if given $(\mathsf{pk}_{T_i}, loc_R, req) \notin DataB$ (the first winning condition) which makes impossible to win with the first condition.

Winning with the second condition: If $\mathsf{pk}_{T_i} \in \{\mathsf{pk}_{T_k}\}_{k=1}^{t}$ and no instance of the tag with pk_{T_i} is close to loc_R during the execution of the AC protocol with C and R, then we can construct an adversary $\mathcal{B}$ which wins MiM-game (Definition 2.12) of DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V)$.

$\mathcal{B}$ receives pk_V and pk_P from MiM-game. Then, it randomly picks $i \in \{1, ..., t\}$ where t is the number of (honest) tags needing to be simulated. The public key pk_{T_i} which will be used to simulate the i^{th} tag T_i is pk_P. Here, T_i will have a role as a prover on MiM-game. For the rest of the tags, $\mathcal{B}$ generates $t-1$ secret/public key pairs $(\mathsf{sk}_{T_j}, \mathsf{pk}_{T_j})$ with using $\mathsf{Gen}_T(1^n)$ which are the secret/public keys of T_j's. Then, it sends pk_V as the controller's public key and $\mathsf{pk}_{T_1}, ..., \mathsf{pk}_{T_{i-1}}, \mathsf{pk}_{\boldsymbol{P}}, \mathsf{pk}_{T_{i+1}}, ..., \mathsf{pk}_{T_t}$ as the tags' public-keys in AC-game to $\mathcal{A}$. Remark that pk_V and pk_P are indistinguishable since they are generated with the same key generation algorithms of controllers and tags, respectively.

At some moment, $\mathcal{B}$ receives $DataB$ from $\mathcal{A}$. If $(\mathsf{pk}_P, ., .) \notin DataB$, then $\mathcal{B}$ loses the MiM-game as in this case, there will be no chance that $\mathcal{A}$ wins the AC-game with this tag. Otherwise, it locates instances of T_i (which corresponds to P's instances in the MiM-game) on the locations that $\mathcal{A}$ decides. $\mathcal{B}$ simulates the instances of AC-game as follows:

- <u>Instances of T_j's where $T_j \neq T_i$</u>: For the signals $\mathsf{Move}(T_j, loc)$ and $\mathsf{Terminate}(T_j)$, $\mathcal{B}$ just simulates. When it receives the signal $\mathsf{Activate}(T_j, req)$, it simulates by running the algorithm $\mathcal{T}(\mathsf{sk}_{T_j}, \mathsf{pk}_{T_j}, \mathsf{pk}_V, req)$. Remark that as $\mathcal{B}$ knows each sk_{T_j}, it can run $\mathcal{T}$.

- <u>Instances of T_i</u>: For the signals $\mathsf{Move}(T_i, loc)$ and $\mathsf{Terminate}(T_i)$, $\mathcal{B}$ moves the corresponding instance of P in the MiM-game to loc and halts the corresponding instance of P in the MiM-game, respectively. Whenever it receives the signal $\mathsf{Activate}(T_i, req)$, it first outputs req and then runs (activates) the corresponding instance of P in the MiM-game. Whatever the instance of P in MiM-game outputs, $\mathcal{B}$ outputs the same.

- <u>Instances of controller and reader:</u> Whenever $\mathcal{A}$ activates $\mathcal{R}$ (via sending req) so that $\mathcal{C}$, $\mathcal{B}$ runs an instance of V.

In the end, if $\mathcal{A}$ picks a reader instance R which sees $\mathsf{pk}_{T_j} = \mathsf{pk}_P$ as a distinguished one, $\mathcal{B}$ wins with the success probability below. Otherwise, $\mathcal{B}$ loses MiM-game as V has to output $\mathsf{Out}_V = 1$ and pk_P in MiM-game.

$$\Pr[\mathcal{B} \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins} \wedge \text{Condition } 2] \times \frac{1}{t}$$

Winning with the third condition: If $\mathsf{pk}_T \notin \{\mathsf{pk}_{T_i}\}_{i=1}^t$ and no instance of the adversary is close to loc_R during the execution with R, then we can construct an adversary $\mathcal{B}'$ which wins DH-game. The reduction is very similar to the previous one except we replace P with an honest prover P'.

$$\Pr[\mathcal{B}' \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins} \wedge \text{Condition } 3] \times \frac{1}{t}$$

In the end, we have

$$\Pr[\mathcal{B} \text{ wins}] + \Pr[\mathcal{B}' \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins}] \times \frac{1}{t}.$$

As we know that the success probability of $\mathcal{B}$ in MiM and $\mathcal{B}'$ in DH game is negligible, then the success probability of $\mathcal{A}$ is negligible as well.

□

Now, we give an example of an AC protocol (Eff-AC) in our framework by converting the public-key DB protocol Eff-pkDB [KV16] (Section 4.4).

Eff-AC: We use Eff-pkDB with its variant. Its variant uses a key agreement protocol Nonce-DH [KV16] (Section 4.2) to agree on a secret S and a symmetric-key DB OTDB [Vau15c] to run with []S. We stress that this is only one example of the generic construction of Eff-pkDB. In particular, we could replace NonceDH by another key agreement protocol which is D-AKA secure [KV16] and possibly eliminate the random oracle assumption.

The public parameters for the key generation algorithms Gen_C ($\mathcal{K}_V$) and Gen_T ($\mathcal{K}_P$) are a group G of prime order q and its generator g. Gen_C and Gen_T pick sk_C and sk_T from $\mathbb{Z}_q$, and set $\mathsf{pk}_C = g^{sk_C}$ and $\mathsf{pk}_T = g^{\mathsf{sk}_T}$, respectively. Eff-AC works as follows:

The tag has the input $\mathsf{sk}_T, \mathsf{pk}_T, \mathsf{pk}_C, req$, the controller $\mathcal{C}$ has the input $\mathsf{sk}_C, \mathsf{pk}_C, B, DataB$ and the reader $\mathcal{R}$ has the input loc_R. $\mathcal{T}$ sends req to R and R sends it along with loc_R to C. Then, C, R and $\mathcal{T}$ run Eff-pkDB. Here, $\mathcal{T}$ runs the proving algorithm of Eff-pkDB, and $\mathcal{C}$ and $\mathcal{R}$ run the verifying algorithm of Eff-pkDB, jointly. The details of these algorithms are as follows: First, $\mathcal{T}$ picks a random value N from $\{0,1\}^n$ and sends N and pk_T. After $\mathcal{C}$ receives them, it computes $S = H(g, \mathsf{pk}_T, \mathsf{pk}_C, \mathsf{pk}_T^{\mathsf{sk}_C}, N)$. Meanwhile, $\mathcal{T}$ also computes $S = H(g, \mathsf{pk}_T, \mathsf{pk}_C, \mathsf{pk}_C^{\mathsf{sk}_T}, N)$. After, $\mathcal{C}$ gives S and B to R so that $\mathcal{R}$ runs the challenge phase. Until this part corresponds to the Nonce-DH protocol. Then, OTDB [Vau15c] is run by $\mathcal{R}$ and $\mathcal{T}$ as follows:

$\mathcal{R}$ picks a value $N_R \in \{0,1\}^{2n}$ and sends it to $\mathcal{T}$. Then, $\mathcal{R}$ and $\mathcal{T}$ compute $X = N_R \oplus S$ before the n-round challenge phase begins. In each round i, $\mathcal{R}$ picks a challenge Q_i and starts the timer. In response, $\mathcal{T}$ sends W_i which is the $2i + Q_i^{th}$ bit of X. When $\mathcal{R}$ receives it, it stops the timer. After the challenge phase, if all responses are correct and arrive on time (i.e. with in less than $2B$), then $\mathcal{R}$ sets $\mathsf{Out} = 1$. Then, $\mathcal{R}$ sends Out to $\mathcal{C}$. This is the end of Eff-pkDB.

$\mathcal{C}$ sets $\mathsf{Out}_C = \mathsf{Out}$. If $\mathsf{Out} = 1$, $\mathcal{C}$ checks if $\mathcal{C}$ has the access privilege by checking if $(\mathsf{pk}_T, loc_R, req) \in \mathit{DataB}$. If it is in DataB, it privately outputs $\mathsf{POut}_C = (\mathsf{pk}_T, loc_R, req)$. Otherwise, it sets $\mathsf{Out}_C = 0$. Finally, $\mathcal{C}$ sends Out_C to $\mathcal{R}$ and $\mathcal{R}$ outputs it as Out_R.

$\underline{\mathcal{C}_i}(\mathsf{sk}_C, \mathsf{pk}_C, B, \mathit{DataB})$		$\underline{\mathcal{R}}(loc_R)$		$\underline{\mathcal{T}}(\mathsf{sk}_T, \mathsf{pk}_T, \mathsf{pk}_C, req)$
	$\xLeftarrow{req, loc_R}$		$\xleftarrow{req}$	**pick** $N \in \{0,1\}^n$
$S = H(g, \mathsf{pk}_T, \mathsf{pk}_C, \mathsf{pk}_T^{\mathsf{sk}_C}, N)$	$\xLeftarrow{\mathsf{pk}_T, N}$		$\xleftarrow{\mathsf{pk}_T, N}$	$S = H(g, \mathsf{pk}_T, \mathsf{pk}_C, \mathsf{pk}_C^{\mathsf{sk}_T}, N)$
	$\xRightarrow{S, B}$	**pick** $N_R \in \{0,1\}^{2n}$	$\xrightarrow{N_R}$	$X = N_R \oplus S$
		$X = N_R \oplus S$		
			for $i = 1$ **to** n	
		start timer_i	$\xrightarrow{Q_i}$	$W_i = X_{2i+Q_i}$
		stop timer_i	$\xleftarrow{W_i}$	
		if $\forall i$ $\mathsf{timer}_i \leq 2B$ **and** $W_i = X_{2i+Q_i}$ $\mathsf{Out} = 1$		
if $\mathsf{Out} = 1$ **and** $(\mathsf{pk}, loc_R, req) \in \mathit{DataB}$ $\mathsf{POut} = (\mathsf{pk}, loc_R, req)$ $\mathsf{Out}_C = \mathsf{Out}$	$\xLeftarrow{\mathsf{Out}}$			
else: $\mathsf{Out}_C = 0$	$\xRightarrow{\mathsf{Out}_C}$		$\xrightarrow{\mathsf{Out}_C}$	

Figure 6.2 – Eff-AC. Double arrow shows that the communication is secure and authenticated while sending the message above it. The gray colored parts are Eff-pkDB.

As Eff-pkDB is MiM and DH-secure [KV16], **Eff-AC which uses Eff-pkDB with the framework in Figure 6.1 is AC-secure thanks to Theorem 6.5.**

Remark: The security proof of Eff-pkDB [KV16] is also valid for a variant where the verifier generates an ephemeral $(\mathsf{sk}_C, \mathsf{pk}_C)$ pair and sends pk_C to the prover. So, tags do not even need to store pk_C in this variant of Eff-pkDB. Therefore, a variant of Eff-AC with an ephemeral key is secure thanks to Theorem 6.5. This variant is very desirable for practical reasons because we can allow many controllers and the tag does not need to store all the corresponding keys.

6.3.2 Private AC with a Private DB

The difficulty in proving privacy in an AC protocol which uses a private DB protocol comes from the fact that ***DataB*** must discriminate tags. This fact may leak information about identities. In DB, the output of V does not depend on pk_P. Hence, the private output of the verifier (pk_P) plays no role in the DB privacy game of Definition 2.14. We show here a generic privacy preservation result with our framework, but only for a trivial ***DataB***. Trivial ***DataB*** makes POut_C play no role in AC. We cannot prove the same result for an arbitrary database. Remember, a database is *trivial* if it is empty or if it contains all possible triplets.

Theorem 6.6. *Assuming that the DB protocol with $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$ is private according to Definition 2.14, then an AC protocol with using this DB protocol with the framework as described in Figure 6.1 is private* ***when DataB is trivial*** *based on Definition 6.4.*

Proof. Assuming that there exists an adversary $\mathcal{A}$ breaking the privacy in AC with a trivial ***DataB***, then we can construct an adversary $\mathcal{B}$ that breaks the privacy of DB.

$\mathcal{B}$ simulates the communication model of AC for $\mathcal{A}$, except the subroutines P and V for honest participants. For each message and signal that $\mathcal{B}$ receives for tags, it works as follows:

- *Receiving a signal* $\mathsf{Draw}(T_i, T_j, k)$*:* It checks the necessary conditions to be paired. If they are satisfied, it calls the Draw oracle in the privacy game of DB with the inputs T_i, T_j. In respond, the Draw oracle sends $vtag$. $\mathcal{B}$ stores the information that $vtag$ corresponds to (T_i, T_j, k).

- *Receiving a signal* $\mathsf{Free}(T_i, T_j, k)$*:* It retrieves the corresponding $vtag$ to (T_i, T_j, k). If it exists, it calls the oracle Free with the input $vtag$ in the privacy game of DB.

- *Receiving a signal* $\mathsf{Activate}$ *or* Move*:* It simulates them.

- *Receiving a message m:* It retrieves $vtag$ and calls the oracle SendP in the privacy game of DB with the input $(vtag, m)$. Then, it receives a respond m' from the SendP oracle and sends m' to $\mathcal{A}$.

To simulate a reader receiving m, $\mathcal{B}$ behaves as follows:

- If it is the first time and $m = req$, $\mathcal{B}$ calls the Launch oracle to get a session identifier π. Then, it calls SendV with π and receives an empty message m'.

- Otherwise, it calls the oracle SendV with the input (π, m) and receives m'.

If m' is not the final message, it sends m' to $\mathcal{A}$. Otherwise, $m' = \mathsf{Out}_V$. In this case, $\mathcal{B}$ assigns $b = 0$ if ***DataB*** is empty and $b = 1$ if it is not empty (meaning that it has all possible relations). In the end, it sends $\mathsf{Out}_C = \mathsf{Out}_V \wedge b$ to $\mathcal{A}$. The simulation is perfect. So, $\mathcal{A}$ and $\mathcal{B}$ have the same advantage.

□

Why only for trivial *DataB*: We can show that Theorem 6.6 does not work for all *DataB* with the following counterexample.

Assume that we have a private DB $(\mathcal{K}_P, \mathcal{K}_V, P, V, B)$. From DB, we can construct another private protocol DB' $(\mathcal{K}_P, \mathcal{K}_V, P', V', B)$ where P' and V' work as defined below:

$\underline{P'(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V):}$
receive $flag$
if $flag = 1$ **and** pk_P is odd
$\quad \mathcal{K}_P \to (\mathsf{sk}'_P, \mathsf{pk}'_P)$
$\quad (\mathsf{sk}_P, \mathsf{pk}_P) \leftarrow (\mathsf{sk}'_P, \mathsf{pk}'_P)$
run $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_C)$

$\underline{V'(\mathsf{sk}_V, \mathsf{pk}_V)}$
send 0
run $V(\mathsf{sk}_V, \mathsf{pk}_V)$

Clearly, DB' is still private because the only change is to remove the identity of the prover by replacing the secret and public keys with some random keys. (We recall that pk_P as a private output of V plays no role in Definition 2.14.)

Now, let's consider the conversion of DB' to an AC protocol with the framework. The adversary can break the privacy of the AC protocol as follows: He first picks two tags T_1 and T_2 which have public keys with different parities and moves them at the same location. It also creates a $DataB = \{(\mathsf{pk}_{T_1}, loc_R, req), (\mathsf{pk}_{T_2}, loc_R, req)\}$. Then, it pairs (T_1, T_2) with the signal $\mathsf{Draw}(T_1, T_2, 0)$ and activates the pair. It sends a message $flag = 1$ to $vtag = (T_1, T_2, 0)$ (by replacing the message $flag = 0$ which comes from a reader R). Then, it lets C, R and $vtag$ execute the protocol. In the end, R outputs Out_R. Depending on the parity, the adversary can find out the left or right game with probability 1 (e.g., if pk_{T_1} is odd and $\mathsf{Out}_R = 1$, it means right game (T_2) is simulated).

In addition, even by weakening Definition 2.14 such that the adversary does not create a database and it is not allowed to pair tags (instead, the game does), we achieve **no** privacy. In this case, the advantage of the adversary with this attack would be $\frac{1}{2}$: If the public keys of paired parties have the same parity, then the attack does not give any more advantage than the privacy game of DB' gives. If they have different parity, the adversary wins with probability 1.

Even though we cannot use our framework to achieve privacy with **all** private DB protocols, we can still have private AC using our framework with **some** DB protocols where one of them is Eff-pkDB^p [KV16]. Now, we describe Eff-AC^p which is converted from Eff-pkDB^p.

Eff-AC^p (See Figure 6.3): It is very similar to Eff-AC. Differently here, the secret/public key pair of C consists of two parts: $(\mathsf{sk}_C, \mathsf{pk}_C) = ((\mathsf{sk}_{C_1}, \mathsf{sk}_{C_2}), (\mathsf{pk}_{C_1}, \mathsf{pk}_{C_2}))$ where $(sk_{C_1}, \mathsf{pk}_{C_1})$ is used for the encryption and $(\mathsf{sk}_{C_2}, \mathsf{pk}_{C_2})$ is used for Nonce-DH (key agreement protocol). The only change on $\mathcal{T}$ is that it sends the encryption of (pk_T, N) and on $\mathcal{C}$ is that it retrieves pk_T, N by decrypting the encryption with sk_{C_1}. The rest is the same with Eff-AC.

Theorem 6.7. *Eff-AC^p is a private access protocol in the random oracle model according to Definition 6.4, assuming that the cryptosystem is IND-CCA secure (Definition 2.18)*

$$\begin{array}{cccc}
\underline{C_i}(\mathsf{sk}_C,\mathsf{pk}_C,B,DataB) & & \underline{R}(loc_R) & & \underline{\mathcal{T}}(\mathsf{sk}_T,\mathsf{pk}_T,\mathsf{pk}_C,req) \\
 & \xLeftarrow{req,loc_R} & & \xleftarrow{req} & \textbf{pick } N\in\{0,1\}^n \\
N,\mathsf{pk}_T=\mathsf{Dec}_{\mathsf{sk}_{C_1}}(e) & \xLeftarrow{e} & & \xleftarrow{e} & e=\mathsf{Enc}_{\mathsf{pk}_{C_1}}(N,\mathsf{pk}_T) \\
 & \multicolumn{3}{c}{\xleftrightarrow{\text{The same as in Eff-AC}}} &
\end{array}$$

Figure 6.3 – Eff-AC^p

and GDH problem is hard (Definition 2.17).

Note that the same result applies to the generic construction of Eff-pkDB^p [KV16], i.e., not only the one based on GDH and the random oracle. We could indeed replace Nonce-DH by another key agreement protocol which is D-AKA^p secure [KV16].

Proof (sketch): We adapt the proof from the privacy proof of Eff-pkDB^p (Theorem 4.14).

We define games Γ_i^b below and the success probability of an adversary is p_i^b.

Γ_0^b : It is the same game that we defined in Definition 6.4 where $b=l$ meaning we are in the left-game or $b=r$ meaning we are in the right-game.

Γ_1^b : We reduce Γ_0^b to Γ_1^b where we simulate the controller instances without decrypting the ciphertext that is sent by a *vtag*. Because of the correctness of the cryptosystem, $p_1^b=p_0^b$.

Γ_2^b : We reduce Γ_1^b to Γ_2^b where *vtag* is simulated by encrypting a random value instead of (pk_T,N). We can easily show $p_2^b-p_1^b$ is negligible by using the IND-CCA security of the cryptosystem.

We reduce Γ_2^l to Γ_2^r where we replace all secret/public keys $(\mathsf{sk}_l,\mathsf{pk}_l)$ which are the keys of the tag in the left-side in *vtag* by replacing secret/public keys $(\mathsf{sk}_r,\mathsf{pk}_r)$ of its paired tag. Using D-AKA^p security of Nonce-DH (Theorem 7 in [KV16]), we can show that $p_2^l-p_2^r$ is negligible.

Remark that if pk_l and pk_r are kept in a plaintext and used by the controller, the replacing pk_l with pk_r make the same Out_C result due to our assumption which says the paired tags have the same access privileges.

So, $p_0^l-p_0^r$ is negligible. $\square$

6.4 Conclusion

In this chapter, we designed a security model for AC which considers the whole interaction between components. The security model integrates the model of DB since the distance of the tag is important to detect the relay attacks. In our model, we preserve the security against adversaries which can be a tag or not. We also let the adversaries construct the database. We constructed a privacy model for AC which includes time of communication as well.

We gave a simple framework which securely transforms a DB to an AC. We proved a similar result for privacy assuming that ***DataB*** is trivial. We showed why the theorem does not work for other types of database. Finally, we constructed two AC protocols Eff-AC and Eff-AC^p which are adapted from Eff-pkDB and Eff-pkDB^p [KV16], respectively. We proved their security and privacy in our security and privacy models.

Chapter 7

Secure Contactless Payment

A contactless payment (CP) lets a card holder execute payment without any interaction (e.g., entering a PIN code or signing) between the terminal and the card holder. Even though the security is the first priority in a payment system, the formal security model of contactless payment does not exist. Therefore, in this chapter, we design an adversarial model and define formally the contactless-payment security against malicious cards and malicious terminals which also deals with relay attacks. Accordingly, we design a contactless-payment protocol and show its security in our security model. At the end, we analyze EMV-contactless which is a commonly used specification by most of the mobile contactless-payment systems and credit cards in Europe.

The content of this chapter is published in ACISP18 [KV18b].

Related Works

Despite the big developments in CP, we realize that some important functionalities such as secure processing of payments has not been considered formally. No standard security model was provided for the contactless payment. Some pre-play attacks were detected for EMV because of poor random generation [BCM$^+$14, BCM$^+$15]. Roland and Langer [RL13] discovered a cloning attack for EMV contactless payment cards as the contactless payment process permits an attacker to learn the necessary credit card data for cloning. The cloned cards can then be used to perform EMV Mag-Stripe transactions at any EMV contactless payment terminal. Another type of pre-play attack [BCM$^+$14] was discovered which relies on the fact that EMV standards do not impose any encryption between a merchant and an acquirer, or between an acquirer and an issuer.

The most important attack specific for EMV-contactless (and also most of the contactless applications) is relay attack which has shown up for a while ago [MFHM12, Wei10, FHMM10, DM$^+$07, FDČ11]. Chothia et al. [CGDR$^+$15] remark that the first version of EMVco is vulnerable to relay attacks and provide a solution for this. The current EMVco [emvb], therefore, take precaution partly against relay attacks using the solution proposed by Chothia et al. [CGDR$^+$15]. It is "partly" because the solution they use is software based, where the terminal does not require a specific hardware. So, it protects against relatively trivial adversaries but does not protect against the adversaries using a sophisticated hardware [FDČ11, CHKM06]. To defend this level of security that they provide against relay attacks, Chothia et al. [CGDR+15] say that "Considering that contactless payments are limited to small amounts, the cost of the hardware would be a disincentive for criminals". However, limiting to small amounts does not necessarily mean that the relay attack outcome will be also a small amount. An attacker in a crowded area (e.g., metro, concert, museum) can execute many numbers of relay attacks and increase its outcome. In addition, some cards are limited to some small amounts in their issued country currency, but when they are abroad, this limit is removed because the conversion from the currency in the issued country to the currency in the current country cannot be computed. Besides this, the solution provided by Chothia et al. [CGDR+15] for EMV-contactless does not protect against DF and DH.

7.1 Our Contributions:

Considering all these attacks and the missing formalism, we design a new security model for contactless-payment protocols and design a secure contactless-payment protocol. In more detail, our contributions are as follows:

- We formally define contactless payment between parties: an issuer, a terminal, a card. Then, we give two security definitions for malicious cards and for malicious terminals in the adversarial and communication model that we define.

- We construct a contactless-payment protocol (ClessPay) which is secure against malicious cards and malicious terminals. ClessPay uses a distance bounding protocol to protect against relay attacks by malicious cards and MiM-adversaries. We proved formally the security of ClessPay in our security model.

- We analyze EMV-contactless protocol in our model. We give some vulnerabilities of EMV-contactless protocol against malicious cards. We prove the security of EMV-contactless protocol against malicious terminal formally. This type of formal cryptographic analysis is the first for EMV-contactless protocol.

Structure of the Chapter: In Section 7.2, we introduce the security model for CP where we consider security against malicious card and malicious terminal. Then, we give our new CP construction ClessPay together with its security analysis in Section 7.3. In Section 7.4, we analyze the contactless EMV protocol in our security model. We conclude this chapter with Section 7.5.

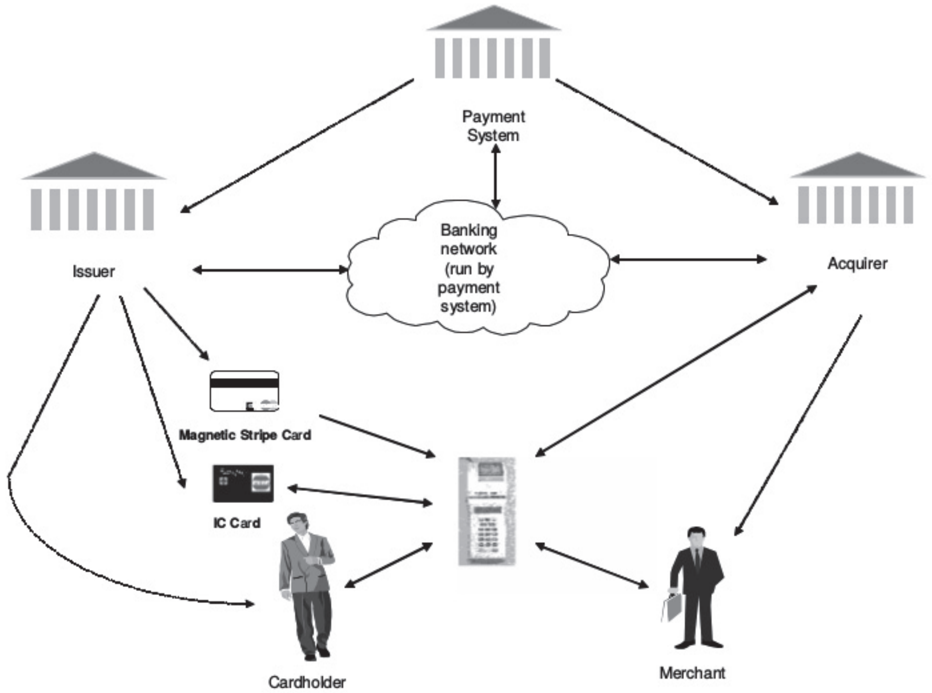

Figure 7.1 – Payment System [emva]

7.2 Security Model of Contactless Payment

According to the EMV specifications [emva], a (contactless) payment system consists of the following components:

Card Holder: It obtains the card from the issuer. It is responsible to present the card to the devices which accept payments.

Merchant: It obtains the payment terminal from the acquirer. It also contacts with the acquirer to receive reimbursements of the purchases by giving transmission details of the payments.

Acquirer: It sets up payment terminals when a merchant requests. It is responsible to pay the transactions to the corresponding merchants. After this, it communicates with the issuer to transmit the completed transactions.

Issuer: It issues a personalized (chip) card to the card holder. The cryptographic keys are installed to the cards by the issuer. The cards may contact with the issuer during the payment process (in online transactions) for the verification of the payment data. It also gives reimbursements of completed transactions to the acquirer. Each issuer has its policy function to approve or disapprove a transaction.

We assume that the issuer has a database *DataB* which stores the card information. *DataB* consists of tuples (Public Key, Card Information) of each card. Card information (CI) may consist of transaction list, the balance or the card limit.

Payment System: It is responsible to certify the issuer's public key and operate the online communication between the acquirer and the issuer.

Cards: They have a technology (e.g. NFC, Bluetooth) to communicate with a payment terminal without any contact. In a contactless payment, cards are the components which interact with the payment terminal to execute a payment with a certain amount. They include a unique card number consisting of a Primary Account Number (PAN) and an Expiration Date (ED). They also store a secret/public key pair in their tamper-resistant module and the issuer's public key. In this paper, we exclude card numbers for simplicity and we identify the cards with their public keys only.

Terminals: In a contactless payment, terminals interact with both cards and issuers via acquirers. They receive an order of payment from a card and validate the payment together with the issuer of the card.

Our following definitions include neither the certification process by the payment system nor the communication between merchant-acquirer and terminal-acquirer. We assume in the following definitions that the setup between payment components has been established. For the sake of simplicity, we assume the terminal represents both the terminal and the acquirer in the payment system and all cards are issued by one issuer.

Definition 7.1 (Contactless Payment)**.** *A contactless payment consists of algorithms for cards, terminals and issuers and a parameter B which is the distance bound. They respectively run the algorithms $C(\mathsf{sk}_C, \mathsf{pk}_C, \mathsf{pk}_I)$, $T(\mathsf{pk}_I, \tau_T)$ and $I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)$. Here, $(\mathsf{sk}_C, \mathsf{pk}_C)$ and $(\mathsf{sk}_I, \mathsf{pk}_I)$ are the secret/public key pair of C and I, respectively. They are generated by the algorithms $\mathcal{G}_C(1^\ell)$ and $\mathcal{G}_I(1^\ell)$ where ℓ is a security parameter. $DataB$ is the database for cards' information. I includes a subroutine $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ where CI represents the card information of a card with pk_C. In the end, I outputs $\mathsf{Out}_I \in \{0,1\}$ ($\mathsf{Out}_I = 0$ means cancel, $\mathsf{Out}_I = 1$ means accept the payment) and privately outputs $\mathsf{POut}_I = (\mathsf{pk}_C, id_I, \tau_I)$. Similarly, T outputs $\mathsf{Out}_T \in \{0,1\}$ ($\mathsf{Out}_T = 0$ means cancel, $\mathsf{Out}_T = 1$ means accept the payment) and private output $\mathsf{POut}_T = (\mathsf{pk}_C, id_T, \tau_T)$ and C privately outputs $\mathsf{POut}_C = (id_C, \tau_C)$. Here, τ is the transaction (τ_T, τ_I and τ_C are the values seen by the terminal , the issuer and the card), id is the identifier of the transaction (id_T, id_I and id_C are similarly defined).*

The algorithm Policy depends on the policy of the transaction approval by the issuer. Therefore, we can consider it as an algorithm which decides if a transaction τ_I is possible for the card with pk_C and CI.

We note that Out_I and $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ can be different. Out_I (similarly Out_T) shows the result of the contactless payment which can be either accepting the payment or canceling the payment. However, $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ shows only if the card with pk_C is able to do the payment. For example, even though the payment is canceled ($\mathsf{Out}_I = 0$) by the issuer, the issuer can approve the payment ($\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I) = 1$). It means that the card is able to this payment but the payment process is canceled (e.g., because of malicious behaviors).

Definition 7.2 (Correctness of Contactless Payment)**.** *We say that a contactless payment is correct for all B, transactions τ, database $DataB$, CI, and generated key pairs $(\mathsf{sk}_C, \mathsf{pk}_C)$ and $(\mathsf{sk}_I, \mathsf{pk}_I)$ if*

- *the algorithms C, T and I are run,*
- *T starts a transaction τ,*
- *there exists a C whose distance from T is at most B,*
- *(pk_C, CI) is in $DataB$ of an issuer I,*

then there exists an id such that

$$\Pr[(\mathsf{Out}_T = \mathsf{Out}_I = \mathsf{Policy}(\mathsf{pk}_C, CI, \tau)) \wedge (\mathsf{POut}_T = \mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)) \wedge (\mathsf{POut}_C = (id, \tau))] = 1.$$

Note that the output of T has to depend on the output of I because actually I is in the position to decide if the transaction is possible with the card (in fact an honest card cannot know if the transaction is possible).

Adversarial and Communication Model: In contactless payment, we consider the similar adversarial and communication model with the access control (AC) security model in Section 6.2.1 [KV17]. Remember that the parties in AC: a controller, a reader, a tag. They correspond to the parties contactless payment: an issuer, a terminal, a card, respectively. Differently than AC, in the adversarial model of contactless payment, terminals can be malicious. In a nutshell, the model is as follows:

- The communication between T and I is secure and authenticated. The adversary cannot attack this part of the communication.
- The communication between the parties is limited by the speed of light.
- All parties have polynomially many instances. An instance of a party is an execution of its corresponding algorithm at a given location. Instances of honest parties cannot be run in parallel.
- The adversaries can change the location of honest instances (but they move at a limited speed) or can activate them (See Section 6.2.1 for details).
- Adversaries create the database.
- Adversaries can change the destination of messages between a terminal and a card.

Definition 7.3 (Security in Contactless Payment with Malicious Cards)**.** *The security game is as follows:*

- *Run the key generation algorithms $\mathcal{G}_I(1^\ell) \to (\mathsf{sk}_I, \mathsf{pk}_I)$ and $\mathcal{G}_C(1^\ell) \to (\mathsf{sk}_{C_i}, \mathsf{pk}_{C_i})$ for the issuer and each card C_i and give the public keys to the adversary.*
- *The adversary creates instances of cards (C_i's) and the terminals at some locations of his choice. There is a distinguished terminal T (T is honest).*

- *The adversary sets a database **DataB** of the issuer. The issuer instance I which communicates with T is the distinguished issuer.*
- *The adversary creates the instances of himself (malicious cards or terminals) which can run independently and communicate together.*

We denote $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$ *and* $\mathsf{POut}_T = (\mathsf{pk}''_C, id_T, \tau_T)$ *the private outputs of* I *and* T*. Following our communication model, the game ends when* T *outputs* Out_T*. A contactless payment is secure, if the adversary wins this game with negligible probability. The adversary wins the game if* $\mathsf{Out}_T = 1$ *and at least one of the following conditions are satisfied:*

1. $(\mathsf{pk}'_C, .) \notin \boldsymbol{DataB}$,
2. $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and the distance between any* C *holding* pk'_C *and* T *is more than* B *during the execution of the protocol with* id_T,
3. $\mathsf{pk}'_C \notin \{\mathsf{pk}_{C_i}\}$ *and no instance of the adversary is close to* T *during the execution of the contactless payment protocol with* T *and* I.
4. $(\mathsf{pk}'_C, id_I, \tau_I) \neq (\mathsf{pk}''_C, id_T, \tau_T)$,
5. $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and there exists no card with* pk'_C *which privately outputs* (id_I, τ_I).

Remarks: The first winning condition shows that a card which does not belong to ***DataB*** should not authenticate. The second and the third conditions are to protect against MiM and DH (DF as well), respectively. Finally, the last two conditions are to be sure that the transaction that I and T approve and complete, and the transaction that I and an honest C approve and complete are the same.

Definition 7.4 (Security in Contactless Payment with Malicious Terminals)**.** *The security game is as follows:*

- *Run the key generation algorithms* $\mathcal{G}_I(1^\ell) \to (\mathsf{sk}_I, \mathsf{pk}_I)$ *and* $\mathcal{G}_C(1^\ell) \to (\mathsf{sk}_{C_i}, \mathsf{pk}_{C_i})$ *for the issuer* I *and each card* C_i *and give away public keys.*
- *The adversary creates instances of* C_i *and the terminals at some locations of his choice. There is a distinguished instance* I.
- *The adversary sets a database **DataB**.*
- *The adversary creates the instances of himself which can run independently and communicate together (as malicious cards or malicious terminals).*

At the end of the game I *outputs* Out_I *and* $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$*. A contactless payment is secure, if the adversary wins this game with negligible probability. The adversary wins the game:*

1. *if* $\mathsf{Out}_I = 1$ *and if at least one of the following conditions are satisfied:*
   (a) $(\mathsf{pk}'_C, .) \notin \boldsymbol{DataB}$,
   (b) $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and there exists no card with* pk'_C *which outputs* (id_I, τ_I),
   (c) $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and the instance of this card with* pk'_C *producing the output* (id_I, τ_I) *has a distance from the adversary and any honest terminal more than* B.

2. *or if there exists an honest-card instance with* $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$ *which privately outputs* $\mathsf{POut}_C = (id_C, \tau_C)$ *and there exists an issuer instance which has* $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_C) = 0$ *and* id_C*.*

The proximity condition (condition 1c) has not been considered by any of the payment systems before. Actually, even though we make sure the payment is completed successfully only when the terminal is close, we still cannot prevent a malicious terminal to execute a payment unbeknown to a card holder. For example, a malicious terminal can be moved close to a card while the card is not in the shop. This means the proximity condition does not prevent the malicious intention of the terminals. If we can be sure that the terminals can be run in a certain location, then we can guarantee the security against malicious terminals with the proximity condition. This can be possible by using proof of location in Chapter 9, but current terminals do not support this. For simplicity, we ignore integrating proof of location into our security model. Therefore, in our protocol, we eliminate 1c. We call **almost-secure against malicious terminals** if a protocol is secure without the proximity condition 1c in Definition 7.4.

The condition 2 is to prevent honest cards to make payment even though the issuer does not approve this payment. For example, this condition prevents attacks where malicious terminals make the honest cards execute payment (maybe without the knowledge of the honest card) for a big amount of money where normally the issuer would not let this amount be paid.

7.3 Contactless Payment Protocol

In this section, we construct a secure contactless-payment protocol from a public-key distance bounding $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$, an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and a signature scheme $(\mathsf{Sign}, \mathsf{Verify})$.

7.3.1 ClessPay

The protocol ClessPay (See Figure 7.2) starts after the terminal T creates a transaction τ and connects with a card C. We do not give the details of τ since it depends on the payment system. It may include the transaction details such as date, amount and currency.

In our protocol, we use signature schemes and an encryption scheme. Therefore, some secret/public key pairs are generated by using their key generation algorithms. More specifically, the key generation algorithm $\mathcal{G}_I$ generates a secret/public key pair $(\mathsf{sk}_I, \mathsf{pk}_I) = ((\mathsf{sk}_{I_s}, \mathsf{sk}_{I_e}), (\mathsf{pk}_{I_s}, \mathsf{pk}_{I_e}))$ where $(\mathsf{sk}_{I_s}, \mathsf{pk}_{I_s})$ is generated by the key generation algorithm of the signature scheme used by issuers and $(\mathsf{sk}_{I_e}, \mathsf{pk}_{I_e})$ is generated by the key generation algorithm of the encryption scheme. The key generation algorithm $\mathcal{G}_C$ generates a secret/public key pair $(\mathsf{sk}_C, \mathsf{pk}_C)$ using the key generation algorithm of the signature scheme used by cards. ClessPay consists of the following phases:

$I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)$		$T(\mathsf{pk}_I, \tau)$		$C(\mathsf{sk}_C, \mathsf{pk}_C, \mathsf{pk}_I)$
		Initialization		**pick** r
		$\mathcal{K}_V(1^\ell) \to (\mathsf{sk}_V, \mathsf{pk}_V)$	$\xrightarrow{\tau, \mathsf{pk}_V}$	$\mathcal{K}_P(1^\ell; r) \to (\mathsf{sk}_P, \mathsf{pk}_P)$
			$\xleftarrow{id, \mathsf{pk}_C}$	**pick** id
		$V(\mathsf{sk}_V, \mathsf{pk}_V) \to \mathsf{Out}_V, \mathsf{pk}_P$	$\xleftrightarrow{\mathsf{DB}}$	$P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$
		if $\mathsf{Out}_V = 0$: cancel		
		Approval		
$\phi = \exists(\mathsf{pk}_C, CI) \in DataB$	$\xleftarrow{\mathsf{pk}_C, \mathsf{pk}_P, id, \tau}$			
s.t. $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau) \to 1$				
if $\phi = \mathsf{False}$: cancel				
$S_I = \mathsf{sign}_{\mathsf{sk}_{I_s}}(id, \tau, \mathsf{pk}_C)$	$\xrightarrow{S_I}$		$\xrightarrow{S_I}$	**if** $\neg\mathsf{Verify}_{\mathsf{pk}_{I_s}}(S_I, id, \tau, \mathsf{pk}_C)$: cancel
		Completion		
				$S_C = \mathsf{sign}_{\mathsf{sk}_C}(id, \tau, r)$
$S_C, r = \mathsf{Dec}_{sk_{I_e}}(E_C)$	$\xleftarrow{E_C}$		$\xleftarrow{E_C}$	$E_C = \mathsf{Enc}_{\mathsf{pk}_{I_e}}(S_C, r)$
$\mathcal{K}_P(1^\ell; r) \to (\mathsf{sk}, \mathsf{pk})$				$\mathsf{POut}_C = (id, \tau)$
if $\neg\mathsf{Verify}_{\mathsf{pk}_C}(S_C, id, \tau, r)$ $\vee \mathsf{pk}_P \neq \mathsf{pk}$:cancel				
$\mathsf{Out}_I = 1$	$\xrightarrow{\mathsf{Out}_I}$	$\mathsf{Out}_T = \mathsf{Out}_I$		
		if $\mathsf{Out}_T = 0$: cancel		
$\mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)$		$\mathsf{POut}_T = (\mathsf{pk}_C, id, \tau)$		

Figure 7.2 – The ClessPay Protocol.

1. **Initialization Phase:** This phase is executed by T and C. If this phase cannot be completed successfully, then T cancels the transaction.

   T and C generate ephemeral secret/public key pairs for the distance bounding protocol $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$. For this, C first picks the random coins r and runs the deterministic algorithm $\mathcal{K}_P(1^\ell; r)$ to generate $(\mathsf{sk}_P, \mathsf{pk}_P)$. Here, what C does is equivalent to running $\mathcal{K}_P(1^\ell)$. C needs to generate the random coins used in $\mathcal{K}_P(1^\ell)$ because they will be needed in the last phase as a one-time proof for having generated pk_P. Then T runs $\mathcal{K}_V(1^\ell)$ to obtain $(\mathsf{sk}_V, \mathsf{pk}_V)$ used for distance bounding. T sends τ and pk_V to C. After receiving them, C picks an identifier id and replies with id and pk_C to introduce itself.

   T and C start the distance bounding protocol so that T determines the distance of C. Therefore, T runs the verifier algorithm $V(\mathsf{sk}_V, \mathsf{pk}_V)$ of DB and C runs the prover algorithm $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ of DB. At the end, V outputs Out_V which shows if C is close or not and private output $\mathsf{POut}_P = \mathsf{pk}_P$. If $\mathsf{Out}_V = 0$, then T cancels the transaction. Otherwise, they continue with the next phase. Remark that, T does not know yet if the card whose distance is determined is an authorized card because C has not authenticated itself with its (static) public key pk_C yet.

2. **Approval Phase:** This phase aims to check with the issuer whether the card can

execute the transaction. T first sends $\mathsf{pk}_C, \mathsf{pk}_P, id, \tau$ to I. I checks if the card with pk_C is in *DataB*. If it is in *DataB*, it retrieves the card information of the card (CI) and runs the algorithm $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau)$ which outputs 1 if the card has the privilege to execute τ[1]. If this algorithm returns 0, the transaction is canceled. Otherwise, I approves the transaction.

If it is approved, I signs with sk_{I_s} the message $(id, \tau, \mathsf{pk}_C)$. This signature is necessary for cards to be sure that they are approved for the payment. Then, it sends this signature S_I to T and T relays it to C. C runs the verification algorithm of the signature scheme $\mathsf{Verify}_{\mathsf{pk}_{I_s}}(S_I, id, \tau, \mathsf{pk}_C)$ to be sure that C and I have the same $(id, \tau, \mathsf{pk}_C)$. If C verifies S_I, then the next phase begins. Otherwise, C cancels the transaction.

3. **Completion Phase:** In this phase, the execution of the transaction τ with id is completed by I, T and C. First, C signs the message (id, τ, r) with sk_C as a proof of execution of the payment. The reason of signing r is showing that C took part in the distance bounding protocol. Then, it encrypts the signature S_C and r by using the key pk_{I_e}. The reason of the encryption is to hide r. At the end, C sends the encryption (E_C) to T. T relays it to I. At this point, the transaction is completed for C and it privately outputs (id, τ).

   In order to obtain S_C and r, I first decrypts E_C with sk_{I_e}. I verifies that r generates pk_P by running $\mathcal{K}_P(1^\ell; r)$. If it is verified, it also verifies S_C with $\mathsf{Verify}_{\mathsf{pk}_C}(S_C, id, \tau, r)$. If the signature is valid, then it sends $\mathsf{Out}_I = 1$ to T and privately outputs $(\mathsf{pk}_C, id, \tau)$. Otherwise, I cancels the transaction.

Cancel the transaction: As it can be seen in the protocol, the cancellation can be done by I, T or C. In the case of a timeout, parties cancel as well. When I cancels, it sets $\mathsf{Out}_I = 0$ and sends Out_I to T. Then, T cancels as well. When T cancels, it sets $\mathsf{Out}_T = 0$ and terminates. When C cancels, it sends a cancel message to T and terminates with $\mathsf{POut}_C = \perp$.

7.3.2 Security

Theorem 7.5. *Assuming that $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ is DF secure (Definition 2.8), DH-secure and MiM-secure, the encryption scheme is IND-CCA secure and the signature scheme used by cards is secure against existential-forgery, key-only message attacks (EF-MA) secure, ClessPay is secure against malicious cards (Definition 7.3).*

Proof. We assume that we have honest cards $\{C_1, C_2, ..., C_k\}$ and their public keys are in a set $\{pk_{C_i}\}$.

Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 7.3. There is a distinguished terminal instance $\mathcal{T}$ which privately outputs $\mathsf{POut}_T =$

[1] The Policy checks the execution right of a card depending on the bank policy. So, we do not discuss how this verification happens.

$(\mathsf{pk}''_C, id_T, \tau_T)$ and $\mathsf{POut}_V = \mathsf{pk}'_P$, and a distinguished issuer I which communicates with $\mathcal{T}$ and privately outputs $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$. Clearly, in Γ_0, the adversary cannot win with **the first condition** in Definition 7.3 ($(\mathsf{pk}'_C, .) \notin DataB$) because the issuer algorithm always cancels the transaction if $(\mathsf{pk}''_C, .) \notin DataB$.

Γ_1 : It is the same game as Γ_0 except that $(\mathsf{pk}'_C, id_I, \tau_I)$ is always equal to $(\mathsf{pk}''_C, id_T, \tau_T)$. Because of our secure and authenticated channel assumption between T and I and because of the honesty of $\mathcal{T}$, they have the same public key, the identifier and the transaction. Besides, $\mathcal{T}$ outputs 1, if I outputs 1. So, $p_1 = p_0$. In Γ_1, the adversary cannot win with **the fourth condition** in Definition 7.3 ($(\mathsf{pk}'_C, id_I, \tau_I) \neq (\mathsf{pk}''_C, id_T, \tau_T)$).

Γ_2 : It is the same game as in Γ_1 except that instances of honest cards do not sign and they encrypt a random message. Basically, each stores the ciphertext together with the identifier, transaction and static/ephemeral public keys to a table. I does not decrypt such random ciphertexts and retrieves their data from the table. More specifically, we simulate them as follows:

$\underline{C(\mathsf{sk}_C, \mathsf{pk}_C, \mathsf{pk}_I)}$	$\underline{I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)}$
same as in the protocol until signature generation	the same as in the protocol
pick R	**receive** E_C
$E_C = \mathsf{Enc}_{\mathsf{pk}_{I_e}}(R)$	**if** $(E_C, id, \tau, \mathsf{pk}_C, .) \in \mathsf{TableEnc}$:
store $(E_C, id, \tau, \mathsf{pk}, \mathsf{pk}_P)$ **in** TableEnc	**retrieve** pk **where** $(E_C, id, \tau, \mathsf{pk}_C, \mathsf{pk}) \in \mathsf{TableEnc}$
send E_C	**if** $\mathsf{pk} \neq \mathsf{pk}_P$:
$\mathsf{POut}_C = (id, \tau)$	cancel
	$\mathsf{Out}_I = 1, \mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)$
	else: as in the protocol after receiving E_C

We can show Γ_1 and Γ_2 are indistinguishable by using the IND-CCA security of the encryption scheme. For this, we first define a game Γ_{2,C_j} where we simulate only the instances of an honest card C_j's encryption as in Γ_2. Then, we define another game Γ^i_{2,C_j} where the first i instances of C_j which we denote $\{C^1_j, ..., C^i_j\}$ is simulated as in Γ_1 and the rest as in Γ_{2,C_j}. The reason of defining these games is to show that $\forall j \in \{1, 2, ..., k\}$ Γ^i_{2,C_j} and Γ^{i+1}_{2,C_j} are indistinguishable implying that Γ_1 and Γ_{2,C_j} are indistinguishable implying that Γ_1 and Γ_2 are indistinguishable.

To show the indistinguishability of Γ^i_{2,C_j} and Γ^{i+1}_{2,C_j}, we use an adversary $\mathcal{B}$ which plays IND-CCA game and simulates either Γ^i_{2,C_j} or Γ^{i+1}_{2,C_j} against an adversary $\mathcal{A}$ which distinguishes Γ^i_{2,C_j} and Γ^{i+1}_{2,C_j}. $\mathcal{B}$ gives the public key that it received from IND-CCA game to $\mathcal{A}$ as a public key of I. $\mathcal{B}$ simulates the first i instance of C_j and other honest cards' instances as in Γ_1 and the rest of C_j's instances as in Γ_{2,C_j} except C^{i+1}_j. It decrypts ciphertexts with using IND-CCA game. When $\mathcal{B}$ needs to simulate C^{i+1}_j, it generates the signature and gives one message which is the signature and the random coin (the message that it needs to be encrypted in Γ_1) and another message which is a random R to IND-CCA game. Then, $\mathcal{B}$ uses the challenge ciphertext received from IND-CCA game as an encryption generated by C^{i+1}_j. If IND-CCA game encrypts the first message $\mathcal{B}$ simulates Γ^i_{2,C_j} and if it encrypts the random message, $\mathcal{B}$ simulates Γ^{i+1}_{2,C_j}. So, if $\mathcal{A}$ succeeds to indistinguish the games, then $\mathcal{B}$ breaks the IND-CCA security. So, Γ^i_{2,C_j}

and Γ_{2,C_j}^{i+1} is indistinguishable. Using the hybrid argument, we can say that Γ_1 and Γ_{2,C_j} are indistinguishable. Since $\forall j \in \{1,2,...,k\}$ Γ_1 and Γ_{2,C_j} are indistinguishable, we can conclude that Γ_1 and Γ_2 are also indistinguishable. The reason of this conclusion comes from the fact that distinguishing Γ_1 and Γ_2 implies distinguishing Γ_1 and either one of Γ_{2,C_j}. So, $|p_2 - p_1|$ is negligible.

Remark that the random coins of the honest cards are not used in Γ_2.

Γ_3 : It is the same game as Γ_2 except that $\mathsf{Out}_V = 0$ after the execution of $V(\mathsf{sk}_V, \mathsf{pk}_V)$ if one of the situations happens:

1. no party is close to $\mathcal{T}$,
2. pk_P is generated by the adversary and there is no adversary close to $\mathcal{T}$,
3. pk_P belongs to an honest card instance but it has no instance close to $\mathcal{T}$.

Γ_3 and Γ_2 are indistinguishable because the probability that $\mathsf{Out}_V = 1$ if one of the situations above happens is negligible. $\mathsf{Out}_V = 1$ when the 1^{st} situation happens with negligible probability due to the DF-security of DB. $\mathsf{Out}_V = 1$ when the 2^{nd} situation happens with negligible probability due to the DH-security of DB. $\mathsf{Out}_V = 1$ when the 3^{rd} situation happens with negligible probability due to the MiM-security of DB. Note that we can simulate an honest card instance in Γ_3 by using a prover instance in MiM-game because the random coins are not used by honest card instances. Therefore, $|p_3 - p_2|$ is negligible.

Γ_4 : It is the same game as in Γ_3 except that I cancels after decrypting and obtaining the random coins r where $\mathcal{K}_P(1^\ell; r) \to (\mathsf{sk}_P, \mathsf{pk}_P)$ and $(\mathsf{sk}_P, \mathsf{pk}_P)$ is generated by an honest card instance.

$\underline{I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)}$
the same as in the protocol
receive E_C
if $(E_C, id, \tau, \mathsf{pk}_C, .) \in \mathsf{TableEnc}$:
  retrieve pk **where** $(E_C, id, \tau, \mathsf{pk}_C, \mathsf{pk}) \in \mathsf{TableEnc}$
  if $\mathsf{pk} \neq \mathsf{pk}_P$: cancel
  $\mathsf{Out}_I = 1, \mathsf{POut}_I = (\mathsf{pk}_C, id_T, \tau_T)$
else:
  $S_C, r = \mathsf{Dec}_{\mathsf{sk}_I}(E_C)$
  $\mathcal{K}_P(1^\ell; r) \to (\mathsf{sk}, \mathsf{pk})$
  if (sk, pk) is generated by an honest instance:
    cancel
  else: the same as in protocol after running $\mathcal{K}_P$

We can easily prove that if there exists an adversary with pk_C in Γ_3 which obtains a randomness r generating the secret/public key pair used by an honest instance, then we can construct another adversary which breaks the MiM-security of DB. Clearly, during the simulation of Γ_3, if I gets r, then it generates the corresponding secret key of the prover in MiM-game and breaks the MiM-security. Because receiving such r in Γ_4 happens with negligible probability, Γ_3 and Γ_4 are indistinguishable. So, $|p_4 - p_3|$ is negligible.

Now, we show that the adversary cannot win with **the third condition** in Γ_4. If the adversary wins with the third condition in Γ_4, then it means that $\mathsf{pk}'_C \notin \{\mathsf{pk}_{C_i}\}$ and no instance of the adversary is close to $\mathcal{T}$ during the execution of the contactless payment protocol with $\mathcal{T}$ and I. Due to the condition 2 in the reduction of Γ_3, pk_P must be generated by an honest card (otherwise, $\mathcal{T}$ cancels) . However, in Γ_4, it is not possible to have $\mathsf{Out}_I = 1$ while $\mathsf{pk}_C \notin \{\mathsf{pk}_{C_i}\}$ and pk_P is generated by an honest card instance (check the dashed underlined parts in the simulation of I in Γ_4). So, it is not possible that $\mathsf{Out}_I = 1$, if the game is in the third condition of Definition 7.3.

As conditions 2 and 5 of Definition 7.3 only remained to win in the game, we can assume that $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$.

Γ_5 : It is the same game as Γ_4 except we simulate Verify algorithm with Verify' such that it only accepts the signature of malicious cards. It does not accept the signatures of honest cards' instances as they never sign any message in this game.

$\underline{\mathsf{Verify}'_{\mathsf{pk}_C}(\boldsymbol{S}, \boldsymbol{id_I}, \boldsymbol{\tau_I}, \boldsymbol{r})}$
if $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$: **return** 0
else: return $\mathsf{Verify}_{\mathsf{pk}_C}(S_C, id_I, \tau_I, r)$

The only difference in Verify and Verify' is in the case of $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$. In this case, while Verify returns the output of the verification of the signature, Verify' returns 0. In Γ_5 and Γ_4, no honest cards' instances generate a signature. So, the only difference between Γ_4 and Γ_5 happens when I obtains a forged signature of an honest card instance.

Now, we show that forging a signature of any honest cards' instances happens with a negligible probability to prove that Γ_5 and Γ_4 are indistinguishable: We use an adversary $\mathcal{B}$ playing EF-MA game and simulating Γ_5 against the adversary $\mathcal{A}$. EF-MA game gives a public key pk_C. $\mathcal{B}$ picks one of the card C_i. In $\mathcal{B}$'s simulation, pk_C corresponds to the public key of C_i which will be seen by the distinguished issuer instance I. Therefore, in key generation of cards, $\mathcal{B}$ generates secret/public keys $(\mathsf{sk}_{C_j}, \mathsf{pk}_{C_j})$ for all honest cards except C_i. It gives $\{\mathsf{pk}_{C_j}\}$ and pk_C to $\mathcal{A}$. Remark that C_i never signs a message so we can simulate it perfectly. If the distinguished instance I does not receive pk_C, then $\mathcal{B}$ loses EF-MA game. Otherwise, at some point, if $\mathcal{B}$ receives a valid signature S, it first checks if S is verified with pk_C. If it is verified with pk_C, $\mathcal{B}$ outputs S to EF-MA game and wins. Otherwise, it loses. Clearly, the success probability of $\mathcal{B}$ is probability that $\mathcal{A}$ forges a signature divided by $\mathsf{poly}(k)$ where poly is a polynomial. Because of the EF-MA security of the signature scheme, the success probability of $\mathcal{B}$ is negligible. So, probability that $\mathcal{A}$ forges a signature is negligible, and Γ_5 and Γ_4 are indistinguishable meaning that $|p_5 - p_4|$ is negligible.

Remark that in Γ_5, I have $\mathsf{Out}_I = 1$, if and only if $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P)$ is in $\mathsf{TableEnc}$. So, we can assume that $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P) \in \mathsf{TableEnc}$.

If the adversary wins with the second condition in Γ_5, it means that $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and the distance between any C holding pk'_C and $\mathcal{T}$ is more than B during the execution of the protocol with id_T. Due to condition 3 in Γ_3, pk'_P should not been generated by this honest card. Then, $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P, .)$ cannot be in $\mathsf{TableEnc}$ which contradicts with our assumption. Hence, the adversary cannot win with **the second** condition.

If the adversary wins with the fifth condition, then it means that $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and there exists no card with pk'_C which privately outputs id_I, τ_I. Then, it means that $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P, .) \notin \mathsf{TableEnc}$ as no honest card instance has (id_T, τ_T). This contradicts with our assumption. Therefore, the adversary cannot win with **the fifth condition**.

Remark that in Γ_5, the adversary cannot win the game So, p_5 is negligible meaning that p_0 is negligible.

□

Theorem 7.6. *Assuming that the signature schemes used are existential forgery chosen message attack (EF-CMA) secure then ClessPay is* ***almost-secure*** *against malicious terminal (Definition 7.4).*

Proof. We recall that in almost-security, we do not need to consider condition 1c of Definition 7.4 .

Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 7.4. We have a distinguished issuer instance I which outputs $(\mathsf{pk}'_C, id_I, \tau_I)$. Remark that in Γ_0, the adversary cannot win **with condition 1*a*** ($(\mathsf{pk}'_C, .) \notin DataB$) because I rejects the cards which are not in $DataB$.

Γ_1 : It is the same game as Γ_2 except that no id selected by an honest card instance repeats. Clearly, $|p_1 - p_0|$ is negligible.

Γ_2 : It is the same game as Γ_1 except that we simulate I and its instances while generating the signature and honest cards' instances in the verification of this signature as follows:

$\boldsymbol{I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)}$	$\boldsymbol{\mathsf{Verify}'_{\mathsf{pk}_{I_s}}(S, id, \tau, \mathsf{pk}_C)}$
$S_I = \mathsf{sign}_{\mathsf{sk}_{I_s}}(id, \tau, \mathsf{pk}_C)$	**if** $(S, \mathsf{pk}_I, id, \tau, \mathsf{pk}_C)$ **in** $\mathsf{Table1}$
store $(S_I, id, \tau, \mathsf{pk}_C)$ **in** $\mathsf{Table1}$	**return** 1
send S_I	**else: return** 0

$|p_2 - p_1|$ is negligible.

The output of issuer instance is the same as issuer instances in Γ_1. Therefore, we have a perfect simulation for it. The only difference happens when honest cards' instances in Γ_1 receive a valid signature verified by pk_{I_s} and not in $\mathsf{Table1}$. In this case, honest cards in Γ_1 verify the signature but they do not in Γ_2. Otherwise, the simulations of them are perfect. We can easily show that the probability of generating a valid signature which is not in the $\mathsf{Table1}$ is negligible in Γ_2 thanks to EF-CMA security of the signature scheme. We can use the public key received from the signing game as a public key of the issuer and simulate signatures of issuer instances by using the signing game. Note that sk_{I_s} is not used in the simulation but the signature generation, so we can simulate the rest of the protocol perfectly. Therefore, $|p_2 - p_1|$ is negligible.

The adversary cannot win the game **with condition 2** in Definition 7.4 (there exists an honest card instance with $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$ which privately outputs $\mathsf{POut}_C = (id_C, \tau_C)$ and there exists an issuer instance which has $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_C) = 0$ and id_C). Assume that the adversary wins with condition 2. It implies that $(., id_C, \tau_C, \mathsf{pk}_C) \notin \mathsf{Table1}$ as id_C is unique. So, no honest card instance outputs (id_C, τ_C) in this case.

Γ_3 : It is the same game as Γ_2 except that we simulate honest cards' instances while generating the signature and I in the verification of this signature as follows:

$\boldsymbol{C(\mathsf{sk}_C, \mathsf{pk}_C, DataB)}$	$\mathsf{Verify''}_{\mathsf{pk}_C}\boldsymbol{(S, id_I, \tau_I, \mathsf{pk}_C, r)}$
$S_C = \mathsf{sign}_{\mathsf{sk}_C}(id, \tau, r)$	**if** $(S, \mathsf{pk}_C, id, \tau, r)$ **in** $\mathsf{Table2}$
store $(S_C, \mathsf{pk}_C, id, \tau, r)$ **in** $\mathsf{Table2}$	**return** 1
$E_C = \mathsf{Enc}_{\mathsf{pk}_{I_e}}(S_C, r)$	**else: return** 0
send E_C	

The only difference is the output of $\mathsf{Verify''}$ and Verify when a forged signature received. To show the indistinguishability of Γ_2 and Γ_3, we can use the similar reduction in the reduction of Γ_4 to Γ_5 in the proof of Theorem 7.5. The only difference in this reduction is using EF-CMA game and simulate the signature generation by using the signing oracle in EF-CMA game. We need a signing oracle here because we do not encrypt a random message instead of the signature as in Γ_5 in the proof of Theorem 7.5.

Remark that in this game, the adversary cannot win with **the condition 1*b*** ($\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and there exists no card with pk'_C which outputs (id_I, τ_I)). If I outputs $(\mathsf{pk}'_C, id_I, \tau_I)$, it means that an honest card instance with pk'_C added $(S, \mathsf{pk}'_C, id_I, \tau_I, .)$ in $\mathsf{Table2}$ and outputted (id_I, τ_I).

Hence, in Γ_3, the adversary cannot win. So, p_0 is negligible.

□

We recommend using Eff-pkDB [KV16] as a public-key distance bounding in ClessPay as we see in Chapter 4, it is the most efficient public-key distance bounding protocol having the necessary security requirements for ClessPay. It only requires one exponentiation and hashing.

The assumption on the signature scheme used by cards differ in Theorem 7.5 (EF-MA) and Theorem 7.6 (EF-CMA). Hence, it looks like to have security against both terminals and cards we need DF, DH, MiM-secure DB protocol, IND-CCA secure encryption scheme, and EF-CMA secure signature schemes. However, we could have the almost security against malicious terminal if we have the following assumptions in Theorem 7.6: the encryption scheme is IND-CCA secure and the signature scheme used by cards is EF-MA secure. In this case, the proof of Theorem 7.6 would need the same games Γ_2 and Γ_5 in the proof of Theorem 7.5 instead of Γ_3 in the proof of Theorem 7.6. So, actually, to have full security in ClessPay, we need DF, DH, MiM-secure DB protocol, IND-CCA secure encryption scheme, EF-CMA secure signature for issuers, and EF-MA secure signature for cards.

7.4 EMV Analysis

EMV key setting is different than our contactless-payment key setting because it has a symmetric key shared between the card and its issuer as well as asymmetric keys.

$\underline{I(S_I,P_I,MK_{AC})}$		$\underline{T(P_{IC})}$		$\underline{C(P_{IC},S_{IC},P_I,MK_{AC})}$
		Initialization		
		retrieve τ	$\xrightarrow{\tau}$	
			$\xleftarrow{\{ED,PAN,P_{IC}\}}$	
		Relay Resistance protocol		
		pick R_1	$\xrightarrow{R_1}$	pick R_2
		check RTT	$\xleftarrow{R_2,timings}$	
		Data Authentication and Transaction		
		pick UN_T	$\xrightarrow{UN_T}$	$ATC \leftarrow ATC++$
$SK_{AC} \leftarrow$ $Gen(MK,ATC)$ if $\mathsf{Verify}(ARQC,SK_{AC})$ $ARC = 1$ else: $ARC = 0$	$\xleftarrow{EARQC}$		$\xleftarrow{ARQC}$	$SK_{AC} \leftarrow Gen(MK,ATC)$ $ARQC =$ $\mathsf{MAC}_{SK_{AC}}(\mathsf{lbl}_{ARQC},UN_T,ATC,\tau)$
$ARPC =$ $\mathsf{MAC}_{SK_{AC}}(ARQC,ARC)$	$\xrightarrow{ARPC}$		$\xrightarrow{ARPC}$	if $ARC = 1$ and $\mathsf{Verify}(ARPC,SK_{AC})$: $TC = \mathsf{MAC}_{SK_{AC}}(\mathsf{lbl}_{TC},UN_T,ATC,\tau)$ **else:** $AAC = \mathsf{MAC}_{SK_{AC}}(\mathsf{lbl}_{AAC},UN_T,ATC,\tau)$ **pick** UN_C
		if $\mathsf{Verify}(P_{IC},SDAD)$: Perform approval **else:** Perform decline	$\xleftarrow{SDAD,TC}$	$SDAD =$ $\mathsf{sign}_{S_{IC}}(TC,UN_C,UN_T,R_1,R_2,timings)$

Figure 7.3 – The Simplified EMV protocol

An issuer I has a secret/public key pair S_I/P_I. It also has a master symmetric key MK_{AC}. A card C shares MK_{AC} with its issuer I. It has a secret/public key pair P_{IC} and S_{IC}. P_{IC} is signed by I's private key S_I. C stores certified P_I. We assume that the terminal T knows the public key of the certificate authority (CA) to verify P_I and so P_{IC}. We also assume that the channel between I and T is authenticated.

For the sake of simplicity, in Figure 7.3 and in our description, we assume that C knows all terminal related information such as TCC, authentication method and also terminal knows the card related information such as AID, PDOL and the elements in CDOL1 and CDOL2 (See Appendix D for EMV abbreviations). The full protocol is in Figure D.1 in Appendix D.

EMV contactless session consists of four phases without card holder (user) verification method (i.e., Online PIN, Signature):

- *Contact Establishment with NFC card:* T detects C.
- *Transaction Initialization:* T sends the transaction τ to C. Then, C responds with

its public key P_{IC} and card information such as PAN and expiration date (ED). If T verifies P_{IC}, it continues to the next phase.

- *Relay Resistance Protocol [emvb]:* This protocol is executed if C and T support it. Here, we assume that they support this feature. T picks a random number R_1 and sends this to C. C responds with another random number R_2. It also sends timing estimates ($timings$): Min Time For Processing Relay Resistance Protocol, Max Time For Processing Relay Resistance Protocol, Device Estimated Transmission Time For Relay Resistance Protocol. Then, T checks if the total time passed after sending R_1 exceeds the limit (let's call it B). If the total time does not exceed B, then the next phase begins. Otherwise, the transaction is canceled.

- *Data Authentication:* There are three type of authentication methods in EMV: Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined Data Authentication (CDA). Because of some weaknesses in SDA and DDA (replay attacks and wedge attacks), we consider CDA which is combined with the next phase.

- *Transaction:* T sends a random number UN_T to request a cryptogram generation from C. In EMV, there are three type of cryptograms: Transaction Certificate (TC), Authorization Request Cryptogram (ARQC), Application Authentication Cryptogram (AAC). Here, we consider the online verification where T requests ARQC for an online verification by the issuer. TC is used for the offline verification by the issuer and AAC is used to cancel the transaction.

  - *Online Verification:* C increases its counter ATC and generates a secret key SK_{AC} by using ATC and the master secret key MK_{AC}. Then, it generates the cryptogram $ARQC$: a MAC of UN_T, ATC, τ (list of objects in CDOL1) with using the secret key SK_{AC}. C sends the cryptogram AC to T and T relays it to I along with the card information. I verifies the MAC and possibly validate the information of C. If the cryptogram passes verification and the card is validated for the transaction, then I makes $ARC = 1$ and generates a MAC of $ARQC$ and ARC with the secret key SK_{AC}. This MAC is called as $ARPC$. After, it sends $ARPC$ with its message to T and T relays it to C if $ARC = 1$. Otherwise, it cancels the transaction.

    C verifies $ARPC$. If the verification and ARC is true then C generates the second cryptogram which is TC. TC is a MAC of CDOL2's objects with SK_{AC} (See [emvc], Table 26)[2] in order to show transaction is complete. Additionally, it picks a random number UN_C and generates a signature of UN_C, UN_T, ATC, TC, $timings, R_1, R_2$. C signs it with S_{IC} and sends the signature $SDAD$ and the signed message to T.

[2]Even if CDOL1 and CDOL2 list the same objects, some terminal related objects change because the payment process continues (e.g., TVR) [Rad03].

– Terminal checks if the signature and the data signed are valid. Later, the terminal contacts with the issuer to receive the reimbursement and gives TC as a proof of transaction completion by the card. In this case, the issuer verifies TC to execute the reimbursement.

EMV in Our Model: We use the following maps to match the EMV protocol with Definition 7.1:

$(\mathsf{sk}_C, \mathsf{pk}_C) = ((MK_{AC}, S_{IC}), P_{IC})$, $(\mathsf{sk}_I, \mathsf{pk}_I) = ((MK_{AC}, S_I), P_I)$, $id = ATC$, $\mathsf{Policy}(\mathsf{pk}_C, CI, id, \tau) = ARC$, $\mathsf{Out}_T =$ approval/ decline, $\mathsf{Out}_I = \mathsf{Verify}(TC, UN_T, ATC, \tau)$, $\mathsf{POut}_I = (P_{IC}, ATC, \tau)$, $\mathsf{POut}_T = (P_{IC}, ATC, \tau)$ and $\mathsf{POut}_C = (ATC, \tau)$.

7.4.1 Security Against Malicious Terminal in EMV

Clearly, the EMV protocol is not secure according to Definition 7.4 as the terminal can approve relay resistance protocol however close C is. However, it is almost-secure against malicious terminals. We prove it this in the following theorem. This proof is the first security proof for the EMV payment system.

Theorem 7.7. *Assuming that* MAC *is EF-CMA secure and* Gen *is a pseudo-random permutation, then EMV protocol is almost-secure against malicious terminals (Definition 7.4).*

Proof. Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 7.4. We have a distinguished issuer instance I which outputs (P_{IC}, ATC, τ_I).

In Γ_0, there exists at most one card instance with P_{IC} having ATC because ATC is a counter and incremented by each new card instance. Let's call this instance as $\mathcal{C}$.

Γ_1 : It is the same game with Γ_0 except that the honest card instances picks a random SK'_{AC} instead of generating it with $\mathsf{Gen}(MK', ATC)$ and stores the random SK'_{AC} in $\mathsf{Table1}$ as (MK', ATC', SK'_{AC}). If an issuer instance receives a card information belongs to an honest card then it retrieves SK'_{AC} from $\mathsf{Table1}$. As Gen is a pseudo-random permutation, $|p_1 - p_0|$ is negligible.

Γ_2 : It is the same game with Γ_1 except that we simulate MAC generation of honest cards and verification of MACs of honest cards' instances by the issuer as follows:

$\underline{I(P'_{IC}, S'_{IC}, P'_{I}, MK'_{AC})}$

$ATC' = ATC' + 1$
pick SK'_{AC}
store $(MK'_{AC}, ATC', SK'_{AC})$
$ARQC = \mathsf{MAC}_{SK'_{AC}}(\mathsf{lbl}_{ARQC}, UN_T, ATC', \tau)$
store $(SK'_{AC}, UN'_T, ATC', \tau', ARQC)$ in Table_{ARQC}
rest is the same until TC/AAC generation
if ARC = 1 **and** $\mathsf{Verify}(ARPC', SK'_{AC})$:
  $TC = \mathsf{MAC}_{SK'_{AC}}(\mathsf{lbl}_{TC}, UN_T, ATC', \tau)$
  store $(SK'_{AC}, UN'_T, ATC', \tau', TC)$ in Table_{TC}
else:
  $AAC = \mathsf{MAC}_{SK'_{AC}}(\mathsf{lbl}_{AAC}, UN_T, ATC', \tau)$
  store $(SK'_{AC}, UN'_T, ATC', \tau', AAC)$ in Table_{AAC}

$\underline{\mathsf{Verify}'(AC, SK_{AC})}$

if $(SK_{AC}, UN_T, ATC, \tau, AC)$ in Table_{AC}
  return 1
else: return 0

Γ_2 is indistinguishable from Γ_1 thanks to the security of MAC. The similar reduction in the proof of Theorem 7.5 from Γ_4 to Γ_5 can be used to prove the indistinguishably. So, $|p_2 - p_1|$ is negligible.

Γ_3 : It is the same game with Γ_2 except that I generates $ARPC$ and then stores it to Table_{ARPC} (similar storing as in Γ_2). Then, the honest cards verify $ARPC$ by checking if it is in the Table_{ARPC}. Γ_3 is indistinguishable from Γ_2 because of the security of MAC. So, $|p_3 - p_2|$ is negligible.

Clearly, in Γ_3, the adversary cannot win with the condition $1b$ because I privately outputs (P_{IC}, ATC, τ) if and only if the card with P_{IC} outputs ATC, τ.

In addition, it cannot win with the condition 2 because if $ARC \neq 1$, then no honest card outputs ATC, τ and if an honest card receives a valid $ARPC$ having $ARC = 1$, then it means that $ARPC$ is in Table_{ARPC}. So, I has (P_{IC}, ATC, τ).

Since the adversary cannot win in Γ_3, p_0 is negligible. □

However, there exists another problem in EMV related to ATC which we do not consider in our security definition. It can be explained as follows: ATC is 16-bit number and incremented at the beginning of each session. If ATC reaches the limit which 65535, then the card is not valid anymore because EMV specification does not let rotating the counter due to the security reasons. According to EMV specification [emvc] if cards are used normally, it will approach the limit (65,535) transaction limit not so fast (60 per day every day for a 3-year card). However, an attacker who does not aim to make a payment but aims to invalidate the card can trigger the card at most 65,535 times. Then, the card cannot be used anymore.

7.4.2 Security Against Malicious Card in EMV

Unfortunately, EMV is not secure against malicious cards. In the following, we show that an adversary can win with the second, third and fourth condition in Definition 7.3.

Fake Transaction Attack: This attack comes from the fact that T cannot validate TC in the signature $SDAD$ because it does not have SK_{AC}. Therefore, a malicious card can generate an invalid TC' in the last cryptogram generation process and use this cryptogram while generating this signature. Then, the terminal will approve the payment because the signature is correct. However, TC' is not valid. So, when T contacts with I, I cannot validate TC'. In this case, the malicious card succeeds to break the security of EMV with breaking **the fourth condition** in Definition 7.3 because I cancels while T does not.

Distance Fraud Attack: A malicious card can initiate a payment process with T, while it is not close T. In this case, it can send R_2 before seeing R_1 in order to reply early enough. In this case, T thinks that the card is close. Here, the malicious card succeed to break the security of EMV with breaking **the third condition** in Definition 7.3. This type of attack is dangerous for an EMV payment because the malicious card can claim later that it does not do the payment by showing that it was in somewhere else.

MiM Attack: The relay resistance protocol in EMV constructed to prevent relay attacks by a MiM-adversary. In this attack scenario, a MiM-adversary relays the messages between the card and the terminal to do the payment without the card's consent. Relay resistance protocol aims to prevent it by checking the distance of the card. The assumption on its security based on the fact that the adversary cannot relay the messages faster than the speed of light. Therefore, the adversary cannot succeed to pass the relay resistance protocol because it cannot guess R_2 before R_2 is picked by the card. However, it has been shown that with guessing attacks [CHKM06] the security against relay attacks is breakable for the protocols with single challenge/response exchanges. In addition, Chothia et al. [CGDR$^+$15] have already explained this vulnerability.

7.5 Conclusion

In this chapter, we concentrated on the formalism of contactless-payment system. In this direction, we first analyzed the components (issuers, terminals and cards) of a contactless-payment system from EMV specification [emva] which majority of contactless-payment systems follow. Then, we formally defined contactless payment by defining the inputs and outputs of the algorithms of issuers, terminals and cards. Based on this definition, we gave two security definitions against malicious cards and malicious terminals. We also considered relay attacks in our security definitions which are very common attacks in contactless payment.

We constructed a contactless-payment protocol ClessPay in our model. In this protocol, the terminal determines the distance of the card by using a secure public-key distance bounding protocol to prevent the relay attack and then the rest of the protocol continues with the authentication of the card and the issuer. We proved the security of ClessPay against malicious cards and malicious terminals formally.

Finally, we analyzed current EMV-contactless protocol [emvd] in our model. We realized that it is not secure against malicious cards because MiM-attack and DF-attack which are based on relay attacks. In addition to this, we formally proved that EMV-contactless protocol is secure against malicious terminals. Our analysis is the first formal cryptographic analysis of EMV-contactless protocol.

If we compare ClessPay and EMV contactless in regard to cryptographic computations executed by the cards, we see that EMV contactless is slightly more efficient since public-key operations are less in EMV contactless. A card in EMV contactless has to compute two MAC, verify one MAC and generate one signature. While a card in ClessPay has to compute one public-key encryption, generate one signature and verify one signature. However, to have the highest level of the security, it is the price to pay and with a dedicated hardware on smart cards, this price is not so high. As a future work, assuming that changing completely EMV specification is very hard, we can recommend some adaptations on EMV contactless to have full security without changing too much the basic structure of the protocol.

Part III

Positioning

Chapter 8

Formalism on Localization

A positioning system is used to help in determining the location of an object. Positioning systems are widely used in our world. Global Positioning System (GPS) is the most popular one among existing ones. Some applications of positioning systems such as military, emergency (e.g, medical), and prison are very critical because any inconsistent result may be very harmful. Therefore, the security of these systems becomes more of an issue because they are run in a malicious environment. In this chapter, we consider one of the problems related to positioning systems which is localization. In localization, a user aims to find its position by using a wireless network. We formally define the problem of localization and construct a formal security model. We describe algorithms and protocols for localization which are secure in our model.

Related Works Global Positioning System (GPS) [ME06] is a widely used positioning system consisting of satellites above earth. A GPS receiver on earth receives signals from at least four satellites and computes its distances from these satellites in order to locate itself. GPS is vulnerable to spoofing attacks [Sco01, NLD$^+$12, TPRČ11, PJ08] by impersonating the signals or delaying the signals. Kuhn [Kuh04] proposed to use asymmetric cryptography to have the integrity on signals of navigation systems (e.g., GPS). However, the problem related to delay of signals is not considered. Ranganathan et al. [RÓČ16] introduced SPREE which is a spoof resistant GPS receiver.

Some other positioning systems are based on wireless networks which work locally (e.g., indoor areas). Therefore, this type of positioning mechanisms is called localization in the literature. A localization system consists of multiple bases which help a user to locate itself. These bases know their own location and a user computes its own location by referencing the locations of bases. Bases are called beacon nodes, locators or anchors in the literature but we call as bases in this chapter. In such a system, the aim of an adversary is to make a user output a wrong location.

There are some techniques used to determine a location. We can divide them into two categories: range-dependent [LPČ05, NN03a, ČH05a, LND05b, SHS01] and range-independent [RL$^+$02, NN03b, SPS02, LP04, LP06] techniques. Range-dependent algorithms require measurement of distances by using the time of arrival as used in GPS, the time difference of signal arrival or the angle of arrival. In general, distance bounding (DB) [BC93] is the main method to measure distances in range-dependent algorithms. Range-independent algorithms do not require distance measurement. They estimate the location using properties of the networks such as hop counts, a topology of the network. Compared to range-dependent ones, their accuracy is low but they do not require a special hardware.

There exist many localization algorithms based on different assumptions. We mention here some important ones. For more details about previous works, there exist surveys by Srinivasan and Wu [SW07] and Zeng et al. [ZCH+13]. Lazos and Poovendran constructed a range-independent localization protocol SeRLoc [LP04] and HiRLoc [LP06] without malicious base assumption and with malicious base assumption, respectively. They have an analysis against wormhole attacks [ZCH+13] and sybil attacks [NSSP04]. In order to protect SeRLoc and HiRLoc against wormhole attack, they assume that the adversary cannot jam the communication which weakens their security model. Lazos et al. [LPČ05] constructed a protocol called ROPE by combining SeRLoc with a method called verifiable multilateration by Čapkun and Hubaux [ČH05a]. Verifiable multilateration uses distance bounding [BC93] in order to verify the computed location. Differently than SeRLoc, this protocol is not affected by communication jamming but still has a weaker security model because of the honest base assumption.

Liu et al. [LND05a] proposed a localization algorithm with using a detection mechanism to detect the cheating bases. They assume that there are some honest detecting bases which are indistinguishable from users. Liu et al. [LND05b] constructed a location-estimation scheme based on filtering the malicious bases. Zhang et al. [ZLFW06] proposed a method where bases execute multiple times distance bounding with a user to detect delays in signals by analyzing inaccurate changes in the distance bounding phase. So, they rely on some mistakes and malfunctioning on the adversary's side which is an underestimation of the adversary. In addition, they assume that the bases are honest. Čapkun et al. [ČČS06] introduced a model in which part of the bases are either hidden from users or mobile. This is meant to avoid the generic attack by preventing attackers to position properly for a localization attack to work. The model is relatively simple but Chandran et al. [CGMO09] showed that the locations of hidden bases can be discovered if a user is allowed multiple executions of the protocol and gets feedback on whether its position claim was accepted or not.

Differently, Zhong et al. [ZJUQ08] analyzed the tolerance of a robust localization algorithm against the maximum number of malicious bases. They proved that it is not always possible to accurately output a location if half of the bases or more than half of the bases are malicious. They proposed two localization algorithms. Both of them have cubic polynomial complexity in a two-dimensional space, but one of them has better average complexity. However, in their complexity analysis, they do not consider the complexity of testing collinearity although they have an assumption related to this. Comparing to Zhong et al. [ZJUQ08], we do not let our algorithms output a location if some malicious behaviors are detected. Instead, they can be rerun after filtering the malicious nodes for robustness. We have this approach because outputting a wrong location can have some bad consequences depending on the application. In addition, thanks to this difference, we have more tolerant localization algorithms against malicious bases.

Although there exists a lot of research on this topic, the security analyses are informal, or a specific attack-based (e.g., wormhole attack, sybil attack) or has no precise formal security model. For example, none of the mentioned protocols (except Zhong et al. [ZJUQ08]) take into account the collinearity of locations. Collinear locations of bases, as we discuss in Section 8.3.1, give a very good advantage to an adversary and even affect the correctness of the algorithm. The protocols [ČČS06, LPČ05, ZLFW06] using distance bounding do not discuss about the security requirement on DB. The distance bounding protocol suggested to use the verifiable multilateration method [ČH05a] is not secure against distance hijacking so it lets a malicious base shorten the actual distance. Therefore, the localization algorithms [LPČ05, ČČS06] using this method is not secure if this level of DB used since their security analysis is based on the assumption that the attacker cannot shorten the distance.

8.1 Our Contribution

In this chapter, we contribute to the formalism of localization. We define a security model and construct protocols. In more detail, our contributions can be enumerated as follows:

- We define two notions, non-interactive and interactive localization, with precise inputs and outputs. The former is given to have a concrete definition of a localization algorithm which outputs a location given some inputs. The latter is given to describe an interactive protocol between bases and a user who wants to learn its location. Our localization definitions based on trilateration method which uses distances to output a location.

- We integrate the adversarial and communication model of distance bounding with interactive localization. As we see in Part I, distance bounding is a well studied problem in cryptography [ABK$^+$11, DFKO11, BMV13a, BMV13b, BMV15, BV14]. Therefore, instead of defining a new model, we benefit from the DB model [BMV13a] that we give in Section 2.2.1. Accordingly, we define the security of non-interactive and interactive localization considering malicious bases. We have a stronger security model compared to previous work because we give more power to adversaries such as replacing honest bases and users to any place that they want and running them polynomially many times. This is a realistic power to be given to the adversary because an adversary can change the place of honest bases in the real life even though it cannot corrupt them.

- We analyze the circumstances under which we are guaranteed to find the correct locations of the users. Consequently, we prove that if at least half of the bases are honest then we can always output a correct location of a user.

- We construct three non-interactive localization algorithms with different levels of resistance against malicious bases. We give a general protocol for an interactive localization protocol which uses a secure non-interactive localization algorithm. We show that an instance of this generic protocol is secure against delays on communication for certain regions.

Structure of the chapter: In Section 8.2.1, we give formal definitions for localization and its security. In pursuit of this, we introduce distance estimate protocols in Section 8.2.2 which is used to output a distance of a party. In Section 8.3.1, we give three non-interactive localization algorithms and analyze their security. Then, in Section 8.3.2, we describe the framework for an interactive positioning with a security proof and show an instance of it with one of our non-interactive localization algorithms. We conclude this chapter with Section 8.4.

8.2 Definitions

Notations: We use $\mathbb{M}$ as an affine space of dimension t and d as the Euclidean distance. Given $loc_x \in \mathbb{M}$, $S(loc_x, d_x)$ is called t-sphere and defined as follows:

$$S(loc_x, d_x) = \{loc \in \mathbb{M} : d(loc_x, loc) = d_x\}$$

We use $Conv(loc_1, loc_2, ..., loc_m)$ to describe the convex hull constructed by $loc_1, loc_2, ..., loc_m \in \mathbb{M}$.

Independent locations: A set of locations are independent if and only if their combinations span the entire space $\mathbb{M}$. For this, we use a function $\mathsf{dep} : \mathbb{M}^m \rightarrow \{0,1\}$ to check if given m locations are dependent or not. If they are dependent it outputs 1. Otherwise, it outputs 0.

The function dep can be computed as follows: Let's take loc_1 as an origin and $\overrightarrow{loc_1 loc_2}, \overrightarrow{loc_1 loc_3}, ..., \overrightarrow{loc_1 loc_m}$ as vectors. The m locations are dependent if and only if these vectors are linearly dependent. For this, dep function can check if the rank of a matrix whose columns or rows are these vectors is less than or equal to $m-1$.

8.2.1 Localization

Localization aims at allowing a user to compute its location with the help of a number of bases. We define two variants: Non-Interactive Localization and Interactive Localization.

We call a location and distance pair (loc_{B_i}, d_i) is **correct** if the distance between the location of a user and loc_{B_i} is d_i.

Definition 8.1 (Non-Interactive Localization (NIL))**.** *An NIL consists of a parameter n, a metric space $\mathbb{M}$ and one probabilistic polynomial time (PPT) algorithm NIL. n corresponds to the number of bases $B_1, B_2, ..., B_n$ with respective locations $loc_{B_1}, loc_{B_2}, ..., loc_{B_n} \in \mathbb{M}$. NIL uses the locations of bases $loc_{B_1}, loc_{B_2}, ..., loc_{B_n}$ and the distance of these locations to the user's location $d_1, \ldots, d_n$, respectively, as an input. At the end, NIL outputs a location $loc_U \in \mathbb{M}$ or $\perp$.*

Correctness: NIL is correct for all $\mathbb{M}$, locations and distances, if all location/distance pairs are correct and there exists no $t+1$ dependent locations, NIL always outputs a correct location of the user.

Definition 8.2 (Interactive Localization (IL))**.** *An IL consists of a tuple $(\mathcal{K}_B, \mathcal{K}_U, B, U, n, \mathbb{M})$ where $\mathcal{K}_B$ and $\mathcal{K}_U$ are key generation algorithms, U is a user algorithm, B is a base algorithm, n is the number of bases and $\mathbb{M}$ is a metric space. $\mathcal{K}_B$ outputs a secret/public pair $(\mathsf{sk}_B, \mathsf{pk}_B)$ for the base algorithm B and $\mathcal{K}_U$ outputs a secret/public key pair $(\mathsf{sk}_U, \mathsf{pk}_U)$ for the user algorithm U. Each base B_i runs B interactively with the input $(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i}, \mathsf{pk}_U, loc_{B_i})$ where $loc_{B_i} \in \mathbb{M}$ is the location of a base B_i. A user runs U interactively with the input $(\mathsf{sk}_U, \mathsf{pk}_U, \mathsf{pk}_{B_1}, \mathsf{pk}_{B_2}, ..., \mathsf{pk}_{B_n})$. At the end, U outputs $loc_U \in \mathbb{M}$ or $\perp$.*

Correctness: An IL is correct for all secret/public key pairs, loc_{B_i}'s of which no $t+1$ are dependent, n and $\mathbb{M}$ if under honest execution of B and U, U always outputs a location loc_U which is the location of the user in $\mathbb{M}$.

NIL/IL and *robust* localizations differ with the outputs. Robust localization algorithms always output a location while the output of NIL/IL may be an abort message $\perp$. Zhong et al. [ZJUQ08] analyzes the maximum number of malicious bases which can offer resistance. By allowing the $\perp$ output, we obtain tolerance to a larger number of malicious bases. Indeed, we give an algorithm (NIL2 in Section 8.3.1) which securely works even if the number of malicious inputs is more than the half of the bases.

We first give the security of an NIL algorithm. In this definition, we cover that a secure NIL algorithm never outputs wrong location if at most k location/distance pairs are wrong.

Definition 8.3 (k-secure NIL)**.** *We define the security of NIL by a game. In this game, the adversary generates n-pairs (loc_{B_i}, d_i) and loc_U where each $loc_{B_i} \in \mathbb{M}$ and $loc_U \in \mathbb{M}$ and each number of $t+1$-locations are independent. Up to k pairs (loc_{B_i}, d_i) can be incorrect. The adversary wins if the NIL algorithm outputs loc'_U such that $loc'_U \neq \perp$ and $loc_U \neq loc'_U$. An NIL protocol is k-secure if for any such game an adversary cannot win.*

Remark that in k-security of NIL, we consider an NIL algorithm secure even if the algorithm outputs $\perp$. With the following definition, the adversary wins if it also achieves to make a NIL algorithm output $\perp$.

Definition 8.4 (k-full-secure IL). *The game is defined as in k-secure IL. The adversary wins if the NIL algorithm outputs loc'_U such that $loc_U \neq loc'_U$. An NIL protocol is k-full-secure if for any such game an adversary cannot win.*

Now, we give the adversarial and communication model that we consider for IL protocols.

Adversarial and Communication Model: We adopt the adversarial and communication model for distance bounding [BMV13a] described in Section 2.2.1. The provers in the DB model corresponds to bases in the localization model and the verifiers in the DB model corresponds correspond to the users in the localization model. Differently than DB, we have the following assumption in our model. Bases can retrieve their location correctly to input the algorithm B.

Definition 8.5 (k-secure IL). *We define the security of IL by a game. The game consists of a number of n-bases. In the game, the adversary chooses the corrupted bases. Let us denote the set of bases by $\mathcal{B}$ and the set of corrupted bases by $\mathcal{C} \subset \mathcal{B}$ with $|\mathcal{C}| \leq k$. Next, we generate secret/public key pairs of honest bases $\{(\mathsf{sk}_B, \mathsf{pk}_B)\}_{B \in \mathcal{B} \setminus \mathcal{C}}$ with $\mathcal{K}_B$ and secret/public key of user $(\mathsf{sk}_U, \mathsf{pk}_U)$. The adversary generates the secret/public key of corrupted bases $\{\mathsf{pk}_B\}_{B \in \mathcal{C}}$ by using $\{\mathsf{pk}_B\}_{B \in \mathcal{B} \setminus \mathcal{C}}$ and pk_U as an input. The adversary can create multiple instances of U and bases and it can move them any location in $\mathbb{M}$. At the end of the game, one instance of U outputs loc'_U. The adversary wins if $loc'_U \neq \perp$ and $loc_U \neq loc'_U$. IL protocol is k-secure if for any such game the probability of the adversary to win is negligible.*

In k-security game of IL, we let the adversary run user and base algorithms at any location multiple times by corrupting at most k-malicious bases. If one of the instances of U outputs a wrong location, then the adversary breaks the k-security. None of the previous work considers such security model. Differently, we give to the adversary the power of replacing honest instances as it wishes.

We note that we can also define k-full-secure IL, by replacing the winning condition $loc'_U \neq \perp$ and $loc_U \neq loc'_U$ with $loc'_U \neq loc_U$ only.

8.2.2 Distance Estimate Protocols

In range dependent localizations, a user algorithm U needs to know its distance relative to some known locations. Therefore, we need a protocol which outputs the distance between two locations. The idea is to use distance bounding protocols to provide U with the distance estimates. DB protocols [BC93] only output an accept/reject bit. Here, we define a slight variant of DB that we call distance estimate (DE) protocols. DE protocols output d, an upper-bound on the distance between V and the prover. We also adapt the MiM-security of DB to DE which will be used to prove the security of IL protocols based on distance estimate protocols.

Definition 8.6 (Public-key DE Protocol)**.** *A distance estimate protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple* $(\mathcal{K}_P, \mathcal{K}_V, V, P)$*.* $(\mathcal{K}_P, \mathcal{K}_V)$ *are the key generation algorithms of* P *and* V*, respectively. Their outputs are* $\mathcal{K}_P(1^\ell) \to (sk_P, pk_P)$ *and* $\mathcal{K}_V(1^\ell) \to (sk_V, pk_V)$*.* P *is the proving algorithm,* V *is the verifying algorithm where the inputs of* P *and* V *are* (sk_P, pk_P) *and* (sk_V, pk_V) *as described.* $P(sk_P, pk_P, pk_V)$ *and* $V(sk_V, pk_V)$ *interact with each other. At the end of the protocol,* $V(sk_V, pk_V)$ *outputs a final message* Out_V *which is either a distance or* $\perp$ *and a private output* $\mathsf{POut}_V = \mathsf{pk}_P$*.*

A DE protocol is correct if and only if under honest execution, whenever a verifier V *and a prover* P *lie at a distance* D *from each other,* V *always outputs* $Out_V = D$*.*

To define the security of DE protocols we, adapt the man-in-the-middle (MiM) security of DB to the security of DE.

Definition 8.7 (Modified MiM Security (mMiM-security))**.** *The game begins by running the key setup algorithms* $\mathcal{K}_V$ *and* $\mathcal{K}_P$*, which output* (sk_V, pk_V) *and* (sk_P, pk_P) *respectively. The adversary receives* pk_P *and* pk_V*. The game consists of several verifier instances including a distinguished one* V*, an honest prover* P *and an adversary. The adversary wins if* $Out_V \neq \perp$ *and there is no prover instance at distance* Out_V *or less from* V*. A DE protocol is mMiM-secure if for any such game, the adversary wins with negligible probability.*

We can easily derive a DB protocol from a DE protocol with the following transformation.

Definition 8.8 (T transformation)**.** *Let* $pDE = DE(\mathcal{K}_P, \mathcal{K}_V, P, V)$ *be a DE protocol. In the end,* V *outputs* d*: a distance estimate or abort message. We transform* pDE *into a DB protocol* $pDB = DB(\mathcal{K}_P, \mathcal{K}_V, P, V', B)$ *with the following verifier algorithm* V'*:*

$\underline{V'(\mathsf{sk}_V, \mathsf{pk}_V)}$

run $V(\mathsf{sk}_V, \mathsf{pk}_V) \to \mathsf{Out}_V, \mathsf{POut}_V$

if $\mathsf{Out}_V \neq \perp$ *and* $\mathsf{Out}_V \leq B$*:*

  output $\mathsf{Out}_V = 1$

else:

  output $\mathsf{Out}_V = 0$

We use the notation $T(pDE, B) = pDB$ *to show this transformation.*

It may not be always possible to have a transformation from a DB protocol to a DE protocol. However, we can show that the MiM-security of the transformed DB protocol and the mMiM-security of DE protocol is equivalent.

Theorem 8.9 (MiM-security $\Leftrightarrow$ mMiM-security)**.** *A DE protocol* $pDE = DE(\mathcal{K}_V, \mathcal{K}_P, V, P, B)$ *is mMiM-secure according to Definition 8.7 if and only if for any* B*,* $T(pDE, B)$ *is MiM-secure (Definition 2.12).*

Proof. **mMiM-secure** $\Rightarrow$ **MiM-secure**: Consider a MiM game Γ for $pDB = T(pDE, B)$. We define the mMiM-game Γ' for pDE by simulating Γ. Whenever Γ succeeds, we have

$Out_V < B$ and no honest prover instance at a distance up to B. So, Γ' succeeds as well and $\Pr[\Gamma \text{ succeeds}] \leq \Pr[\Gamma' \text{ succeeds}]$. If *pDE* is mMiM-secure, $\Pr[\Gamma' \text{ succeeds}]$ is negligible. So, $\Pr[\Gamma \text{ succeeds}]$ is negligible.

mMiM-secure $\Leftarrow$ MiM-secure: Consider an mMiM game Γ' for *pDE*. Let B be the distance from V to the closest instance. Consider the same game Γ for $pDB = T(pDE, B)$. Whenever Γ' succeeds, we have $Out_V < B$ so Γ succeeds as well. Thus, $\Pr[\Gamma' \text{ succeeds}] \leq \Pr[\Gamma \text{ succeeds}]$. If *pDB* is MiM-secure, those probabilities are negligible. □

8.3 Localization Protocols

8.3.1 Non-Interactive Localization

In a t-dimensional $\mathbb{M}$, given number of $t+1$ *independent* locations $loc_{B_1}, loc_{B_2}, ..., loc_{B_{t+1}} \in \mathbb{M}$ and their corresponding distances $d_1, d_2, ..., d_{t+1}$ to a certain location $loc_U \in \mathbb{M}$, we can compute loc_U as

$$I = \{loc_U\} = S(loc_{B_1}, d_1) \cap S(loc_{B_2}, d_2) \cap ... \cap S(loc_{B_{t+1}}, d_{t+1}).$$

We can see that for all $i \in \{1, 2, ..., t+1\}$, $loc_U \in S(loc_{B_i}, d_i)$. Therefore, it is clear that we can find loc_U by intersecting spheres. Now, we show why we need at least $t+1$ spheres.

Lemma 8.10. *In a t-dimensional Euclidean space $\mathbb{M}$, the intersection of $t+1$ spheres with independent centers has a cardinality at most 1.*

Proof. Let's take loc_{B_1} as an origin and let $x = \overrightarrow{loc_{B_1} loc_{loc_U}}$ be a vector which is equivalent to loc_U. We have the following equation:

$$\begin{aligned}
d_i^2 - d_1^2 &= d(loc_U, loc_{B_i})^2 - d(loc_U, loc_{B_1})^2 \\
&= ||\overrightarrow{loc_U loc_{B_i}}||^2 - ||\overrightarrow{loc_U loc_{B_1}}||^2 \\
&= ||\overrightarrow{loc_{B_1} loc_U} - \overrightarrow{loc_{B_1} loc_{B_i}}|| - x \cdot x \\
&= (x - \overrightarrow{loc_{B_1} loc_{B_i}}) \cdot (x - \overrightarrow{loc_{B_1} loc_{B_i}}) - x \cdot x \\
&= \overrightarrow{loc_{B_1} loc_{B_i}} \cdot (\overrightarrow{loc_{B_1} loc_{B_i}} - 2x)
\end{aligned} \tag{8.1}$$

So, x is a solution of system of equations as in Equation (8.1) for all $i \in \{2, 3, ..., t+1\}$. All $\overrightarrow{loc_{B_1} loc_{B_i}}$'s are linearly independent vectors since $loc_{B_1}, loc_{B_2}, loc_{B_3}, ..., loc_{B_{t+1}}$ are independent. Therefore, this system of equations can have at most one solution x.

□

Note that for the intersection of t spheres, the linear system gives a line. The intersection between a line and a sphere has cardinality limited to two. Thus, the intersection of t-spheres with independent centers gives at most two points.

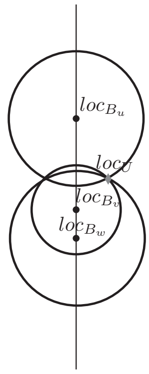

Figure 8.1 – Dependent locations loc_{B_u}, loc_{B_v} and loc_{B_w} in 2-dimensional metric space. The spheres intersect on two points so it is not possible to decide which one is loc_U.

The NIL algorithm is essentially finding the intersection of $t+1$ spheres. However, the crucial point of an NIL algorithm is to see if we can find the correct location loc_U with an NIL algorithm, which takes n-location/distance pairs as input, given that k of those are wrong. Clearly, if $n = t+1$ and $k > 0$, it is not possible to have a correct NIL algorithm. Therefore, $n > t+1$ is a necessary requirement. However, we need to know what is the requirement on k in order to obtain loc_U with a correct NIL algorithm.

The following lemma shows under which circumstances we can be sure that we obtain loc_U from the possible intersection points.

Lemma 8.11. *Given number of n spheres $\{S(loc_{B_j}, d_j)\}_{i=1}^{n}$ where at least $n-k$ of them include a location $loc_U \in \mathbb{M}$ (equivalently, the distance between loc_{B_j} and loc_U is d_j), let us define the following score for all $loc_x \in \mathbb{M}$:*

$$\#i(loc_x) = |\{i \in \{1,\ldots,n\}\,|loc_x \in S(loc_{B_i}, d_i)\}|.$$

Given that any number of $t+1$ of loc_{B_i}'s are independent and $n \geq 2k+t$, then for any $loc_x \in \mathbb{M}$, we have the following results:

- *if $loc_x = loc_U$, then $\#i(loc_x) \geq \frac{n+t}{2}$ and*
- *if $\#i(loc_x) > \frac{n+t}{2}$, then $loc_x = loc_U$.*

Proof. Let's analyze the cardinality of $\#i(loc_x)$ for $loc_x \neq loc_U$ and $loc_x = loc_U$.

If $loc_x = loc_U$, from the assumption, $\#i(loc_U) \geq n-k$.

If $loc_x \neq loc_U$, $loc_x \in S(B_j, d_j)$ for at most number of t spheres which also include loc_U due to Lemma 8.10. Besides, if $loc_x \neq loc_U$, $loc_x \in S(B_j, d_j)$ for at most number of k spheres which do not include loc_U. Thus, if $loc_x \neq loc_U$, $\#i(loc_x) \leq k+t$.

- We prove that if $loc_x = loc_U$ then $\#i(loc_x) \geq \frac{n+t}{2}$:

  If $loc_x = loc_U$, we know that $\#i(loc_U) \geq n-k$. Since $n \geq 2k+t$ implies that $n-k \geq \frac{n+t}{2}$, we can conclude that $\#i(loc_U) \geq n-k \geq \frac{n+t}{2}$.

- We prove that if $loc_x \neq loc_U$, then $\#i(loc_x) \leq \frac{n+t}{2}$: Since $n \geq 2k+t$ implies that $k+t \leq \frac{n+t}{2}$, $\#i(loc_x) \leq k+t \leq \frac{n+t}{2}$.

□

Lemma 8.11 implies that $loc_x = loc_U$ is equivalent to $\#i(loc_x) > \frac{n+t}{2}$ when any number of $t+1$ loc_{B_i}'s are independent and $n > 2k+t$. If $n+t$ is odd $loc_x = loc_U$ is equivalent to $\#i(loc_x) > \frac{n+t}{2}$ when any number of $t+1$ loc_{B_i}'s are independent and $n \geq 2k+t$

Using the result of Lemma 8.11, we construct an NIL-1 algorithm.

NIL-1: We describe NIL-1 in Algorithm 1. Here is an overview of it.

Let $\mathbb{L} = \{loc_{B_j}\}_{j=1}^{n}$ be the set of different positions in $\mathbb{M}$ and let $\{d_j\}_{j=1}^{n}$ be the distance between loc_{B_j} and a location $loc_U \in \mathbb{M}$, $n > t$, $n \geq 2k+t$ if $\frac{n+t}{2}$ is odd and $n > 2k+t$ if $\frac{n+2}{2}$ is even.

With each location/distance pairs (loc_{B_u}, d_u), NIL-1 outputs either loc_U or abort message $\perp$ by trilateration using spheres.

In order to guarantee a certain security level, NIL-1 has to check independence of each number of $t+1$-locations. Therefore, it first runs a dependency test function dep. If dep outputs 1 for a $t+1$ location tuple, the NIL-1 aborts and outputs $\perp$. As the security can be corrupted with dependent locations, the algorithm does not continue.

If there exists no dependent $t+1$ locations, it continues as follows: To avoid enumerating every sphere, it starts by first picking t different locations of bases $loc_{B_{u1}}, loc_{B_{u2}}, \ldots, loc_{B_{ut}} \in \mathbb{L}$ at random such that the intersection of the spheres $\{S(loc_{B_{ui}}, d_{ui})\}_{i=1}^{t}$ is not empty. At this point, the intersection I includes at most two locations. Then, NIL-1 keeps track of the number of spheres which are different than $\{S(loc_{B_{ui}}, d_{ui})\}_{i=1}^{t}$ and which include the location(s) in I. Whenever it finds a location which is on more than $\frac{n+t}{2}$ spheres, then it outputs this location as loc_U.

Correctness: If all inputs are correct and there exists no $t+1$ dependent locations, then $loc_U \in I$ at the first iteration of the for all loop. So, $count[loc_U] = t$ at this point. Then, the algorithm continues to intersect I with the rest of spheres. As all location/distance pairs are correct and locations are independent, the rest of the intersections include only loc_U. Therefore, $count[loc_U] = n$. Since, we have $n > \frac{n+t}{2}$, NIL-1 always outputs loc_U.

Complexity: We consider the dependency test and the intersection computation in our complexity analysis. Dependency test and intersection are counted as $O(t^3)$ which is the complexity of classical Gaussian Elimination. Therefore, the best, worst and expected complexity is $O(n^{t+1}t^3)$ which is a polynomial. In real life cases, dimension two or three is used so the complexity is $O(n^3)$ and $O(n^4)$, respectively.

Theorem 8.12 (k-security of NIL-1)**.** *If $n \geq 2k+t$, NIL-1 is k-secure (as defined in Definition 8.3).*

Algorithm 1 NIL-1($loc_{B_1}, \ldots, loc_{B_n}, d_1, \ldots, d_n$)

for all possible $t+1$-tuple $(loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut+1}})$ **do**
  if $\mathsf{dep}(loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut+1}}) \to 1$ **then**
    return $\perp$
  end if
end for
for all possible t-tuple $(loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut}})$ **do**
  $I \leftarrow \cap_{i=1}^{t} S(loc_{B_{ui}}, d_{ui})$ (comment: $|I| \leq 2$)
  if $I \neq \emptyset$ **then**
    for $loc_x \in I$ **do**
      $\text{count}[loc_x] = t$
    end for
    for $loc_{B_{ut+1}} \in \mathbb{L} \setminus \{loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut}}\}$ **do**
      $\{loc_x\} \leftarrow I \cap S(loc_{B_{ut+1}}, d_{ut+1})$ (comment: $|\{loc_x\}| \leq 1$)
      if $\{loc_x\} \neq \emptyset$ **then**
        $count[loc_x] \leftarrow count[loc_x] + 1$
        if $count[loc_x] > \frac{n+t}{2}$ **then**
          return loc_x
        end if
      end if
    end for
  end if
end for
return $\perp$

Proof. Assume that there exists $loc_x \neq loc_U$ such that $\text{count}[loc_x] > \frac{n+t}{2}$ (which is the only case that an adversary wins). Remark that $\frac{n+t}{2} < count[loc_x] \leq \#i(loc_x)$. From Lemma 8.11, we know that given $n \geq 2k+t$ and independence of every $t+1$ location, if $\#i(loc_x) > \frac{n+t}{2}$, then $loc_x = loc_U$ which contradicts our assumption. $\square$

Theorem 8.13 (k-full-security of NIL-1)**.** *If $n \geq 2k+t$ and $\frac{n+t}{2}$ is odd, NIL-1 is k-full-secure (as defined in Definition 8.4) and if $n > 2k+t$ and $\frac{n+t}{2}$ is even, NIL-1 is k-full-secure (as defined in Definition 8.4).*

Proof. We need to prove that NIL-1 always outputs loc_U. We prove in Theorem 8.12 that if NIL-1 outputs a location $loc_x \neq \perp$ then $loc_x = loc_U$.

Now, we show that NIL-1 never outputs $\perp$. Let's assume that NIL-1 outputs $\perp$. It means that for all $loc_x \in \bigcup I$, $\#i(loc_x) = count[loc_x] \leq \frac{n+t}{2}$.

- If $n \geq 2k+t$ and $\frac{n+t}{2}$ is odd, $n-k \geq \frac{n+t}{2}$. Since $\frac{n+t}{2}$ is an odd number, actually, $n-k > \frac{n+t}{2}$ because $n-k$ is a positive integer. We also know from Lemma 8.11 that $\#i(loc_U) \geq n-k > \frac{n+t}{2}$ which contradicts with our assumption.

- If $n > 2k+t$ and $\frac{n+t}{2}$ is even, $n-k > \frac{n+t}{2}$. We also know from Lemma 8.11 that $\#i(loc_U) \geq n-k > \frac{n+t}{2}$ which contradicts our assumption.

$\square$

NIL-1 is secure and always returns the right location (if there are no dependent $t+1$ locations) as long as $n > 2k+t$ and $\frac{n+t}{2}$ is even or $n \geq 2k+t$ and $\frac{n+t}{2}$ is odd. We now propose another algorithm whose resistance to corrupted pairs is higher but which returns $\perp$ as soon as malicious inputs are detected.

NIL-2: We give NIL-2 in Algorithm 2. NIL-2 first has to check the independence of each $t+1$-locations as NIL-1. Therefore, it first runs a dependency test function dep. If dep outputs 1 for a $t+1$-location tuple, NIL-2 aborts and outputs $\perp$. As the security can be corrupted with dependent locations, the algorithm does not continue.

If there exists no dependent $t+1$ locations, it picks a $t+1$-location tuple at random and intersects the spheres constructed from these location. Note that the intersection has at most one location. Then, NIL-2 checks if the rest of the spheres includes the location in this intersection. If one sphere does not include it, it outputs $\perp$. Otherwise, it outputs the location in the intersection.

Algorithm 2 NIL-2($loc_{B_1}, \ldots, loc_{B_n}, d_1, \ldots, d_n$)

for all possible $t+1$-tuple $(loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut+1}})$ **do**
    if $\mathsf{dep}(loc_{B_{u1}}, loc_{B_{u2}}, ..., loc_{B_{ut+1}}) \to 1$ **then**
        return $\perp$
    end if
end for
pick a tuple $(loc_{B_{w1}}, loc_{B_{w2}}, ..., loc_{B_{wt+1}})$
$\{loc_x\} \leftarrow \cap_{i=1}^{t+1} S(loc_{B_{wi}}, d_{wi})$ (remark that $|\{loc_x\}| = 1$ or $I = \emptyset$)
if $\{loc_x\} = \emptyset$ **then**
    return $\perp$
end if
for all $loc_{B_i} \in \mathbb{L} \setminus \mathsf{tuple}$ **do**
    if $loc_x \notin S(loc_{B_i}, d_I)$ **then**
        return $\perp$
    end if
end for
return loc_x

Correctness: If all inputs are correct and there exists no $t+1$ dependent locations, clearly, the algorithm always outputs loc_U.

Complexity: The worst case and the best case complexity of NIL-2 is $O(n^{t+1}t^3)$. Here, we consider dependency test, intersection computation and checking if a sphere contains location in our complexity analysis. In real life cases, dimension two or three is used so the complexity is $O(n^3)$ and $O(n^4)$, respectively.

Theorem 8.14 (k-security of NIL-2)**.** *If $n > k+t$, NIL-2 is k-secure (as defined in Definition 8.3).*

Proof. Assume that NIL-2 is not k-secure when $n > k+t$. So, the adversary wins k-security game for NIL-2. In line 7 of Algorithm 2, if $I = \{loc_U\}$, then the adversary cannot win. So, we can assume that NIL-2 computes $I = \{loc'_U\}$ where $loc_U \neq loc'_U$ when the adversary wins.

NIL-2 outputs loc'_U if all spheres include it. From Lemma 8.11, we know that we can have at most t correct spheres which include loc'_U. So, if $n > k+t$, there exists a sphere which does not include loc'_U and NIL-2 outputs $\perp$. Therefore, the adversary cannot win k-security game when $n > k+t$. This contradicts our assumption. So, NIL-2 is k-secure when $n > k+t$.

□

NIL-2 is not k-full-secure because if there exists an incorrect location whose sphere does not intersect with any other spheres, then NIL-2 outputs $\perp$.

We give a variant of NIL-2 which has lower computational complexity and have the same security level *with an extra assumption*.

NIL-3: We give NIL-3 in Algorithm 3 with the assumption that $n > k+t$ and any $t+1$-locations which have correct distances are independent. NIL-3 first picks $t+1$-locations. If they are dependent, then it outputs $\perp$. Otherwise, it intersects the spheres constructed from $t+1$-independent location-distance pairs. If the intersection is not empty, then it checks if the spheres constructed from other location-distance pairs includes the location in the intersection. If all spheres include, NIL-3 outputs this location. Otherwise, it outputs $\perp$.

Algorithm 3 NIL-3($loc_{B_1}, \ldots, loc_{B_n}, d_1, \ldots, d_n$)

pick a tuple $loc_{B_{u1}}, loc_{B_{u2}}, \ldots, loc_{B_{ut+1}} \in \mathbb{L}$
if $\mathsf{dep}(loc_{B_{u1}}, loc_{B_{u2}}, \ldots, loc_{B_{ut+1}}) \to 1$ **then**
  return $\perp$
end if
$\{loc_x\} \leftarrow \bigcap_{loc_{B_{ui}} \in \mathsf{tuple}} S(loc_{B_{ui}}, d_{ui})$ (remark that $|\{loc_x\}| = 1$ or $I = \emptyset$)
if $\{loc_x\} = \emptyset$ **then**
  return $\perp$
end if
for all $loc_{B_i} \in \mathbb{L} \setminus \mathsf{tuple}$ **do**
  if $loc_x \notin S(loc_{B_i}, d_I)$ **then**
    return $\perp$
  end if
end for
return loc_x

Theorem 8.15 (k-security of NIL-3)**.** *Assuming that any $t+1$-locations which have correct distances are independent and $n > k+t$, NIL-3 is k-secure (as defined in Definition 8.3).*

Proof. The proof is very similar to the proof of Theorem 8.14. The only case that the adversary can win the k-security game is when $\mathsf{tuple} = \{loc_x\} \neq \{loc_U\}$. Therefore, assume that NIL-3 is in this case. NIL-3 outputs loc_x if all spheres includes it. From Lemma 8.11, we know that we can have at most t correct spheres which include loc_x as any $t+1$-locations which have correct distances are independent. So, if $n > k+t$, there exists a sphere which does not include loc_x and NIL-3 outputs $\perp$.

□

Correctness: If all inputs are correct and there exists at least one $t+1$ independent locations, clearly, the algorithm always outputs loc_U.

Complexity: In our complexity analysis, we take into account the dependency test, the intersection computation and checking if a sphere includes a location. The complexity of NIL-3 is $O(t^3 + nt)$ because it does one dependency check and checks whether number of n-spheres include loc_x.

Remarks: NIL-3 is the most efficient algorithm comparing to NIL-1 and NIL-2 as long as there exists at least one correct $t+1$-independent location/distance. NIL-3 can be useful if location of bases are designated considering their independency. However, if it is not the case, NIL-2 and NIL-1 are better options. NIL-1 is k-full secure algorithm while NIL-2 is secure (not full) with more incorrect pairs. However, it does not correct bad inputs as much as NIL-1 does. It rather outputs $\perp$ instead of trying to correct. Therefore, differently than robust localization [ZJUQ08], with our localization definition (Definition 8.1), we can achieve higher security. By using NIL-2, we give an IL protocol which is secure in a given area with unlimited delay attack.

8.3.2 Interactive Localization

A user algorithm in an IL protocol does not have locations of bases and distances between bases' locations and the user's location as an input. However, they can be deduced during the protocol. Once U obtains locations and distances, it can use a k-(full)-secure NIL algorithm (e.g., Algorithm 1, 2 and 3) which outputs the location of the user.

Apparently, a secure DE protocol can be used to learn the distances once U learns the locations of bases. However, if some locations are not correctly obtained (due to dragging attacks) or if some precise delays are introduced during the DE protocol execution (by delay attacks), then security problems occur. In more details, these attacks which cause problems on security are as follows:

Delay Attack on DE (See Figure 8.2 and 8.3): It is not possible to prevent delays on arrival of messages in a DE protocol. If a distance computation is based

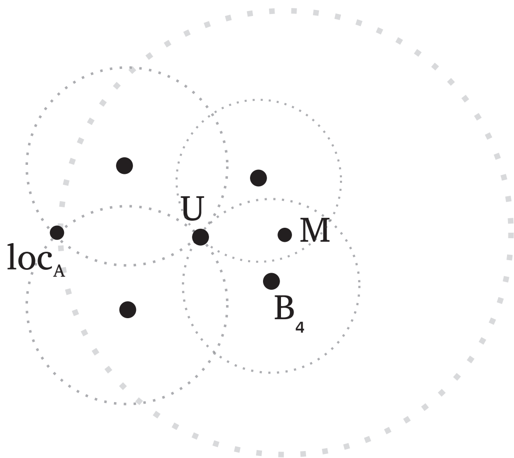

Figure 8.2 – No adversary delays. Points without label are honest bases. B_4 is honest.

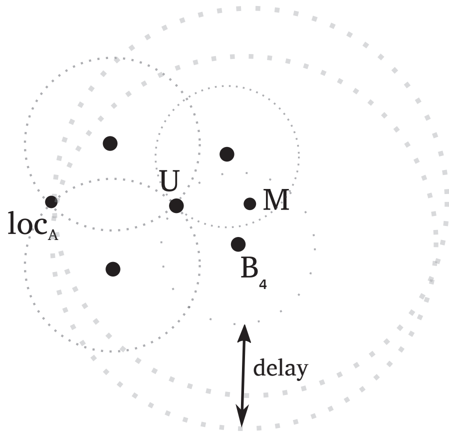

Figure 8.3 – Points without label are honest bases, B_4 is honest. B_4 is delayed.

on communication time in a DE protocol, delays cause incorrectness. For example, consider an adversary which actively involve in m-rounds challenge/response phase and delays the communication time. More specifically, in each round i, the adversary delays the exchange of challenge/response in total by $2\Delta_{\mathcal{A}}$ amount of time so that the response arrives V at $2d_P + 2\Delta_{\mathcal{A}}$ instead of $2d_p$. At the end, V outputs the distance as $d_P + \Delta_{\mathcal{A}}$ where there is no prover at this distance. If the adversary executes the delay attack to a DE protocol between the user and the base, then the pair (loc_{B_i}, d_i) will be corrupted as d_i is not correct.

Dragging Attack (See Figure 8.4):

Our adversarial model lets a malicious base run an arbitrary algorithm B^*. So, B^* may use an arbitrary location loc'_B as if loc'_B is its location instead of its real location loc_B (e.g., $d(loc_U, loc_B) < d(loc_U, loc'_B)$). If U obtains the correct distance $d_i = d(loc_U, loc'_B)$ with an incorrect location loc'_B, this effectively shortens the perceived distance to loc_{B_i}. We call this attack dragging attack, as it seems as if B_i moved away while dragging U behind. This is illustrated in Figure 8.4.

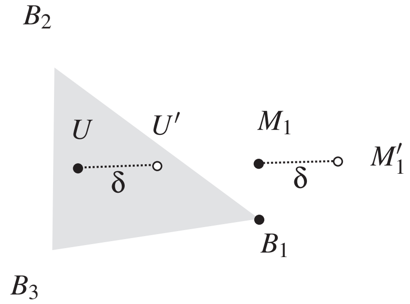

Figure 8.4 – Dragging Attack. M_1 convinces U that it is closer to loc_{M_1} than it really is by pretending to be at loc'_{M_1}.

Generic Construction: We propose a generic construction for an IL protocol in Figure 8.5. First, U generates a random nonce N and broadcasts it. After receiving N, each base B_i generates a signature S_{B_i} of the message (N, loc_{B_i}) with their secret key sk_{B_i} and sends the signature to U. U verifies the signature to be sure that the locations are sent by the bases. If all signatures are valid, then U starts a DE protocol with each base B_i sequentially in order to obtain its distance to each bases' locations. U runs the verifier algorithm of the DE protocol and B_i runs the prover algorithm of the DE protocol. At the end, U has all (loc_{B_i}, d_i) pairs and obtains its location by running an NIL algorithm (e.g., NIL-1, NIL-2, NIL-3).

$U(\mathsf{sk}_U, \mathsf{pk}_U, \{\mathsf{pk}_{B_i}\})$		$B_i(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i}, \mathsf{pk}_U, loc_{B_i})$
pick $N \in \mathbb{I}$	$\xRightarrow{\mathsf{N}}$	
if $\mathsf{Verify}_{\mathsf{pk}_{B_i}}(S_{B_i}, N, loc_{B_i}) \to \mathsf{False}$	$\xleftarrow{S_{B_i}, loc_{B_i}}$	$S_{B_i} = \mathsf{Sign}_{\mathsf{sk}_{B_i}}(N, loc_{B_i})$
cancel		
run $V(\mathsf{sk}_U, \mathsf{pk}_U) \to d_i$	$\xleftrightarrow{DE}$	**run** $P(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i}, \mathsf{pk}_U)$
run $\mathrm{NIL}(\{loc_{B_i}\}, \{d_i\}) \to loc_U$		
output loc_U		

Figure 8.5 – The generic construction of an IL protocol. Double arrow represents broadcasting (i.e., all bases receive N).

Theorem 8.16 (k'-security)**.** *Assume that number of k bases out of n bases are malicious and number of ℓ honest bases have their communications delayed (but not modified) by the adversary during the execution of IL, and $k' \geq k + \ell$. If the signature scheme is EF-CMA secure, the underlying DE protocol is mMiM-secure and NIL is k'-secure then IL in Figure 8.5 is k'-secure (as in Definition 8.5).*

Proof. Γ_0: Instances of bases and instances of the user play the game in Definition 8.5 with our assumptions.

Γ_1: We reduce Γ_0 to Γ_1 where (loc_{B_i}, N) pairs do not repeat. With u queries, the probability that (loc_{B_i}, N) repeats in Γ_0 is at most $\frac{u}{|\mathbb{I}|}$, which is negligible if $\mathbb{I}$ is large enough. Therefore, $|p_1 - p_0|$ is negligible.

Γ_2: We reduce Γ_1 to Γ_2 where we simulate honest bases' instances and the verification algorithm Verify with Verify' as follows:

$\underline{\boldsymbol{B_i}(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i}, \mathsf{pk}_{\boldsymbol{U}}, \boldsymbol{loc}_{B_i})}$	$\underline{\mathsf{Verify}'_{\mathsf{pk}_{B_i}}(\boldsymbol{S}_{B_i}, \boldsymbol{N}, \boldsymbol{loc}_{B_i})}$
receive N	**if** $(S_{B_i}, \mathsf{pk}_{B_i}, N, loc_{B_i})$ **in** Table
$S_{B_i} = \mathsf{sign}_{\mathsf{sk}_{B_i}}(loc_{B_i} N, \tau, r)$	**return** 1
store $(S_{B_i}, \mathsf{pk}_{B_i}, N, loc_{B_i})$ **in** Table	**else: return** 0
send S_C, loc_{B_i}	
run $P(\mathsf{sk}_{B_i, \mathsf{pk}_{B_i}, \mathsf{pk}_U})$	

The difference occurs between Γ_1 and Γ_2 when U receives a valid and forged signature. In this case, U in Γ_2 outputs $\perp$ while U in Γ_1 continues. Therefore, to prove that the difference between Γ_1 and Γ_2 is negligible, we assume the existence of an adversary $\mathcal{A}$ that makes U receive a forged signature in Γ_1 with probability p. We can then build $\mathcal{B}$ that simulates the k'-security game (Γ_1) to win the EF-CMA game for a given public key pk as follows. $\mathcal{B}$ first sets up keys for the honest bases and generates $n-k-1$ secret/public key pairs. It selects a base B_i among the honest bases at random and assigns pk as its public key. Then, it gives all public keys. It then simulates the k'-security game for $\mathcal{A}$. The simulation of U is as in IL protocol and the simulation of the bases is as follows: If $\mathcal{A}$ sends N_i to B_i, $\mathcal{B}$ sends (loc_{B_i}, N_i) to the EF-CMA signature oracle, and replies with the signature from the oracle. Otherwise, $\mathcal{B}$ simulates each honest $B_j \neq B_i$ as in IL protocol.

Whenever U receives a signature σ for a $loc'_{B_j} \neq loc_{B_j}$ that was not asked in previous queries, $\mathcal{B}$ uses it as a forgery attempt. The probability that the forged signature was produced using pk is $\frac{1}{n-k'}$ and $\mathcal{A}$ produces a valid signature with probability p. Thus, $\mathcal{B}$ wins the EF-CMA game with probability $\frac{p}{n-k'}$. As the signatures scheme is EF-CMA secure, $\frac{p}{n-k'}$ is negligible. So, $|p_1 - p_2|$ is negligible.

Remark that in Γ_2, the location of honest bases are always correct.

Γ_3: We reduce Γ_2 to Γ_3 where the algorithm V run by a user instance cancels if the estimated distance d_i of an honest base from U is such that $d_i < d(loc_{B_i}, loc_U)$. To prove that the difference between Γ_2 and Γ_3 is negligible, we assume the existence of an adversary $\mathcal{A}$ that makes U output a distance $d_i < d(loc_{B_i}, loc_U)$ with probability p' in Γ_2. Then, we can build $\mathcal{B}$, an mMiM adversary, with the advantage $\frac{p'}{n}$. Let V be the distinguished verifier instance for $\mathcal{B}$'s mMiM-security game. $\mathcal{B}$ simulates U using verifier instances and the bases using prover instances. $\mathcal{B}$ picks a base B_i and executes the DE protocol with V. The rest of the DE protocols are executed with other verifier instances. With probability $\frac{1}{n}$, the protocol executed with V is the one targeted by $\mathcal{A}$ and V outputs $d_i < d(loc_{B_i}, loc_V)$ with probability p'. Thus, $\mathcal{B}$ wins with probability $\frac{p'}{n}$. As the DE protocol is mMiM-secure, this probability and $|p_3 - p_4|$ are negligible.

In Γ_3, number of $n-k-\ell = n-k'$ honest bases's location/distance pair is correct and U obtains loc_U using the NIL algorithm. So, Γ_3's security is equivalent to k'-security of NIL algorithm. As we know that the NIL algorithm is k'-secure, Γ_3 is k'-secure as well.

□

The main problem in the IL protocol is that the number of honest bases whose communication is delayed is independent from the number of malicious bases. Even if there exists no malicious base, all honest bases' communication can be delayed. So, it is impossible to achieve secure IL in Theorem 8.16 if we do not limit the number of delays. Remark that in Theorem 8.16, we have the assumption that $k' \geq k+\ell$ to limit the number of delays. Therefore, we give another theorem in which we do not need to limit the number of delays. We start with some preliminary lemmas:

Lemma 8.17 ([ČH05a]). *For any two points loc_U and loc'_U ($loc_U \neq loc'_U$) located within a $Conv(loc_{B_i}, loc_{B_j}, loc_{B_k})$, at least one, but not more than two, of the following inequalities hold:*

$$d_i > d'_i; d_j > d'_j; d_k > d'_k$$

where d_i represents the distance between loc_U and loc_{B_i} and d'_i the distance between loc'_U and loc_{B_i}.

Lemma 8.18. *(Lemma 8.17's extension) For all $B_1,\dots,B_h$ and for all $loc_U, loc'_U \in Conv(B_1,\dots,B_h)$, there exists B_i such that $d(loc_{B_i}, loc'_U) < d(loc_{B_i}, loc_U)$.*

Proof. Let $B_1,\dots,B_h$ be arbitrary, $loc_U \in Conv(B_1,\dots,B_h)$ and $loc'_U \neq loc_U$ such that $\forall i, d(loc_{B_i}, loc'_U) \geq d(loc_{B_i}, loc_U)$. We will prove $loc'_U \notin Conv(B_1,\dots,B_h)$. Let Π be the half space of all P's such that $d(loc_P, loc'_U) \geq d(loc_P, loc_U)$. We have $\forall i, B_i \in \Pi$, so $Conv(B_1,\dots,B_h) \subseteq \Pi$. Since $loc'_U \notin \Pi$ we have $loc'_U \notin Conv(B_1,\dots,B_h)$. □

Let us call the IL protocol which uses NIL-2 as IL-2 protocol. In the following theorem, we prove that IL-2 is k-secure in a specific area.

Theorem 8.19 (k'-security of IL-2). *Let $\mathcal{H}$ be the set of honest bases. Let $\mathbb{O} = Conv(\mathcal{H})$ be the convex hull of honest bases.*

Assuming that a user located at $loc_U \in \mathbb{O}$, if the signature scheme is EF-CMA secure, the underlying DE protocol is mMiM-secure and NIL-2 is k'-secure then U of IL-2 outputs loc'_U in the security game in Definition 8.2 such that:

$$\Pr[loc'_U \in \mathbb{O} \wedge loc_U \neq loc'_U] < \delta$$

where δ is negligible.

Proof. Let $\ell = |\{B_i \in \mathcal{H} | d(loc_{B_i}, loc'_U) \geq d(loc_{B_i}, loc_U)\}|$.

Γ_0: The adversary plays a very similar game to the k-security game from Definition 8.5 with the following change: the adversary wins this game if U outputs $loc'_U \in \mathbb{O}$, $loc_U \neq loc'_U$.

We use the reductions from Theorem 8.16 to produce a game Γ_3 such that $|p_0 - p_3|$ is negligible.

In Γ_3, all d_i's are such that $d_i \geq d(loc_{B_i}, loc_U)$ and at most number of k' location/distance pairs are incorrect.

Now, we are in k'-security game for NIL-2. We know from Lemma 8.18, there exists B_j such that $d_j = d(loc_{B_j}, loc'_U) < d(loc_{B_j}, loc_U) \leq d_i$. NIL-2 outputs $loc'_U \in \mathbb{O}$ if and only if all spheres including $S(loc_{B_j}, d_j)$ has loc'_U. In Γ_3, all d_i's are such that $d_i \geq d(loc_{B_i}, loc_U)$. So, NIL-2 never outputs loc'_U.

□

8.4 Conclusion

In this chapter, we developed formal model for the security of localization. We first defined the security of a localization algorithm and a localization protocol by integrating the communication and adversarial model of distance bounding. Then, we analyzed the number of corruption on bases in order to guarantee outputting the correct location of the user. Thanks to this result, we constructed a secure localization algorithm NIL-1 which does not need to enumerate all t-tuples after checking the independence of locations. Then, we described another algorithm NIL-2 which works securely with more number of incorrect locations. We also constructed NIL-3 which is more efficient comparing to NIL-1 and NIL-2 with an extra assumption of having at at least $t+1$-independent correct location/distance pairs. However, NIL-2 and NIL-3 are not full-secure as they directly aborts if they find some inconsistency. On the other hand, NIL-1 is secure with less numbers of incorrect locations but it is also full secure. Then, we constructed an interactive localization protocol between bases and a user. This protocol consists of a secure distance estimate protocol, a secure signature scheme and a secure localization protocol. We first analyzed its security in case of limited delay attack executed by adversaries. Second, we showed that our protocol is secure in certain areas without limitation on delay attacks.

Chapter 9

Proof of Location

Chandran et al. [CGMO09] bring a novel approach to cryptography which is using the position of a person as a credential because the location of a person may also define the identity of this person (e.g., we trust a bank teller behind the window without checking her identity because of her location). One of the fundamentals of position-based cryptography is secure positioning, where a party convinces multiple verifiers that (s)he is at a certain location. Unfortunately, it is not possible to achieve secure positioning in Vanilla model as shown in [CGMO09]. Because of this, we propose a different model. We build our new model that we call proof of location (PoL). PoL is constructed on top of secure hardware model (SHM) for distance bounding as described in Chapter 5, and our localization model given in Chapter 8. In this integrated model, we assume that the secure hardware is the part of the prover but it is always honest while the prover can be malicious. We do not have any key set up for the prover. It only authenticates himself with its location by using his hardware. Our model can fit in real life situations. For instance, consider that only people in an office can access printers and the prover who wants to print has to show that he is at the office. In this case, the prover using his hardware can prove his position. Consider a pizza company which has a delivery service. This pizza company can produce its hardware to be distributed to people who use the delivery service. Later on, whenever a person orders a pizza, this person can also prove his location to be verified with a given address. Thus, the company can prevent fraudulent people who order pizza just for the denial of service.

Related Works Secure positioning is a well-studied problem in wireless security. Sastry et al. [SSW03] give a secure positioning protocol which uses an echo distance bounding. The verifiers aim to understand if the prover is in a claimed area. They assume that the verifiers are always honest but this assumption weakens the security model comparing to other works. There are other secure positioning protocols [ZLFW06, SP05] in this weaker model as well.

Čapkun and Hubaux [ČH05b] introduce a mechanism called Verifiable Multilateration (VM) consisting of at least three verifiers (in two-dimensional space) which have constant location, an authority and a prover. The verifiers determine the proximity of the prover by using a distance bounding protocol. Then, the authority runs a test protocol between them to validate the distances learned. If any delay attack is detected, the validation fails. The validation detects delays due to the fact that a prover inside the triangle which is consisted of three verifiers cannot prove different position without delaying. The adversarial model of VM is consisting of an external attacker and compromised nodes. The main drawback of VM is informal security analysis. For example, their security analyzes are based on that the attacker cannot shorten its distance but the DB protocol used in VM actually lets a compromised node shorten its distance with distance hijacking attacks. Perazzo and Dini [PD15] discuss about the negative effect of non-ideal distance bounding against VM as implementing an ideal DB is hard.

Čapkun et al. [ČČS06] also introduced a model in which part of the infrastructure bases are either hidden from users or mobile. This is meant to avoid the generic attack by preventing attackers to position properly for the attack to work. The model is relatively simple but Chandran et al. [CGMO09] show that hidden bases' positions can be discovered if a user is allowed multiple executions of the protocol and gets feedback on its position claim was accepted or not (both reasonable assumptions). Another attack with constant probability of success is described to defeat the protocol based on mobile stations.

Delaët et al. [DMRT11] consider stronger adversarial model than previous works, where each node in a wireless sensor network does not have any information about the other nodes. This means that a node can cheat on its location. They give an algorithm which detects faking nodes and analyze their algorithm in which cases the algorithm works correctly. In the similar adversarial model, Hwang et al. [HHK07] propose a faking node detection algorithm which works probabilistically.

Secure positioning is a problem of vehicular ad hoc network (VANET). Song et al. [SWL08] propose a method to detect and prevent spoofing attack by using an honest neighbor node. So, the adversarial model is rather limited.

Chandran et al. [CGMO09] brings a novel cryptographic approach to the secure positioning and introduce position-based cryptography. They also propose to use the position for cryptographic protocols such as a secure key exchange. They prove that secure positioning is impossible in the Vanilla model. Therefore, they provide a new model for secure positioning with a bounded storage. However, this model is limited because its security is based on the adversary has a limited storage. There are also quantum approaches [CFG$^+$10, BK11, BCF$^+$14, LL11] for position-based cryptography.

Lastly, Akand and Safavi-Naini [ASN18] introduced region authentication. Here, a prover proves that it is in a region instead of in a specific location. In their setup, prover has a secret key and identified with this so it cannot apply to position-based cryptography.

9.1 Our Contribution

- We define the problem of secure positioning in a different way in order to obtain an achievable security. We consider a model which integrates localizations and tamper-proof devices for proof of location. Our model for the proof of location is based on the secure hardware model [KV18a] and localization model.
- In our model, the prover does not have any credential than its location as suggested by Chandran et al. [CGMO09].
- We propose a protocol which is constructed on top of a weak variant of distance bounding and localization algorithm. We formally prove its security according to our security model.

Structure of the Chapter: In Section 9.2, we give the definition of proof of knowledge (PoL) and its security model. Then, we propose a PoL protocol and prove its security in Section 9.3. We conclude this chapter with Section 9.4.

9.2 Definitions

We first give the definition of a proof of location and then show the security model.

Definition 9.1 (Proof of Location (PoL)). *A proof of location protocol PoL consists of a tuple* $(\mathcal{K}_H, V_l, P_l, H_l)$ *and an IL tuple* $(\mathcal{K}_B, \mathcal{K}_U, B, U, n, \mathbb{M})$ *as defined in Definition 8.2.* $\mathcal{K}_{\mathcal{H}}$ *is the key generation algorithm outputting* $(\mathsf{sk}_{H_l}, \mathsf{pk}_{H_l})$*.* V_l *is the algorithm of a location verifier with the input* pk_{H_l}*,* P_l *is the algorithm of a location prover with no input and* H_l *is the algorithm of the prover's hardware with the input* $(\mathsf{sk}_{H_l}, \mathsf{pk}_{H_l}, \{\mathsf{pk}_{B_i}\})$*. At the end of the interactions between these algorithms,* V_l *outputs a location* $loc \in \mathbb{M}$ *or* $\perp$*.*

Correctness: A proof of location protocol with secure hardware is correct for all loc_{B_i}'s, secret/public key pairs, n and $\mathbb{M}$ if V_l, P_l, H_l and B run correctly, V_l always outputs the location of P_l.

Remark that in our model for PoL, we do not require a key setup for the prover. The verifier identifies the prover according to his location and the key of the hardware is not related with the prover's credential.

Adversarial and Communication Model: Our adversarial and communication model for PoL is integrated from localization model in Section 8.3.1 and the secure hardware model (SHM) for DB in Section 5.2. In PoL model, we add the following assumptions related to the SHM in addition to those from the localization model explained in the previous sections.

- Hardware are always honest.

- Provers can be corrupted by an adversary and corrupted provers can run an arbitrary algorithm P_l^*.
- Each prover possesses a secure hardware.

We note that we do not have the assumption '*the secure hardware of an honest prover can only communicate with its prover and they are both at the same location*' of the SHM in our PoL model. We do not want a location prover to have any other credential or identity other than its location. If we had this assumption, then the hardware would be the part of the location prover's credential, which is equivalent to having a key setup for the location prover. So, our model can be imagined as there are some hardware which help any party to prove its location as verifiers in secure positioning.

Definition 9.2 ((k,ε)-PoL Security)**.** *The game is played with an IL protocol. It begins with the same game set up as in Definition 8.2. It also runs the key generation algorithm $\mathcal{K}_H$. The adversary can create polynomially many instances of the location prover P_l, the hardware H_l and the location verifier V_l. If one of the location-verifier instances outputs $loc \in \mathbb{M}$ message while $d(loc, loc_P) > \varepsilon$ for all locations of P_l's instances loc_P, then the adversary wins the game. We say PoL is secure if for any such game the probability of an adversary to win is negligible.*

We now give a different DB definition than the one we see until now. We remove the key setup of DB and obtain Keyless DB. This type of DB lets us construct PoL without any key setup on the location-prover side.

Definition 9.3 (Keyless DB)**.** *A keyless distance bounding protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple (V, P, ε). P is the proving algorithm with no input, $V(1^\ell)$ is the verifying algorithm where ℓ is the security parameter. At the end of the protocol, $V(1^\ell)$ outputs a final message $Out_V \in \{0,1\}$.*

A Kless DB protocol is correct if and only if under honest execution, whenever a verifier V and a prover P lie at most a distance ε from each other, V always outputs $Out_V = 1$.

Definition 9.4 (Security of Keyless DB)**.** *In keyless DB game, the adversary can run multiple instances of V and P. If one of the instances of V outputs 1 while there exists no instance close to V, then the adversary wins. A keyless DB is secure, if the success probability of an adversary in this game is negligible.*

We give an example of a keyless DB protocol in Figure 9.1. In the protocol, V first picks randomly a bit string C whose each bit corresponds to challenges. Then, in the challenge phase, V sends the challenge c_i (i^{th} bit of C). P responds with the same challenge. At the end, V checks if all responses r_i are equal to c_i and if all arrived on time.

Theorem 9.5. *Echo is a secure keyless DB.*

Proof. Lemma 3.4 implies that the echo protocol is secure. So, the success probability of the adversary is at most $\frac{1}{2^\ell}$. □

$$
\begin{array}{lcr}
\underline{V}(1^\ell) & & \underline{P} \\
\textbf{pick } C \in \{0,1\}^\ell & & \\
 & \textbf{challenge phase} & \\
 & \text{for } i = 1 \text{ to } \ell & \\
c_i = C[i], \text{ start } \mathsf{timer}_i & \xrightarrow{\quad c_i \quad} & \textbf{receive } c_i \\
\text{stop } \mathsf{timer}_i & \xleftarrow{\quad r_i \quad} & r_i = c_i \\
 & \textbf{verification phase} & \\
\text{check } \mathsf{timer}_i \leq 2\varepsilon, r_i = C[i] & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure 9.1 – Echo Protocol for Keyless DB

9.3 Proof of Location Protocol

We can achieve PoL with secure hardware by the following protocol PoL^H (See Figure 9.2). It is straightforward due to our hardware assumption. First, the hardware checks if a party is around it, learns its own location and executes the proof by sending the signature of its location. The details are below.

$$
\begin{array}{lcccr}
\underline{H_l(\mathsf{sk}_{H_l}, \mathsf{pk}_{H_l}, \{\mathsf{pk}_{B_i}\})} & & \underline{P_l} & & \underline{V_l(\mathsf{pk}_{H_l})} \\
 & & \underline{\textbf{Distance Bounding}} & & \\
 & & & & N \leftarrow \mathcal{N} \\
 & \xleftarrow{\quad N \quad} & & \xleftarrow{\quad N \quad} & \textbf{start } \mathsf{timer} \\
\textbf{run } V(1^\ell) \to \mathsf{Out}_V & \xleftrightarrow{\mathsf{KlessDB}} & \textbf{run } P & & \\
\text{if } \mathsf{Out}_V = 0\text{:} & & & & \\
\qquad \text{cancel} & & \underline{\textbf{Localization}} & & \\
 & & & & \text{with all } \underline{B_i} \\
\textbf{run } U(\mathsf{sk}_U, \mathsf{pk}_U\{\mathsf{pk}_{B_i}\}) \to loc & & \xleftrightarrow{\qquad IL \qquad} & & \textbf{run } B(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i}, \mathsf{pk}_U, loc_{B_i}) \\
 & & \underline{\textbf{Proof}} & & \\
 & & & & \underline{V_l(\mathsf{pk}_{H_l})} \\
\sigma = sign_{sk_S}(loc_H || N) & \xrightarrow{\quad \sigma, loc \quad} & & \xrightarrow{\quad \sigma, loc \quad} & \textbf{stop } \mathsf{timer} \\
 & & & & \textbf{if } \mathsf{Verify}_{\mathsf{pk}_S}(\sigma, N, loc) \to 0 \\
 & & & & \textbf{or } \mathsf{timer} > T\text{:} \\
 & & & & \text{output } \perp \\
 & & & & \textbf{else:} \\
 & & & & \text{output } loc
\end{array}
$$

Figure 9.2 – PoL^H

PoL^H: In the key setup of PoL^H, $\mathcal{K}_H$ first generates a secret/public key pair $(\mathsf{sk}_U, \mathsf{pk}_U)$ by running $\mathcal{K}_U$ of an IL protocol and generates another pair $(\mathsf{sk}_S, \mathsf{pk}_S)$ by using the key generation algorithm of a signature scheme. At the end, it outputs $(\mathsf{sk}_H, \mathsf{pk}_H) = ((\mathsf{sk}_U, \mathsf{sk}_S), (\mathsf{pk}_U, \mathsf{pk}_S))$. Here, we assume that there exits n bases which have secret/public key pairs $\{(\mathsf{sk}_{B_i}, \mathsf{pk}_{B_i})\}$ generated by $\mathcal{K}_B$ of an IL algorithm.

In the first stage of the protocol, V picks a nonce N and sends it to the location prover P_l. It also starts the timer. Then, P_l sends the nonce N to H. After receiving N, the hardware and the location prover start to run a keyless DB protocol so that the hardware makes sure that there exists a party in ε-neighborhood of it. Here, the hardware runs

$V(1^\ell)$ of $\mathsf{KlessDB}$ and the location prover runs P. If the location prover proves that it is in ε-neighborhood, then the hardware starts to learn its own location. To do so, it starts an IL protocol with n bases. During localization, it runs $U(\mathsf{sk}_U, \mathsf{pk}_U, \{\mathsf{pk}_{B_i}\})$ and obtains a location $loc \in \mathbb{M}$. Then, the proving phase begins. The hardware generates the signature of the message $N||loc$ by sk_S. Then, it sends loc and the signature σ to the verifier via the location prover. The verifier stops the timer and verifies both the nonce and the signature. If the signature passes the verification and if timer is not more than expected ($\mathsf{timer} < T$)[1], the verifier outputs loc. Otherwise, it outputs $\perp$.

Theorem 9.6. *If the signature scheme is EF-CMA secure and IL protocol is k-secure and* $\mathsf{KlessDB}$ *is secure (Definition 9.4), then* PoL^H *in Figure 9.2 is secure with the presence of at most k malicious bases.*

Proof. Γ_0: This is the PoL's security game with the protocol PoL^H.

Γ_1: We reduce Γ_0 to Γ_1 where (loc_U, N) pairs do not repeat. With r queries, the probability that (loc, N) repeats in Γ_0 is at most $\frac{r}{|\mathcal{N}|}$, which is negligible if $\mathcal{N}$ is large enough. Therefore, $|p_0 - p_1|$ is negligible.

Γ_2: We reduce Γ_1 to Γ_2 where the location verifier always rejects if the signature received is not generated by H_l. We can easily show that the case which differs Γ_2 and Γ_3 happens with negligible probability by using the EF-CMA security of the signature scheme. So, $|p_2 - p_1|$ is negligible.

Γ_3 : We reduce Γ_2 to Γ_3 where the location that H_l learns cannot be wrong. We can show that if H obtains a wrong location, we can construct an adversary which breaks the security of IL by simulating the bases in IL. Therefore, $|p_3 - p_2|$ is negligible.

In Γ_3, locations received by all verifier instances are correct.

Γ_4 : We reduce Γ_3 to Γ_4 where we simulate the hardware instance by canceling the protocol if there exists no close party which has a distance at most ε.

We show that if there exists an instance P_l which is further than ε and its matching instance H_l do not cancel, then we can construct a keyless DB adversary $\mathcal{A}$. $\mathcal{A}$ simulates the hardware against P_l by V in keyless DB security game. $\mathcal{A}$ behaves same P_l against keyless DB security game. If P_l succeeds, then $\mathcal{A}$ succeeds. Therefore, $|p_4 - p_3|$ is negligible.

So, in Γ_4, the hardware continues if there exists a party in ε-neighborhood of loc which is the location of the hardware. In this case, the adversary cannot win Γ_4. □

9.4 Conclusion

In this chapter, we described a security model for the proof of location problem. In our model, we considered the existence of honest hardware which are possessed by provers who want to prove their location. There exists no key setup for the prover. The PoL

[1] T can be considered as the maximum time given to H in order to execute the Keyless DB and localization protocols.

model is the combination of localization and secure hardware DB models. We constructed a protocol where the hardware obtains the location of the prover and prove this location to a verifier. We proved that this protocol is secure in our model. Our PoL model is achievable and so far the only alternative to the bounded memory assumption which is used in [CGMO09].

Chapter 10

Conclusion and Future Work

In this thesis, we focused on the role of position in cryptography. So, we solved some existing problems related to distance bounding, we provided security models for specific applications of proof of proximity with distance bounding, and we constructed localization algorithms and secure positioning protocols.

First, we concentrated on the theory of distance bounding. We considered different structures for DB and analyzed the optimal security bounds for it. In one of the structures called sync structure, we included the prover in time computation of the challenge phase. We obtained better optimal security bounds for MiM-security with the prover's involvement. In the other structure, we randomized the time of sending challenges by the verifier. We added this change to the verifier in the common structure and the sync structure and obtained better security bounds for MiM and DF security. Then, we constructed the most efficient public-key distance bounding protocol in its security level. Our efficient protocol Eff-pkDB and its private variant Eff-pkDB^p is a generic construction which consists of a D-AKA secure key agreement protocol and a one-time secure symmetric DB protocol. We also provided a variant of it which requires less security from the symmetric DB. In the end, we compared the efficiency and the security of our protocols and we saw that our protocols are better in terms of efficiency and security. Moreover, we constructed a model called the secure hardware model by considering the problem of defining the terrorist-fraud security. With this model, we showed that the terrorist-fraud security is possible. We also showed some relations between SHM and plain model.

Second, we integrated distance bounding with contactless access control and contactless payment. These applications need a proof of proximity for their security. So, our integration provided full security. For contactless access control, we constructed a security and a privacy model. In the AC-security model, we used the same adversarial and communication model as distance bounding and provided a security definition which covers MiM-attacks, DH-attacks, and impersonation attacks. In the privacy model, we considered the same adversarial and communication model and we provided a privacy definition which takes into account timing as the AC security model. We showed how to convert a distance bounding protocol into an AC protocol securely. However, this conversion does not preserve privacy so we suggested analyzing privacy before using the conversion. For contactless payment, we provided a security model considering the malicious terminal and malicious card. We constructed a new secure contacless payment protocol ClessPay. Besides, we proved the security of the contactless EMV-protocol against malicious terminals in our model. We show some vulnerabilities of contactless EMV against malicious cards.

Third, we focused on localization and secure positioning. We considered the security of localization algorithms and the security of interactive-localization protocols. We designed three secure localization algorithms. We constructed a generic protocol for interactive localization which consists of a distance estimate protocol (equivalent to distance bounding), a signature scheme and a localization algorithm. Then, we considered secure positioning. Instead of using the structure of the existing secure positioning protocols, we defined the new one and called proof of location. The security model of proof of location consists of the combination of the secure hardware model and the localization model. In the end, we constructed a proof of location protocol and proved its security.

Future Work: Privacy in symmetric DB has not been considered formally. It can be an interesting issue to work because the prover who runs a symmetric DB may need privacy as much as the prover who runs a public-key DB. Beyond the theory of DB, there are still issues for implementing distance bounding even though we have some developments on it [CHKM06, SLČ17, HK05, RČ10, HK08]. This is an important problem to solve.

In IL, we consider a perfect environment in which the user always obtains the correct distance from distance estimate protocols in an honest environment. However, in real life, this is not always the case. The messages may arrive later than they are supposed to arrive owing to some communication traffic. In such environments, our protocols do not work because they always output $\perp$. As a future direction, the IL security can be defined considering an imperfect environment and an error margin can be suggested considering the security of IL protocols.

Another future work can be about how to adapt PoL in position-based cryptography introduced by Chandran et al. [CGMO09]. For example, PoL can be considered for the position-based secure communication or the position-based decryption.

List of Figures

List of Definitions

List of Tables

List of Algorithms

Bibliography

[ABG+17] Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 800–814. ACM, 2017. *Cited on pages:* xv, 7, 37, 38, 58, 61, 148, 150, 176, 177, 178, and 179.

[ABK+09] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardas, Cédric Lauradoux, and Benjamin Martin. A formal framework for cryptanalyzing RFID distance bounding protocols. *IACR Cryptology ePrint Archive*, 2009:543, 2009. *Cited on pages:* 9, 62, 63, and 75.

[ABK+11] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardaş, Cédric Lauradoux, and Benjamin Martin. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security*, 19(2):289–317, 2011. *Cited on pages:* 9, 62, 63, 75, and 119.

[All03] Smart Card Alliance. Using smart cards for secure physical access. *Smart Card Alliance Report*, 54, 2003. *Cited on pages:* 79 and 80.

[All13] Smart Card Alliance. Industry technical contributions: Opacity, 2013. *Cited on pages:* 79, 80, and 84.

[ASN17] Ahmad Ahmadi and Reihaneh Safavi-Naini. Directional distance-bounding identification. In *International Conference on Information Systems Security and Privacy*, pages 197–221. Springer, 2017. *Cited on page:* 62.

[ASN18] Mamunur Rashid Akand and Reihaneh Safavi-Naini. In-region authentication. In *International Conference on Applied Cryptography and Network Security*, pages 557–578. Springer, 2018. *Cited on page:* 138.

[BB05] Laurent Bussard and Walid Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Security and Privacy in the Age of Ubiquitous*

Computing, IFIP Advances in Information and Communication Technology Volume 181, pages 223–238. Springer, 2005. *Cited on pages:* 37 and 61.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, pages 566–582. Springer, 2001. *Cited on pages:* 16, 56, 73, and 149.

[BC93] Stefan Brands and David Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, LNCS 765, pages 344–359. Springer-Verlag, 1993. *Cited on pages:* xv, 2, 7, 37, 38, 58, 61, 62, 64, 75, 118, 122, 148, 167, and 168.

[BCF+14] Harry Buhrman, Nishanth Chandran, Serge Fehr, Ran Gelles, Vipul Goyal, Rafail Ostrovsky, and Christian Schaffner. Position-based quantum cryptography: Impossibility and constructions. *SIAM Journal on Computing*, 43(1):150–178, 2014. *Cited on page:* 138.

[BCM+14] Mike Bond, Omar Choudary, Steven J Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and skim: cloning EMV cards with the pre-play attack. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 49–64. IEEE, 2014. *Cited on page:* 95.

[BCM+15] Mike Bond, Marios O Choudary, Steven J Murdoch, Sergei Skorobogatov, and Ross Anderson. Be prepared: The EMV preplay attack. *IEEE Security & Privacy*, 13(2):56–64, 2015. *Cited on page:* 95.

[BD91] Thomas Beth and Yvo Desmedt. *Identification tokens-or: Solving the chess grandmaster problem.* LNCS 537. Springer, 1991. *Cited on page:* 2.

[BG07] Daniel RL Brown and Kristian Gjøsteen. A security analysis of the NIST SP 800-90 elliptic curve random number generator. In *CRYPTO 2007*, pages 466–481. Springer, 2007. *Cited on page:* 170.

[BGG+16] Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 121–133. ACM, 2016. *Cited on page:* 61.

[BK11] Salman Beigi and Robert König. Simplified instantaneous non-local quantum computation with applications to position-based cryptography. *New Journal of Physics*, 13(9):093036, 2011. *Cited on page:* 138.

[BMV13a] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Secure and lightweight distance-bounding. In *Lightweight Cryptography for Security and Privacy*, LNCS 8162, pages 97–113. Springer, 2013. *Cited on pages:* 7, 9, 39, 61, 62, 75, 119, and 122.

[BMV13b] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Towards secure distance bounding. In *Fast Software Encryption*, LNCS 8424, pages 55–67. Springer, 2013. *Cited on pages:* 7, 9, 15, 39, 61, and 119.

[BMV15] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. Practical and provably secure distance-bounding. *Journal of Computer Security*, 23(2):229–257, 2015. *Cited on pages:* 7, 9, 15, 39, 61, and 119.

[BNPS03] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-RSA-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16:185–215, 2003. *Cited on page:* 170.

[BR93] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Annual International Cryptology Conference*, LNCS 773, pages 232–249. Springer, 1993. *Cited on page:* 79.

[BR04] Laurent Bussard and Yves Roudier. Embedding distance-bounding protocols within intuitive interactions. In *Security in Pervasive Computing*, pages 143–156. Springer, 2004. *Cited on pages:* 62 and 72.

[BV14] Ioana Boureanu and Serge Vaudenay. Optimal proximity proofs. In *Inscrypt*, LNCS 8957, pages 170–190. Springer, 2014. *Cited on pages:* 7, 8, 9, 11, 15, 21, 22, 23, 24, 29, 30, 31, 32, 33, 34, 35, 38, 61, 62, 69, 71, 75, 119, 149, and 173.

[ČBH03] Srdjan Čapkun, Levente Buttyan, and Jean-Pierre Hubaux. SECTOR: secure tracking of node encounters in multi-hop wireless networks. In *In ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 21–32, 2003. *Cited on page:* 61.

[ČČS06] Srdjan Čapkun, Mario Čagalj, and Mani Srivastava. Secure localization with hidden and mobile base stations. In *in Proceedings of IEEE INFOCOM*. Citeseer, 2006. *Cited on pages:* 118, 119, and 138.

[CFG+10] Nishanth Chandran, Serge Fehr, Ran Gelles, Vipul Goyal, and Rafail Ostrovsky. Position-based quantum cryptography. *arXiv preprint arXiv:1005.1750*, 2010. *Cited on page:* 138.

[CGDR+15] Tom Chothia, Flavio D Garcia, Joeri De Ruiter, Jordi Van Den Breekel, and Matthew Thompson. Relay cost bounding for contactless EMV payments. In *International Conference on Financial Cryptography and Data Security*, pages 189–206. Springer, 2015. *Cited on pages:* 38, 95, 96, and 113.

[CGMO09] Nishanth Chandran, Vipul Goyal, Ryan Moriarty, and Rafail Ostrovsky. Position based cryptography. In *Advances in Cryptology-CRYPTO 2009*, pages 391–407. Springer, 2009. *Cited on pages:* 118, 137, 138, 139, 143, 146, and 171.

[ČH05a] Srdjan Čapkun and J-P Hubaux. Secure positioning of wireless devices with application to sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1917–1928. IEEE, 2005. *Cited on pages:* 117, 118, 119, and 134.

[ČH05b] Srdjan Čapkun and J-P Hubaux. Secure positioning of wireless devices with application to sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1917–1928. IEEE, 2005. *Cited on page:* 137.

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952. *Cited on page:* 9.

[CHKM06] Jolyon Clulow, Gerhard Hancke, Markus Kuhn, and Tyler Moore. So near and yet so far: Distance-bounding attacks in wireless networks. *Security and Privacy in Ad-hoc and Sensor Networks*, pages 83–97, 2006. *Cited on pages:* 9, 96, 113, and 146.

[CRSČ12] Cas Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Čapkun. Distance hijacking attacks on distance bounding protocols. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 113–127. IEEE, 2012. *Cited on pages:* 1 and 167.

[Des88] Yvo Desmedt. Major security problems with the “unforgeable” (Feige-)Fiat-Shamir proofs of identity and how to overcome them. In *Congress on Computer and Communication Security and Protection Securicom*, pages 147–159. SEDEP Paris France, 1988. *Cited on pages:* 1 and 2.

[DFF+14] Jean Paul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, Giorgia Azzurra Marson, Arno Mittelbach, and Kenneth G Paterson. Unpicking PLAID. In *International Conference on Research in Security Standardisation*, LNCS 8893, pages 1–25. Springer, 2014. *Cited on pages:* 79 and 80.

[DFG+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of opacity. In *European Symposium on Research in Computer Security*, LNCS 8134, pages 345–362. Springer, 2013. *Cited on pages:* 79, 80, and 84.

[DFKO11] Ulrich Dürholz, Marc Fischlin, Michael Kasper, and Cristina Onete. A formal approach to distance-bounding RFID protocols. In *Information Security*, LNCS 7001, pages 47–62. Springer, 2011. *Cited on pages:* 9, 38, 62, 66, 75, 119, 170, and 176.

[DH76] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976. *Cited on page:* 42.

[DM+07] Saar Drimer, Steven J Murdoch, et al. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX security symposium*, volume 312, 2007. *Cited on page:* 95.

[DMRT11] Sylvie Delaët, Partha Sarathi Mandal, Mariusz A Rokicki, and Sébastien Tixeuil. Deterministic secure positioning in wireless sensor networks. *Theoretical Computer Science*, 412(35):4471–4481, 2011. *Cited on page:* 138.

[emva] EMV acquirer and terminal security guidelines. *Cited on pages:* 97, 113, and 148.

[emvb] EMV contactless specifications for payment systems, Book C-2: Kernel 2 specification. *Cited on pages:* 95, 110, and 186.

[emvc] EMV integrated circuit card specifications for payment systems, Book 2: Security and key management. *Cited on pages:* 110, 112, 186, and 187.

[emvd] EMVCo: EMV contactless specifications for payment systems, version 2.4 (2014). *Cited on page:* 114.

[FDČ11] Aurélien Francillon, Boris Danev, and Srdjan Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011. *Cited on pages:* 79, 95, and 96.

[FHMM10] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, LNCS 6370, pages 35–49. Springer, 2010. *Cited on pages:* 79 and 95.

[FO13] Marc Fischlin and Cristina Onete. Terrorism in distance bounding: modeling terrorist-fraud resistance. In *Applied Cryptography and Network Security*, LNCS 7954, pages 414–431. Springer, 2013. *Cited on pages:* 7, 61, and 62.

[gDoHSD10] Centrelink: Australian government's Department of Human Services (DHS). Protocol for lightweight authentication of identity (PLAID), 2010. *Cited on pages:* 79 and 80.

[GOR14a] Sébastien Gambs, Cristina Onete, and Jean-Marc Robert. Prover anonymous and deniable distance-bounding authentication. In *ASIA CCS*, Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pages 501–506, 2014. *Cited on pages:* xv, 7, 37, 38, 148, 172, and 173.

[GOR14b] Sebastien Gambs, Cristina Onete, and Jean-Marc Robert. Prover anonymous and deniable distance-bounding authentication. Cryptology ePrint Archive, Report 2014/114, 2014. `http://eprint.iacr.org/`. *Cited on page:* 173.

[Han05] Gerhard P Hancke. A practical relay attack on ISO 14443 proximity cards. *Technical report, University of Cambridge Computer Laboratory*, 59:382–385, 2005. *Cited on pages:* 61 and 79.

[Han06] Gerhard P Hancke. Practical attacks on proximity identification systems. In *Security and Privacy, 2006 IEEE Symposium on*, pages 6–pp. IEEE, 2006. *Cited on page:* 79.

[HHK07] Joengmin Hwang, Tian He, and Yongdae Kim. Detecting phantom nodes in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2391–2395. IEEE, 2007. *Cited on page:* 138.

[HK05] Gerhard P Hancke and Markus G Kuhn. An RFID distance bounding protocol. In *SecureComm 2005*, pages 67–73. IEEE, 2005. *Cited on pages:* 9, 13, and 146.

[HK08] Gerhard P Hancke and Markus G Kuhn. Attacks on time-of-flight distance bounding channels. In *Proceedings of the first ACM conference on Wireless network security*, pages 194–202. ACM, 2008. *Cited on page:* 146.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963. *Cited on page:* 9.

[HPO13] Jens Hermans, Roel Peeters, and Cristina Onete. Efficient, secure, private distance bounding without key updates. In *WiSec*, Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, pages 207–218, 2013. *Cited on pages:* xv, 7, 37, 38, 58, 75, 148, 169, 170, and 171.

[HPVP11] Jens Hermans, Andreas Pashalidis, Frederik Vercauteren, and Bart Preneel. A new RFID privacy model. In *ESORICS*, LNCS 6879, pages 568–587. Springer, 2011. *Cited on pages:* 14, 69, 85, and 149.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001. *Cited on page:* 58.

[KAK+08] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife RFID distance bounding protocol. In *Information Security and Cryptology–ICISC 2008*, pages 98–115. Springer, 2008. *Cited on pages:* 9, 62, and 66.

[KAK+09] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. The swiss-knife RFID distance bounding protocol. In *ICISC*, pages 98–115. Springer, 2009. *Cited on page:* 9.

[Kra05] Hugo Krawczyk. Hmqv: A high-performance secure Diffie-Hellman protocol. In *CRYPTO*, pages 546–566. Springer, 2005. *Cited on page:* 181.

[Kuh04] Markus G Kuhn. An asymmetric security mechanism for navigation signals. In *International Workshop on Information Hiding*, pages 239–252. Springer, 2004. *Cited on page:* 117.

[KV15] Handan Kılınç and Serge Vaudenay. Optimal proximity proofs revisited. In *ACNS*, LNCS 9092, pages 478–494. Springer, 2015. *Cited on page:* 21.

[KV16] Handan Kılınç and Serge Vaudenay. Efficient public-key distance bounding protocol. In *ASIACRYPT*, LNCS 10032, pages 873–901, 2016. *Cited on pages:* 7, 37, 38, 61, 75, 81, 88, 89, 91, 92, 93, 108, and 184.

[KV17] Handan Kılınç and Serge Vaudenay. Contactless access control based on distance bounding. In *International Conference on Information Security*, LNCS 10599, pages 195–213. Springer, 2017. *Cited on pages:* 79 and 99.

[KV18a] Handan Kılınç and Serge Vaudenay. Formal analysis of distance bounding with secure hardware. In *International Conference on Applied Cryptography and Network Security*, LNCS 10892, pages 579–597. Springer, 2018. *Cited on pages:* 61 and 139.

[KV18b] Handan Kılınç and Serge Vaudenay. Secure contactless payment. In *Australasian Conference on Information Security and Privacy*, LNCS 10946, pages 579–597. Springer, 2018. *Cited on page:* 95.

[LL11] Hoi-Kwan Lau and Hoi-Kwong Lo. Insecurity of position-based quantum-cryptography protocols against entanglement attacks. *Physical review a*, 83(1):012322, 2011. *Cited on page:* 138.

[LLM07] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *Provable Security*, pages 1–16. Springer, 2007. *Cited on pages:* 39, 40, 57, 181, and 182.

[LM06] Kristin Lauter and Anton Mityagin. Security analysis of KEA authenticated key exchange protocol. In *Public Key Cryptography-PKC 2006*, pages 378–394. Springer, 2006. *Cited on page:* 181.

[LMQ+03] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003. *Cited on page:* 181.

[LND05a] Donggang Liu, Peng Ning, and Wenliang Du. Detecting malicious beacon nodes for secure location discovery in wireless sensor networks. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 609–619. IEEE, 2005. *Cited on page:* 118.

[LND05b] Donggang Liu, Peng Ning, and Wenliang Kevin Du. Attack-resistant location estimation in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 13. IEEE Press, 2005. *Cited on pages:* 117 and 118.

[LP04] Loukas Lazos and Radha Poovendran. Serloc: Secure range-independent localization for wireless sensor networks. In *Proceedings of the 3rd ACM workshop on Wireless security*, pages 21–30. ACM, 2004. *Cited on pages:* 117 and 118.

[LP06] Loukas Lazos and Radha Poovendran. HiRLoc: High-resolution robust localization for wireless sensor networks. *IEEE Journal on selected areas in communications*, 24(2):233–246, 2006. *Cited on pages:* 117 and 118.

[LPČ05] Loukas Lazos, Radha Poovendran, and Srdjan Čapkun. ROPE: Robust position estimation in wireless sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 43. IEEE Press, 2005. *Cited on pages:* 117, 118, and 119.

[ME06] Pratap Misra and Per Enge. Global positioning system: Signals, measurements and performance second edition. *Massachusetts: Ganga-Jamuna Press*, 2006. *Cited on page:* 117.

[MFHM12] Konstantinos Markantonakis, Lishoy Francis, Gerhard Hancke, and Keith Mayes. Practical relay attack on contactless transactions by using NFC mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012. *Cited on pages:* 79 and 95.

[MOV14] Aikaterini Mitrokotsa, Cristina Onete, and Serge Vaudenay. Location leakage in distance bounding: Why location privacy does not work. *Computers & Security*, 45:199–209, 2014. *Cited on pages:* 28 and 85.

[NLD+12] Tyler Nighswander, Brent Ledvina, Jonathan Diamond, Robert Brumley, and David Brumley. GPS software attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 450–461. ACM, 2012. *Cited on page:* 117.

[NN03a] Dragos Niculescu and Badri Nath. Ad hoc positioning system (APS) using AOA. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1734–1743. Ieee, 2003. *Cited on page:* 117.

[NN03b] Dragoş Niculescu and Badri Nath. DV based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280, 2003. *Cited on page:* 117.

[NSSP04] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004. *Cited on page:* 118.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *Public Key Cryptography*, pages 104–118. Springer, 2001. *Cited on pages:* 16 and 149.

[PD15] Pericle Perazzo and Gianluca Dini. Secure positioning with non-ideal distance bounding protocols. In *Computers and Communication (ISCC), 2015 IEEE Symposium on*, pages 907–912. IEEE, 2015. *Cited on page:* 138.

[PJ08] Panagiotis Papadimitratos and Aleksandar Jovanovic. GNSS-based positioning: Attacks and countermeasures. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7. IEEE, 2008. *Cited on page:* 117.

[Rad03] Cristian Radu. *Implementing electronic card payment systems.* Artech House, 2003. *Cited on pages:* 110 and 187.

[RČ08] Kasper Bonne Rasmussen and Srdjan Čapkun. Location privacy of distance bounding protocols. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 149–160. ACM, 2008. *Cited on page:* 28.

[RČ10] Kasper Bonne Rasmussen and Srdjan Čapkun. Realization of RF distance bounding. In *USENIX Security Symposium*, pages 389–402, 2010. *Cited on page:* 146.

[RL+02] C Savarese J Rabaey, Koen Langendoen, et al. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX technical annual conference*, pages 317–327, 2002. *Cited on page:* 117.

[RL13] Michael Roland and Josef Langer. Cloning credit cards: A combined pre-play and downgrade attack on EMV contactless. In *WOOT*, 2013. *Cited on page:* 95.

[RLS13] Michael Roland, Josef Langer, and Josef Scharinger. Applying relay attacks to Google Wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6. IEEE, 2013. *Cited on page:* 79.

[RNTS07] Jason Reid, Juan M Gonzalez Nieto, Tee Tang, and Bouchra Senadji. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 204–213. ACM, 2007. *Cited on page:* 61.

[RÓČ16] Aanjhan Ranganathan, Hildur Ólafsdóttir, and Srdjan Čapkun. SPREE: A spoofing resistant GPS receiver. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 348–360. ACM, 2016. *Cited on page:* 117.

[Sco01] Logan Scott. Anti-spoofing & authenticated signal architectures for civil navigation systems. In *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS/GNSS 2003)*, pages 1543–1552, 2001. *Cited on page:* 117.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption (version 2.0), 2001. *Cited on page:* 58.

[SHS01] Andreas Savvides, Chih-Chieh Han, and Mani B Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 166–179. ACM, 2001. *Cited on page:* 117.

[SLČ17] Mridula Singh, Patrick Leu, and Srdjan Čapkun. UWB with pulse reordering: Securing ranging against relay and physical layer attacks. 2017. *Cited on page:* 146.

[SP05] Dave Singelee and Bart Preneel. Location verification using secure distance bounding protocols. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 7–pp. IEEE, 2005. *Cited on pages:* 62, 72, and 137.

[SP07] Dave Singelée and Bart Preneel. Distance bounding in noisy environments. In *Security and Privacy in Ad-hoc and Sensor Networks*, LNCS 4572, pages 101–115. Springer, 2007. *Cited on pages:* 9, 61, and 180.

[SPS02] Andreas Savvides, Heemin Park, and Mani B Srivastava. The bits and flops of the multilateration primitive for node localization problems. In

Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 112–121. ACM, 2002. *Cited on page:* 117.

[SSW03] Naveen Sastry, Umesh Shankar, and David Wagner. Secure verification of location claims. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 1–10. ACM, 2003. *Cited on page:* 137.

[SW07] Avinash Srinivasan and Jie Wu. A survey on secure localization in wireless sensor networks. *Encyclopedia of Wireless and Mobile communications*, page 126, 2007. *Cited on page:* 118.

[SWL08] Joo-Han Song, Vincent WS Wong, and Victor CM Leung. Secure location verification for vehicular ad-hoc networks. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1–5. IEEE, 2008. *Cited on page:* 138.

[TI10] Onur Tekdas and Volkan Isler. Sensor placement for triangulation-based localization. *IEEE transactions on Automation Science and Engineering*, 7(3):681–685, 2010. *Cited on page:* 3.

[TPRČ11] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Čapkun. On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 75–86. ACM, 2011. *Cited on page:* 117.

[Ust08] Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Designs, Codes and Cryptography*, 46(3):329–342, 2008. *Cited on page:* 181.

[Vau07] Serge Vaudenay. On privacy models for RFID. In *ASIACRYPT*, LNCS 4833, pages 68–87. Springer, 2007. *Cited on pages:* 14 and 85.

[Vau13] Serge Vaudenay. On modeling terrorist frauds. In *Provable Security*, LNCS 8209, pages 1–20. Springer, 2013. *Cited on pages:* 7, 61, 62, 66, and 75.

[Vau15a] Serge Vaudenay. On privacy for RFID. In *Provable Security*, pages 3–20. Springer, 2015. *Cited on pages:* 7, 37, 38, 61, 148, 174, and 175.

[Vau15b] Serge Vaudenay. Privacy failure in the public-key distance-bounding protocol. *IET Information Security*, 2015. *Cited on pages:* 37, 170, and 173.

[Vau15c] Serge Vaudenay. Private and secure public-key distance bounding application to NFC payment. In *Financial Cryptography*, LNCS 8975, pages 207–216, 2015. *Cited on pages:* xv, 7, 8, 9, 11, 12, 13, 37, 38, 39, 45, 46, 57, 58, 62, 75, 88, 148, 149, 175, 176, and 184.

[Vau15d] Serge Vaudenay. Sound proof of proximity of knowledge. In *Provable Security*, LNCS 9451, pages 105–126. Springer, 2015. *Cited on pages:* xv, 7, 11, 37, 38, 61, 62, 148, 173, and 174.

[Wei10] Michael Weiß. Performing relay attacks on ISO 14443 contactless smart cards using NFC mobile equipment. *Master's Thesis in Computer Science, University of Munich*, 2010. *Cited on page:* 95.

[WKC+12] Erik Ramsgaard Wognsen, Henrik Søndberg Karlsen, Marcus Calverley, Mikkel Normann Follin, Bent Thomsen, and H Huttel. A secure relay protocol for door access control. In *Proceedings of the Xii Brazilian Symposium on Information and Computer System Security*. SBC-Sociedade Brasileira de Computação, 2012. *Cited on page:* 79.

[ZCH+13] Yingpei Zeng, Jiannong Cao, Jue Hong, Shigeng Zhang, and Li Xie. Secure localization and location verification in wireless sensor networks: a survey. *The Journal of Supercomputing*, 64(3):685–701, 2013. *Cited on page:* 118.

[ZJUQ08] Sheng Zhong, Murtuza Jadliwala, Shambhu Upadhyaya, and Chunming Qiao. Towards a theory of robust localization against malicious beacon nodes. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1391–1399. IEEE, 2008. *Cited on pages:* 118, 119, 121, and 130.

[ZLFW06] Yanchao Zhang, Wei Liu, Yuguang Fang, and Dapeng Wu. Secure localization and authentication in ultra-wideband sensor networks. *IEEE Journal on Selected areas in communications*, 24(4):829–835, 2006. *Cited on pages:* 118, 119, and 137.

Appendix A

Review of Public-Key DBs

A.1 Brands and Chaum Protocol [BC93]

Brands and Chaum [BC93] introduced the first DB protocol in Figure A.1.

The verifier V knows the public key (pk) of the prover, and the prover P has the corresponding secret key (sk). P picks a random n-bits message a and commits on it (let's denote the commitment by A). V picks an n-bits challenge c as well. P sends the commitment A to V. Then, the challenge phase begins which consists of n rounds. V sends a challenge c_i in each round i which is the i^{th} bit of c. P sends the response r_i which is $a_i \oplus c_i$ where a_i is the i^{th} bit of a. In the verification phase, P signs the transcript $c_1|r_1|...|c_n|r_n$ with sk. Finally, P sends the decommitment of A which is a, ρ and the signature σ. Then, V accepts P if the following conditions hold:

- σ is valid which means that $c_1|r_1|...|c_n|r_n$ is signed by P.
- a, ρ is correct which means $A = \mathsf{Commit}(a; \rho)$.
- Each response r_i is correct which means that $m_i = r_i \oplus c_i$ for all $i \in \{1, ..., n\}$.
- Each response r_i is received on time (for a round trip at distance B).

Security: The protocol in Figure A.1 is DF-secure and MiM-secure [BC93]. The security proofs are based on the secure signature and commitment schemes. It is not TF-secure [BC93] and not DH-secure [CRSČ12]. Clearly, this protocol does not cover privacy because the signature makes the identity of the prover clear.

A probabilistic polynomial time (PPT) adversary breaks DF-security and MiM-security of the Brands-Chaum protocol with the probability $(\frac{1}{2})^n$.

We prove the DF and MiM security of the Brands-Chaum protocol in the model defined in Chapter 2.

Theorem A.1. *If* Commit *is computationally binding, then the Brands-Chaum protocol is DF-resistant. More precisely, any DF attack has a probability of success bounded by* $\sqrt{2^{-n} + \mathsf{negl}}$.

Verifier		Prover
input: pk		$(\mathsf{sk}, \mathsf{pk})$
	initialization phase	
		pick $a \in \mathbb{Z}_2^n$
	$\xleftarrow{A}$	$A = \mathsf{Commit}(a;\rho)$
	challenge phase	
	for $i = 1$ to n	
pick $c_i \in \mathbb{Z}_2$		
start timer_i	$\xrightarrow{c_i}$	receive c_i
receive r_i, stop timer_i	$\xleftarrow{r_i}$	$r_i = a_i \oplus c_i$
	verification phase	
$\mathsf{Verify}_{\mathsf{pk}}(\sigma, \mathsf{transcript})$, verify $A = \mathsf{Commit}(a;\rho)$	$\xleftarrow{a,\rho,\sigma,c,r}$	$\sigma = \mathsf{Sign}_{\mathsf{sk}}(\mathsf{transcript})$
check $\mathsf{timer}_i \leq 2B, r_i = a_i \oplus c_i$	$\xrightarrow{\mathsf{Out}_V}$	

Figure A.1 – The Brands-Chaum Protocol [BC93].

Proof. We consider a DF attack. The prover picks some random tape t then succeeds in the attack with probability $p(t)$ over the distribution of c_i's. We simulate the attack by picking r then running the attack twice with the same r and thus the same commitment $A = \mathtt{Commit}(a;\rho)$. In the first attack, the challenges seen are c. In the second run, if one c_i changes, then the challenges seen is $c' \neq c$. For the first c_i changed, the view of the prover is unchanged at the time the prover must release r_i, so $r_i \neq r'_i$. In this case, the value on which to open the commitment must change as well. If the two runs succeed and if the verifier did not select the same set of challenges, then $a' = r' \oplus c' \neq r \oplus c = a$ is changed so we break the binding property. This happens with probability at least $E(p(t)^2) - 2^{-n}$. Hence, by assumption, we must have $E(p(t)^2) = 2^{-n} + \mathsf{negl}$. The DF attack succeeds with probability $E(p(t))$. Due to the Jensen inequality, we have

$$E(p(t)) = E(\sqrt{p(t)^2}) \leq \sqrt{E(p(t)^2)} \leq \sqrt{2^{-n} + \mathsf{negl}}$$

□

Theorem A.2. *If* Commit *is computationally hiding and the signature scheme resists to existential forgery under chosen message attacks, then the Brands-Chaum protocol is MiM-secure.*

Proof. We consider a MiM game with winning probability p. First, we reduce to a game in which no two prover instances select the same a value. The winning probability is at least $p - \mathsf{poly} \cdot 2^{-n}$ where poly is some polynomial. Second, we reduce to a game in which the signature accepted by $\mathcal{V}$ is not a forgery. The winning probability is at least $p - \mathsf{poly} \cdot 2^{-n} - p_{\mathsf{forge}}$. We have $p_{\mathsf{forge}} = \mathsf{negl}$ thanks to the security of the signature.

For each i, we define a game$_i$ in which winning implies that a verifier instance receives the signature produced by the i^{th} instance of the prover. Let p_i be the winning probability of game$_i$. We have $p \leq \mathsf{poly} \cdot 2^{-n} + p_{\mathsf{forge}} + \sum_i p_i$. The number of i's is polynomial. To bound p_i, we remark that the ith instance of the prover and the matching verifier instance must see the same c and r because otherwise, the signature will not be valid and so the verifier instance will reject the prover (game$_i$ fails). This defines $a = c \oplus r$. We can then reduce game$_i$ to a game in which the adversary is given the signing key to simulate all other prover instances except the i^{th} one. For the i^{th} one, the adversary cannot use the signing key because the winning condition is that the signature has to come from i^{th} prover instance. So, the i^{th} instance commits to a random value a' and the adversary runs a man-in-the-middle attack between this instance and $\mathcal{V}$ which succeeds only if he gets the right $a = a'$. We let m be the number of c'_i that the adversary sends to this prover instance before receiving the corresponding c_i from $\mathcal{V}$. Remark that the adversary should send c'_i before receiving the c_i to be able to receive the response from the prover on time. For the attack to succeed, the adversary must guess $c_i = c'_i$. So, this works with probability 2^{-m}. For the remaining $n-m$ rounds, the adversary must guess the a_i from the commit value only to compute r_i correctly. Guessing a_i is easier than guessing c_i from an information theoretic view point as the adversary has a clue A on a. So, we could assume $m = 0$ without loss of generality. The game reduces to guessing a completely which is impossible due to the computationally binding property. So p_i is negligible. Hence, $p_i \leq 2^{-n} + \mathsf{negl}$. □

Performance: The prover commits on n-bits message and signs $2n$-bits message. In return, the verifier checks the commitment and the signature.

A.2 HPO [HPO13]

Hermans et al. [HPO13] constructed a public-key DB protocol (Figure A.2).

The verifier V and the prover P use elliptic curve cryptography in this protocol. Therefore, the domain parameters of the protocol are an elliptic curve $\mathbb{E}$ defined over the field $\mathbb{Z}_p$ and its subgroup $\mathbb{G}_\ell$ which has prime order ℓ, a generator $G \in \mathbb{G}_\ell$.

V and P have secret-public key pairs $(\mathsf{sk}_V, \mathsf{pk}_V = \mathsf{sk}_V G)$, $(\mathsf{sk}_P, \mathsf{pk}_P = \mathsf{sk}_P G)$, respectively. In the initialization phase, V and P agree on a secret key. For this, P selects two ephemeral secret keys $r_1, r_2 \in \mathbb{Z}_\ell^*$ and sends the ephemeral public keys $R_1 = r_1 G, R_2 = r_2 G$. Similarly, V selects his ephemeral secret key $r_3 \in \mathbb{Z}_\ell^*$ and sends $R_3 = r_3 G$. In the end, both agree on the secret key $a_0 | a_1 = a'_0 | a'_1 = [\mathtt{xcoord}(r_1 R_3)]_{2\ell} = [\mathtt{xcoord}(r_3 R_1)]_{2\ell}$ which is the first 2ℓ bits of `xcoord` function. `xcoord` function maps a point $U = (U_x, U_y)$ to $U_x \bmod \ell$. Then, V selects an element $e \in \mathbb{Z}_\ell^*$ and the first n bits of e represent the challenge bits c_i.

Next, the challenge phase which consists of n rounds begins. In each round i, P computes the response $f_i = a'_{c_i,i}$ when he receives challenge c_i from V. Here, $a'_{c_i,i}$ is the i^{th} bit of a_{c_i}.

In the verification phase, V verifies if the responses arrived on time and whether they are correct. If so, V sends e and P checks if it is consistent with c. Then, P computes a blinding factor (for privacy) $\mathtt{xcoord}(\mathtt{r_2 pk_V})$ and $s = \mathsf{sk}_P + e r_1 + r_2 + \mathtt{xcoord}(r_2 \mathsf{pk}_V)$ and

Verifier		Prover
$(\mathsf{sk}_V, \mathsf{pk}_V)$		$(\mathsf{sk}_P, \mathsf{pk}_P)$
		input: pk_V
	initialization phase	
		pick $r_1, r_2 \in \mathbf{Z}_\ell^*$
pick $r_3, e \in \mathbf{Z}_\ell^*$	$\xleftarrow{R_1, R_2, \mathsf{pk}_P}$	$R_1 = r_1 G,\ R_2 = r_2 G$
$R_3 = r_3 G$	$\xrightarrow{R_3}$	
$a_0 \| a_1 = \mathsf{xcoord}_{2n}(r_3 R_1)$		$a'_0 \| a'_1 = \mathsf{xcoord}_{2n}(r_1 R_3)$
	challenge phase	
	for $i = 1$ to n	
$c_i = \mathsf{bit}_i(e)$		
start timer_i	$\xrightarrow{c_i}$	
stop timer_i	$\xleftarrow{f_i}$	$f_i = a'_{c_i,i}$
	verification phase	
check $\mathsf{timer}_i \leq 2B, f_i = a_{c_i,i}$	$\xrightarrow{e}$	check $\mathsf{bit}_i(e) = c_i$
	$\xleftarrow{s}$	$s = \mathsf{sk}_P + e r_1 + r_2 + \mathsf{xcoord}(r_2 \mathsf{pk}_V)$
$\mathsf{pk}_P = (s - \mathsf{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2$	$\xrightarrow{\mathsf{Out}_V}$	
private output: pk_P		

Figure A.2 – The Hermans-Peeters-Onete (HPO) Protocol [HPO13].

sends s. After receiving s, V computes $\mathsf{pk}_P = (s - \mathtt{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2$. In the end, pk_P is the private output of V and V outputs Out_V.

Security: The HPO protocol is DF-secure, MiM-secure and weak-private (proofs are in [HPO13]) in the security model of Dürholz et al. [DFKO11] (which is different than the model in Section 2.2). It is not strong private [Vau15b] and not TF-secure [HPO13] and not DH-secure because of the attack explained below.

The assumptions on the security and privacy of HPO are "One More Discrete Logarithm (OMDL) [BNPS03]", "x-Logarithm (XL) [BG07]", "Diffie Hellman (DH)", "extended Oracle Diffie Hellman (eODH) [HPO13]" and "Conjecture 1 [HPO13]". Some are ad-hoc assumptions.

A PPT adversary breaks the DF-security of HPO with the probability $(\frac{3}{4})^n$ and MiM-security of HPO with the probability $(\frac{1}{2})^n$.

DH-attack to HPO: Our DH-attack is shown in Figure A.3. The malicious prover P knows the public key $\mathsf{pk}_{P'}$ of an honest and close prover P'. So, he picks his public key pk_P as $\mathsf{pk}_{P'} - K$ where $K = kG$ and k is selected from $\mathbb{Z}_\ell^*$. P does not involve into the interaction between V and P' in the initialization phase and the challenge phase and sees all transcripts between them. After the challenge phase, P replaces s generated by P' with $s' = s - k$. When V receives s', he computes $(s' - \mathtt{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2$ which

Verifier		**Honest Prover**		**Malicious Prover**
$(\mathsf{sk}_V, \mathsf{pk}_V)$		$(\mathsf{sk}_{P'}, \mathsf{pk}_{P'})$		$\mathsf{pk}_P = \mathsf{pk}_{P'} - kG, G$ **input**: pk_V
		initialization phase		
		pick $r_1, r_2 \in \mathbf{Z}_\ell^*$		
		$R_1 = r_1G,\ R_2 = r_2G$	$\xrightarrow{R_1,R_2}$	
pick $r_3, e \in \mathbf{Z}_\ell^*$	$\xleftarrow{R_1,R_2}$			
$R_3 = r_3G$	$\xrightarrow{R_3}$			
			$\xleftarrow{R_3}$	
$a_0 \| a_1 = \mathsf{xcoord}_{2n}(r_3R_1)$		$a'_0 \| a'_1 = \mathsf{xcoord}_{2n}(r_1R_3)$		
		challenge phase		
	for $i = 1$ to n			
$c_i = \mathsf{bit}_i(e)$				
start timer_i	$\xrightarrow{c_i}$			
stop timer_i	$\xleftarrow{f_i}$	$f_i = a'_{c_i,i}$		
		verification phase		
check $\mathsf{timer}_i \leq 2B, f_i = a_{c_i,i}$	$\xrightarrow{e}$			
		check $\mathsf{bit}_i(e) = c_i$	$\xleftarrow{e}$	
		$s = \mathsf{sk}_P + er_1 + r_2 +$ $\mathsf{xcoord}(r_2\mathsf{pk}_V)$	$\xrightarrow{s}$	$s' = s - k$
	$\xleftarrow{s'}$			
$\mathsf{pk}_P = (s' -$ $\mathsf{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2$ **private output**: pk_P	$\xrightarrow{\mathsf{Out}_V}$			

Figure A.3 – The DH-attack to the HPO Protocol [HPO13].
[CGMO09]

equals pk_P and accepts P due to the following equation:

$$
\begin{aligned}
(s' - \mathtt{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2 &= (s - k - \mathtt{xcoord}(\mathsf{sk}_V R_2))G - eR_1 - R_2 \\
&= sG - kG - \mathtt{xcoord}(\mathsf{sk}_V R_2)G - eR_1 - R_2 \\
&= (\mathsf{sk}_{P'} + er_1 + r_2 + \mathsf{xcoord}(r_2\mathsf{pk}_V))G \\
&\quad - kG - \mathtt{xcoord}(\mathsf{sk}_V R_2)G - eR_1 - R_2 \\
&= \mathsf{pk}_{P'} - kG = \mathsf{pk}_P
\end{aligned}
$$

One drawback of this attack is that the malicious prover does not hold any secret key corresponding to his pk_P. Depending on how the protocol infrastructure is implemented in practice, key registration may thus fail. However, our general definition for DH allows considering this type of attack in which a malicious prover succeeds to register any public key, as long as it differs from the public key of P'.

Performance: The HPO protocol requires 4 EC multiplications both in the prover and the verifier side. Multiplication on EC group corresponds to exponentiation operation in modular arithmetic and it is more efficient.

A.3 GOR [GOR14a]

Verifier		**Prover**
secret key: sk_V		secret key: sk_P
public key: $\mathsf{pk}_V = \mathsf{sk}_V G$		public key: $\mathsf{pk}_P = \frac{1}{\mathsf{sk}_P} Q$
input: Q		**input**: pk_V
	initialization phase	
		pick $r_1, r_2 \in \mathbf{Z}_\ell^*$, $R_1 = r_1 G$, $R_2 = r_2 G$, pick $\mathsf{HEnc}, \mathsf{HDec}$
pick $r_3, e \in \mathbf{Z}_\ell^*$, $R_3 = r_3 G$	$\xleftarrow{R_1, R_2, \mathsf{HEnc}, h, \pi}$	$h = \mathsf{HEnc}(\mathsf{pk}_P), \pi = \mathsf{NIZK}(h)$
check π, $h' = r * h$	$\xrightarrow{R_3, h'}$	
$a_0 \| a_1 = \mathsf{xcoord}_{2n}(r_3 R_1)$		$a'_0 \| a'_1 = \mathsf{xcoord}_{2n}(r_1 R_3)$
	challenge phase for $i = 1$ to n	
$c_i = \mathsf{bit}_i(e)$		
start timer_i	$\xrightarrow{c_i}$	
stop timer_i	$\xleftarrow{f_i}$	$f_i = a'_{c_i, i}$
	verification phase	
	$\xrightarrow{e}$	check $\mathsf{bit}_i(e) = c_i$
check $\mathsf{timer}_i \leq 2B, f_i = a_{c_i, i}$	$\xleftarrow{S}$	$S = \mathsf{sk}_P \mathsf{HDec}(h') + eR_1 + R_2 + r_2 \mathsf{pk}_V$
check $rQ = S - \mathsf{sk}_V R_2 - eR_1 - R_2$	$\xrightarrow{\mathsf{Out}_V}$	

Figure A.4 – The Gambs-Onete-Robert protocol (GOR) [GOR14a].

The GOR protocol [GOR14a] in Figure A.4 is similar to HPO, but adapted to provide anonymous authentication: the verifier does not identify the prover but rather checks that he belongs to a given group. The secret/public key $(\mathsf{sk}_V, \mathsf{pk}_V)$ setup of the verifier (V) and secret key sk_P of the prover (P) is the same, only difference is in the public key pk_P of the prover which is $\frac{1}{\mathsf{sk}_P} Q$. Here, $Q = \mathsf{sk}_P \prod_{i=1}^{k-1} \mathsf{sk}_{P_i}$ and each sk_{P_i} is the secret key of a prover P_i and k is the number of provers in the system.

The other differences are in the initialization and the verification phases. In the initialization phase, the prover computes R_1, R_2 as in HPO and additionally uses a homomorphic encryption algorithm HEnc and its decryption algorithm HDec. He encrypts his public key pk_P with the key pk_V and obtains h. In the end of the initialization phase, he sends R_1, R_2, h and π which is a non-interactive zero knowledge (NIZK) that h is well formed. After receiving all values, V computes R_3 as in HPO. Additionally, he checks if π is valid. If everything is valid, V sends R_3 and $h' = r * h$ where $r \in \mathbb{Z}_\ell^*$. The rest of the GOR is the same with HPO until the verification phase. Here, P computes and sends $S = \mathsf{sk}_P \mathsf{HDec}(h') + eR_1 + R_2 + r_2 \mathsf{pk}_V$. V accepts if $S - \mathsf{sk}_V R_2 - eR_1 - R_2$ equals rQ, all responses are correct and on time.

Security: GOR is DF-secure and MiM-secure [GOR14a] but it is not TF-secure. As it is constructed to be anonymous to the verifier, it is not DH-secure but this is purposely done. It is constructed to have strong-privacy but unfortunately, it is shown in [Vau15b] by an attack that it is neither strong-private nor weak-private. Subsequently to this attack, GOR was modified. The modified version of GOR [GOR14b] is resistant the attack in [Vau15b]. Some attacks following the DF game work with probability $\left(\frac{3}{4}\right)^n$. Some attacks following the MiM game work with probability $\left(\frac{1}{2}\right)^n$.

The assumptions for DF-security are the DDH assumption, sound NIZK proof and the discrete logarithm assumption. The assumptions for privacy Replayable CCA security (IND-RCCA) and Indistinguishability of Keys against CCA (IK-CCA) security.

Performance: The prover in GOR does 4 EC multiplications, 1 encryption and 1 NIZK. The verifier verifies the NIZK and does 4 EC multiplications. The EC operations do not require a lot of computations, but NIZK is expensive for a DB protocol.

What is proven in NIZK in GOR is not explained in a clear statement in both versions. However, it is critical as it may cause an authentication problem. The prover has to prove that he constructed the encryption correctly and also the public key in the encryption belongs to $\{\mathsf{pk}_1, ..., \mathsf{pk}_k\}$. If this proof is not considered in GOR in this case any arbitrary participant can authenticate himself without knowing any sk_i. Otherwise, it is very costly because the prover needs to do k times OR-proof (e.g. the public key in HEnc includes pk_1 or pk_2 or ... or pk_k).

A.4 ProProx [Vau15d]

Vaudenay [Vau15d] constructed a public-key protocol in Figure A.5. The protocol specifications are the following:

The verifier (V) has the public key $\mathsf{pk} = \mathsf{Com}_H(\mathsf{sk})$ of the prover. Com_H is a set commitments. Each commitment commits the hash of each bit of the input sk where H is a hash function. sk is the secret key of the prover P. P first picks $\{a_{i,j} \in \mathbb{Z}_2\}_{i=1,j=1}^{n,s}$ and a corresponding random value $\rho_{i,j}$. Then, he commits all $a_{i,j}$'s and gets $A_{i,j} = \mathsf{Com}(a_{i,j}; \rho_{i,j})$. After, he sends all $A_{i,j}$'s.

The challenge phase consists of ns rounds. In each round $(i, j) \in \{1, ..., n\} \times \{1, ..., s\}$, V sends $c_{i,j} \in \mathbb{Z}_2$. As a response, P sends $r_{i,j} = a_{i,j} \oplus c_{i,j} b_{i,j} \oplus c_{i,j} \mathsf{sk}_j$. where sk_j is the j^{th} bit of sk.

In the verification phase, V first checks if all responses arrived on time. If everything goes well, P and V run zero-knowledge proof (ZKP) to show that the responses are consistent with $A_{i,j}$ and pk_j for each (i, j) where pk_j is the j^{th} bit of pk. The details of ZKP is showed in Figure A.6.

Security: ProProx is DF-secure, MiM-secure, extractor based TF-secure [BV14] and DH-secure as shown in [Vau15d]. However, it is not strong private or weak private. Some attacks following the DF game work with probability $\left(\frac{1}{\sqrt{2}}\right)^{ns}$. Some attacks following the

Verifier		**Prover**
input: pk	$\mathsf{pk} = \mathsf{Com}_H(\mathsf{sk})$	$(\mathsf{sk}, \mathsf{pk})$
	initialization phase	
	for $i = 1$ to n and $j = 1$ to s	
		pick $a_{i,j} \in \mathbf{Z}_2$, $\rho_{i,j}$
	$\xleftarrow{\quad A_{i,j} \quad}$	$A_{i,j} = \mathsf{Com}(a_{i,j}; \rho_{i,j})$
	challenge phase	
	for $i = 1$ to n and $j = 1$ to s	
pick $c_{i,j} \in \mathbf{Z}_2$		
start $\mathsf{timer}_{i,j}$	$\xrightarrow{\quad c_{i,j} \quad}$	receive $c_{i,j}$
receive $r_{i,j}$, stop $\mathsf{timer}_{i,j}$	$\xleftarrow{\quad r_{i,j} \quad}$	$r_{i,j} = a_{i,j} \oplus c_{i,j} b_{i,j} \oplus c_{i,j} \mathsf{sk}_j$
	verification phase	
check $\mathsf{timer}_{i,j} \leq 2B$		
$z_{i,j} = A_{i,j} \left(\theta^{b_{i,j}} \mathsf{pk}_j\right)^{c_{i,j}} \theta^{-r_{i,j}}$	$\xleftrightarrow{ZKP_\kappa(z_{i,j}:\zeta_{i,j}; i \in I_j, j=1,\ldots,s)}$	$\zeta_{i,j} = \rho_{i,j} H(\mathsf{sk}, j)^{c'_{i,j}}$
	$\xrightarrow{\quad \mathsf{Out}_V \quad}$	

Figure A.5 – ProProx: a Sound and Secure PoPoK. [Vau15d]

Verifier		**Prover**
input: z	$z = \zeta^2$	(ζ, z)
	$\xleftarrow{\quad \mathsf{pk} \quad}$	generate sk, $\mathsf{pk} = \mathsf{Gen}(\mathsf{sk})$
pick $e \in \mathbf{Z}_2$, pick r	$\xrightarrow{\mathsf{Commit}_{\mathsf{pk}}(e;r)}$	
	$\xleftarrow{\quad h \quad}$	pick $g \in G$, $h = g^2$
	$\xrightarrow{\quad e,r \quad}$	open commitment
check $z^e = \ell^2 h^{-1}$, $\mathsf{pk} = \mathsf{Gen}(\mathsf{sk})$	$\xleftarrow{\quad \ell,\mathsf{sk} \quad}$	$\ell = g\zeta^e$
	$\xrightarrow{\quad \mathsf{Out}_V \quad}$	

Figure A.6 – $\mathsf{ZKP}(z : \zeta)$: a Sound and Zero-Knowledge Proof for z Being a Square.

MiM game work with probability $\left(\frac{1}{2}\right)^{ns}$. Some attacks following the DH game work with probability $\left(\frac{1}{2}\right)^{ns}$. Some attacks following the TF game work with probability $\left(\frac{1}{\sqrt{2}}\right)^{ns}$.

ProProx is secure under the assumptions that Com is a homomorphic bit commitment, Com is a perfectly binding and computationally hiding, ZKP is sound and computationally zero-knowledge proof of membership.

Performance: ProProx has heavy computations both on the prover and the verifier side. The prover computes ns commitments, ns exponentiations for ZKP. Similarly, the verifier computes $n(s+1)$ exponentiations and ns inversions.

Now, we explain a version of ProProx which is called as eProProx [Vau15a].

Variant: eProProx [Vau15a] is a variant of ProProx. It has an extra phase showed in Figure A.7 for privacy before the initialization phase of ProProx. In this phase, the

Verifier		**Prover**
$(\mathsf{sk}_V,\mathsf{pk}_V)$		$(\mathsf{sk}_P,\mathsf{pk}_P)$
		input: pk_V
		pick $\delta=(\delta_1,\ldots,\delta_s)$
$\delta\|\mathsf{pk}=\mathsf{Dec}_{\mathsf{sk}_V}(B)$	$\xleftarrow{\quad B \quad}$	$B=\mathsf{Enc}_{\mathsf{pk}_V}(\delta\|\mathsf{pk}_P)$
$\mathsf{pk}'_j=\mathsf{pk}_j\mathsf{Com}(0;\delta_j)$		$H'(.,j)=H(.,j)\delta_j$
	$\xleftrightarrow{\mathsf{ProProx}_{H'}(\mathsf{pk}')}$	
private output: pk		

Figure A.7 – eProProx: a Privacy Extension for ProProx.

Verifier		**Prover**
$(\mathsf{sk}_V,\mathsf{pk}_V)$		$(\mathsf{sk}_P,\mathsf{pk}_P)$
		input: pk_V
pick N	$\xrightarrow{\quad N \quad}$	pick s, $\sigma=\mathsf{Sign}_{\mathsf{sk}_T}(N)$
$s\|\mathsf{pk}\|\sigma=\mathsf{Dec}_{\mathsf{sk}_P}(e)$	$\xleftarrow{\quad e \quad}$	$e=\mathsf{Enc}_{\mathsf{pk}_V}(s\|\mathsf{pk}_P\|\sigma)$
$\mathsf{Verify}_{\mathsf{pk}}(\sigma,N)$	$\xleftrightarrow{\mathsf{OTDB}(s)}$	
private output: pk		

Figure A.8 – privDB: Private Public-Key DB [Vau15c].

prover first picks $\delta_1,...,\delta_s$ and encrypts all δ_i's concatenated with pk_P all with pk_V. Then, sends the encryption B to V. V decrypts B and gets δ_i's and $\mathsf{pk}=\mathsf{pk}_P$. Then, he commits each bit of pk and gets $\mathsf{pk}'_j=\mathsf{pk}_j\mathsf{Com}(0;\delta_j)$. Meanwhile, P computes $H'(.,j)=H(.,j)\delta_j$. In the end, they run $\mathsf{ProProx}_{H'}(\mathsf{pk}')$.

Security: eProProx preserves the security of ProProx and additionally it is strong private as shown in [Vau15a].

Performance: eProProx does not improve the performance of the ProProx. Conversely, it adds extra s commitments and 1 encryption on the prover side. It also adds extra s commitments and 1 decryption on the verifier side.

A.5 PrivDB [Vau15c]

PrivDB is introduced by Vaudenay [Vau15c]. The verifier (V) and the prover (P) have their own private/public key pairs $(\mathsf{sk}_V,\mathsf{pk}_V)$ and $(\mathsf{sk}_P,\mathsf{pk}_P)$, respectively. First, V picks N and sends it to P. P picks a secret s and signs N with sk_P. Then, he encrypts $s||\mathsf{pk}_P||\sigma$ with pk_V where σ is the signature. After, he sends the encryption e to V. V first decrypts e and learns s, $\mathsf{pk}=\mathsf{pk}_P$ and σ. He verifies the signature and if the verification is ok then V and P run the OTDB protocol which is shown in Chapter 2, in Figure 2.1 with s.

Security: PrivDB is DF-secure, MiM-secure, DH-secure as shown in [Vau15c]. Additionally, PrivDB is strong private [Vau15c]. However, it is not TF-secure.

The assumptions on the security of PrivDB are EF-CMA secure signature scheme and IND-CCA secure encryption scheme.

A PPT adversary breaks DF-security of PrivDB with the probability $(\frac{3}{4})^n$ and MiM-security and DH-security of PrivDB with the probability $(\frac{1}{2})^n$.

Performance: PrivDB requires computation of a signature and an encryption on the prover's side. It requires a decryption and a verification of the signature on the verifier's side.

A.6 TREAD [ABG$^+$17]

The protocol TREAD is constructed by Avoine et al. [ABG$^+$17]. It has symmetric and public-key variants but we explain the protocol with the public-key variant.

The initialization phase of TREAD is very similar to PrivDB [Vau15c]. The prover P picks a bit string $\alpha||\beta$ from $\{0,1\}^{2n}$ and signs it with its secret-key sk_P. Then, he encrypts the signature σ_P, his identity id_P and $\alpha||\beta$ with the public key of the verifier pk_V and sends the encryption to the verifier V together with id_P. The verifier decrypts the encryption and then verifies if the signature is valid. If it is not valid, he aborts. Otherwise, he picks a message m from $\{0,1\}^{2n}$ and sends it to P. Then, the challenge phase begins. In each round i, V sends a challenge bit c_i. If $c_i = 0$, P replies with a response $r_i = \alpha_i$. Otherwise, it replies with $r_i = \beta_i \oplus m_i$. Here, α_i and β_i are the i^{th} bits of α and β, respectively. In the verification phase, V checks if all responses arrived on time and correctly.

Security: TREAD is DF, DH and MiM secure. It is also claimed to be simulator based TF (SimTF) secure in the DFKO model [DFKO11]. However, we find problems in its SimTF proof. Therefore, their claim is not correct until the correct proof is provided. We give more details about it below.

It is strong private and if the signature scheme is a group signature and id_P is the identifier of a group then it is anonymous as well.

The assumptions are IND-CCA security on the encryption scheme and EF-CMA security on the signature scheme.

The security and privacy proofs are in [ABG$^+$17]. A PPT adversary can break the MiM, DH and DF security with the probability $(\frac{3}{4})^n$.

Performance: Its performance is the same with PrivDB [Vau15c].

Problems in the SimTF-security Proof of TREAD: We first define a tainted session which is used in the definition of SimTF-security. Each session of a distance-bounding protocol is associated with a unique identifier sid. The sessions can be between

$$
\begin{array}{rcl}
\textbf{Verifier} & & \textbf{Prover} \\
(\mathsf{sk}_V, \mathsf{pk}_V) & & (\mathsf{sk}_P, \mathsf{pk}_P) \\
 & & \textbf{input}\text{: } \mathsf{pk}_V \\
 & \textbf{Initialization Phase} & \\
 & & \alpha||\beta \in \{0,1\}^{2n} \\
 & & \sigma_P = \mathsf{Sign}_{\mathsf{sk}_P}(\alpha||\beta||id_P) \\
\alpha||\beta||idP||\sigma_P = \mathsf{Dec}_{\mathsf{sk}_V}(e) & \xleftarrow{\quad e, id_P \quad} & e = \mathsf{Enc}_{\mathsf{pk}_V}(\alpha||\beta||idP||\sigma_P) \\
\textbf{if } \mathsf{Verify}_{\mathsf{pk}_P}(\sigma_P) \to 0\text{: abort} & & \\
\textbf{pick } \ m \in \{0,1\}^{2n} & \xrightarrow{\quad m \quad} & \\
 & \textbf{Challenge Phase} & \\
 & \textbf{for } i = 1 \textbf{ to } n & \\
\text{pick } c_i & & \\
\text{start } \mathsf{timer}_i & \xrightarrow{\quad c_i \quad} & \textbf{if } c_i = 0, r_i = \alpha_i \\
\text{stop } \mathsf{timer}_i & \xleftarrow{\quad r_i \quad} & \textbf{else } r_i = \beta_i \oplus m_i \\
\textbf{store } \mathsf{timer}_i & & \\
 & \textbf{Verification Phase} & \\
\textbf{if } |\{i : r_i \text{ and } \mathsf{timer}_i \text{ correct}\}| = n & & \\
\mathsf{Out}_V = 1 & \xrightarrow{\quad \mathsf{Out}_V \quad} &
\end{array}
$$

Figure A.9 – TREAD [ABG+17]

the prover and the verifier, the prover and the adversary, and the adversary and the verifier. Let us define a function $\mathsf{clock}(.,.)$ where it has inputs session id and a message and where it outputs the arrival time of the message to the receiver party in the session. Consider a verifier and adversary session sid and consider consecutive messages m_k, m_{k+1} for $k \geq 1$ with m_k is received by the adversary in challenge phase of sid. sid is called *tainted*, if there exists a prover-adversary session sid' such that for any m_i

$$\mathsf{clock}(sid, m_k) < \mathsf{clock}(sid', m_i) < \mathsf{clock}(sid, m_{k+1})$$

Basically, a session is tainted if the adversary communicates with the prover after receiving a challenge.

Now, we give the exact definition of SimTF that the authors used in [ABG+17].:

Definition A.3 (SimTF Security [ABG+17])**.** *"For a DB authentication scheme DB, a (t, q_V, q_P, q_{obs})-terrorist fraud adversary pair $(\mathcal{A}, P)$ and a simulator S running in time t_S, the malicious prover P and his accomplice $\mathcal{A}$ win against DB if $\mathcal{A}$ authenticates in at least one of q_V adversary-verifier sessions without tainting it with probability $p_{\mathcal{A}}$, and if S authenticates in one of q_V sessions with the view of $\mathcal{A}$ with probability p_S, then $p_A \leq p_S$."*

Note that the missing quantifiers in SimTF-definition makes hard to analyze the SimTF-security proof. So, we assume that for all (t, q_V, q_P, q_{obs})-terrorist fraud adversary pair $(\mathcal{A}, P)$, there exists a simulator S running in time t_S as defined in the definition. There are two theorems related to SimTF-security of TREAD in [ABG+17].

- The proof of Theorem 4 in [ABG$^+$17] is not correct. Theorem 4 states that "if the challenges are drawn uniformly at random by the verifier, TREAD is SimTF-resistant' and prove that for any $(\mathcal{A},P)$, they can construct a simulator where $p_S = p_{\mathcal{A}}$. We show that for the following attack by $(\mathcal{A},P)$ in Figure A.10, $p_S \neq p_{\mathcal{A}}$. In the attack, the malicious prover sends e, id_P as in the protocol to $\mathcal{A}$. Then, P receives m sent by the verifier. At this point, P guesses the first k challenges $\tilde{c_1}, \tilde{c_2}, ..., \tilde{c_k}$ and sends their corresponding responses $\tilde{r_1}, \tilde{r_2}, ..., \tilde{r_k}$ to $\mathcal{A}$. Then, the challenge phase begins. The adversary replies the first k challenges with the given responses. Note that, if the guessed challenge by P is not the same with the challenge sent by the verifier, the response may not be correct. Before receiving $k+1^{st}$ challenge, $\mathcal{A}$ sends all the challenges received from the verifier $c_1, c_2, ..., c_k$ to P. P checks if $c_1, c_2, ..., c_k$ is equal to guessed challenges $\tilde{c_1}, \tilde{c_2}, ..., \tilde{c_k}$. If they are equal, then P gives correct responses $R^0 = r^0_{k+1}, ..., r^0_n$ and $R^1 = r^1_{k+1}, ..., r^1_n$ where r_i^b is the response for the challenge $c_i = b$. Otherwise, it aborts and $\mathcal{A}$ may continue the next $n-k$ rounds by guessing the responses.

  Let us analyze the success probability of $(\mathcal{A},P)$ when $q_V = 1$:

$$
\begin{aligned}
p_{\mathcal{A}} &= \sum_{i=0}^{k} \mathsf{Pr}[(\mathcal{A},P) \text{ wins} \wedge i \text{ challenges out of the first } k \text{ chalenges guessed}] \\
&= \sum_{i=0}^{k-1} \left(\frac{1}{2}\right)^n \binom{k}{i} \left(\frac{1}{2}\right)^{k-i} + \left(\frac{1}{2}\right)^k \\
&= \left(\frac{1}{2}\right)^n \left(\left(\frac{3}{2}\right)^k - 1\right) + \left(\frac{1}{2}\right)^k \\
&= \left(\frac{3}{4}\right)^k \left(\frac{1}{2}\right)^{n-k} - \left(\frac{1}{2}\right)^n + \left(\frac{1}{2}\right)^k
\end{aligned}
$$

  Remark that $p_{\mathcal{A}}$ is equivalent to $\left(\frac{1}{2}\right)^k$ for $k \approx \lambda n$ where $\lambda \leq \frac{\log 2}{\log 3}$.

  We consider the simulator described in the proof of Theorem 4 in [ABG$^+$17]. The simulator's view differs according to the abort by the prover. The view of the simulator from the adversary-prover session where the prover aborts is e, m and the responses of the adversary $\bar{r}_1, \bar{r}_2, ..., \bar{r}_k, \tilde{r}_{k+1}, ..., \tilde{r}_n$. The view of the simulator from the adversary-prover session which the prover does not abort $e, m, \bar{r}_1, \bar{r}_2, ..., \bar{r}_k, R^0, R^1$. The simulator runs number of q_S session with the verifier. In each session, it sends e to the verifier and receives m' from the verifier. In the challenge phase, the simulator responds to each first k-challenge as follows: if $c_i = 0$, it responds with $\tilde{r}_i$, otherwise, it responds with $\tilde{r}_i \oplus m_i \oplus m'$. For the rest of the $n-k$-challenges, the simulator having the view of the adversary where the prover aborts replies randomly. The simulator having the view of the adversary where the prover did not abort responds with $R^0[i]$ if $c_i = 0$ and $R^1[i] \oplus m_i \oplus m'_i$ if $c_i = 1$. Below, we upper bound p_S for any simulator using q_S sessions.

  Let's analyze p_S. We denote $\mathsf{Pr}[S \text{ wins one of } q_S\text{-sessions} | P \text{ aborts}]$ by p_{S_1} and

$\mathsf{Pr}[S$ wins one of q_S-sessions$|\neg P$ aborts$]$ by p_{S_2}. So, p_S is as follows:

$$p_S = p_{S_1}\mathsf{Pr}[P \text{ aborts}] + p_{S_2}\mathsf{Pr}[\neg P \text{ aborts}])$$

For any S, we have $p_{S_1} \leq q_S\left(\frac{1}{2}\right)^{n-k}$ as the simulator having the view from the aborted adversary does not have any information about the last $n-k$-responses. We take $\lambda = \frac{1}{3}$, so $k \approx \frac{n}{3}$, hence p_{S_1} is negligible against $p_{\mathcal{A}}$.

The simulator having the view of the non-aborted adversary replies the last $n-k$ rounds correctly. However, it still needs to reply correctly to the first k-challenges since it only knows how to answer the challenges $\tilde{c}_1, \tilde{c}_2, ..., \tilde{c}_k$. After j-sessions with the verifier, we let i denote the number of challenges which are different from the one known. Let d, e be a constant between 0 and 1. As j is polynomially bounded, we assume that n is large enough so that $e2^{dn} > j$. If $i \leq d.n$, the probability of success is bounded by 1. If $i > dn$, the simulator has tried up to j combinations of out of 2^i, so the probability of success is bounded by $\frac{1}{2^i - j}$, thus by $\frac{1}{(1-e).2^i}$. Therefore, the success probability of the simulator after j attempts can be bounded as follows:

$$\begin{aligned}
p_j &\leq \sum_{i>dn} \binom{k}{i} \left(\frac{1}{2}\right)^k \frac{1}{2^i(1-e)} + \sum_{i \leq dn} \binom{k}{i} \left(\frac{1}{2}\right)^k \\
&\leq \sum_{i} \binom{k}{i} \left(\frac{1}{2}\right)^k \frac{1}{2^i(1-e)} + \mathsf{Pr}[i \leq dn] \\
&= \left(\frac{3}{4}\right)^k (1-e)^{-1} + \mathsf{Pr}[i \leq dn]
\end{aligned}$$

We take d such that $d < \frac{1}{2}$. So, p_j is negligible. As $p_{S_2} \leq \sum_{j \leq q_S} p_j$, p_{S_2} is negligible too. Now, we have

$$p_S = p_{S_1}\mathsf{Pr}[P \text{ aborts}] + p_{S_2}\mathsf{Pr}[\neg P \text{ aborts}]) \leq p_{S_1} + p_{S_2}\left(\frac{1}{2}\right)^k$$

Since p_A is equivalent to $\left(\frac{1}{2}\right)^k$, p_{S_1} is negligible against p_A, and p_{S_2} is negligible, p_S is negligible against p_A. This contradicts with Theorem 4 in [ABG$^+$17].

- The proof of Theorem 5 in [ABG$^+$17] is not correct. Theorem 5 states this: "For any adversary $\mathcal{A}$ authenticating with the help of a prover with non-negligible probability, there is an algorithm amplify using the internal view of $\mathcal{A}$ and oracle access to a verifier such that after polynomial number of steps, $\mathsf{Pr}[\mathsf{amplify}$ authenticates$] = 1$, almost surely. ".

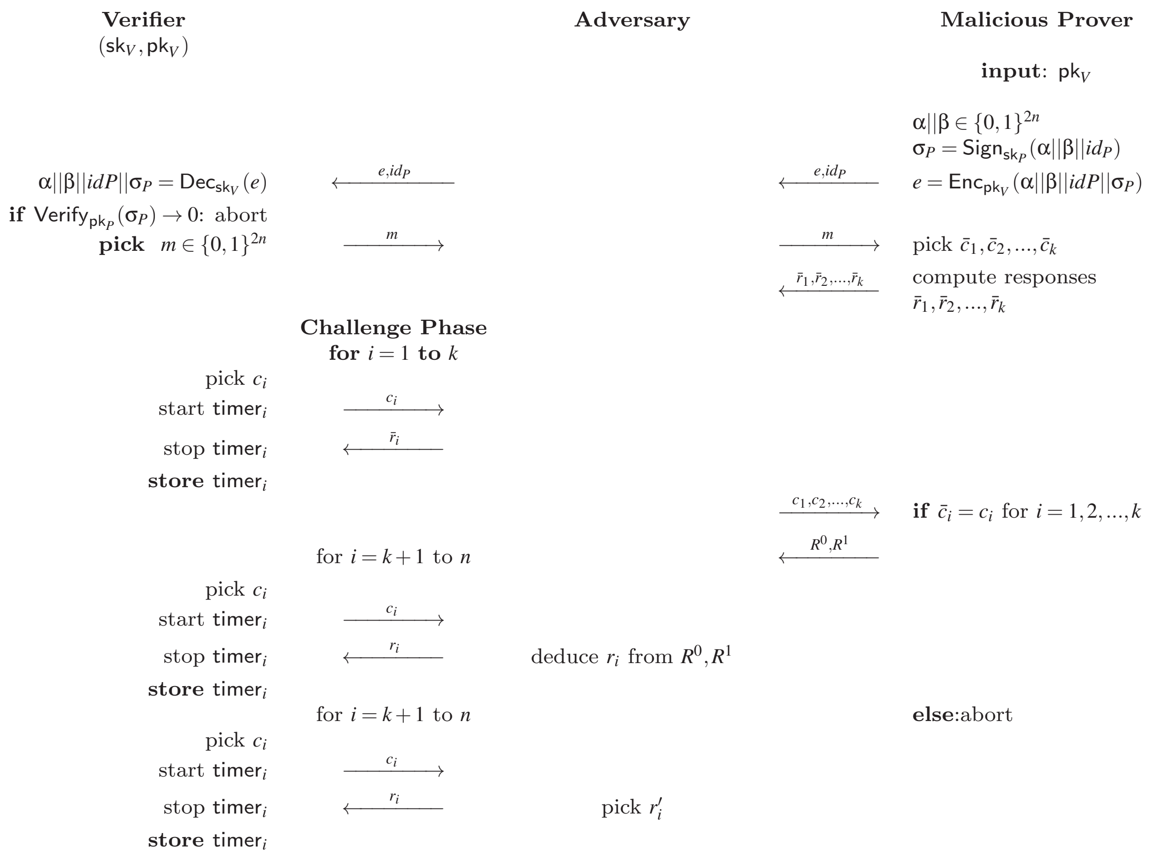

Figure A.10 – A TF-attack that contradicts with the Sim-TF-security of TREAD.

Consider another TF attack where the malicious prover gives all responses R^0, R^1 to $\mathcal{A}$ when n is even and gives no response when n is odd. In this case, $p_{\mathcal{A}}$ is 1 when n is even and $\frac{1}{2^k}$ when n is odd. So, $p_{\mathcal{A}}$ is not negligible as assumed in Theorem 5. However, we cannot construct an amplifier as described in the proof of Theorem which makes the success probability of the an algorithm $\mathsf{amplify}$ 1 for every n. The problem in the proof of Theorem 5 comes from the fact that two quantifiers have been exchanged when dealing with the "non-negligible" notion.

Remark: All of the protocols reviewed above are considered in a noiseless channel which means that the prover always receives the challenge sent by the verifier and the verifier always receives the response sent by the prover in the challenge phase. However, in real world, a noiseless channel is hard to achieve. This problem in DB protocols was introduced by Singelée and Preneel [SP07]. All of the public-key DB protocols can be adapted to noisy channels by changing number of correct response from n to τ to be accepted by the verifier. τ can be determined based on the noise probability.

Appendix **B**

More results about D-AKA security model

The Extended Canetti-Krawczyk (eCK) Security Model [LLM07]

The eCK security model consists of t parties with their certificated public keys. The key exchange protocol is executed between two parties A and B. When A starts a key exchange protocol with B, it is called as a session and A is the owner of the session and B is the peer. A (initiator) starts the protocol by sending a message $\mathsf{M_A}$, then B (responder) responds with a message $\mathsf{M_B}$. The session id *sid* corresponds to an instance of A or B.

There is a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ controlling all communication and some instances. The activation of the parties starts by `Send`(A,B,message) (or `Send`(B,A,message)). Besides `Send`, $\mathcal{A}$ can do following queries:

- `Long-Term Key Reveal`(A): Outputs the long term public-key of A.
- `Ephemeral Key Reveal(`*sid*`)` Outputs an ephemeral key of a session *sid*.
- `Reveal(`*sid*`)`: Outputs the session key of a completed session *sid*.

KA Protocol	Efficiency	Security
MQV [LMQ+03]	2.5	unproven
HMQV [Kra05]	2.5	CK
KEA+ [LM06]	3	CK
NAXOS [LLM07]	4	eCK
CMQV [Ust08]	3	eCK
Nonce-DH	1	D-AKA

Table B.1 – Existing KA protocols with their security and efficiency. Efficiency column shows the number of exponentiation done by per party.

- `Test`(sid): If sid is clean then outputs s by running the query $\texttt{Reveal}(sid)$. If $b=1$, outputs $s \leftarrow \{0,1\}^{\lambda}$ if $b=0$ (λ is the size of the session key).

  The advantage is the difference of the probability that $\mathcal{A}$ gives 1 for $b=0$ and $b=1$.

A clean session is basically a session where winning the game for $\mathcal{A}$ is not trivial. See [LLM07] for more details.

Theorem B.1. *If a key agreement protocol is eCK secure [LLM07], then it is D-AKA secure.*

Proof. Let's assume that there is an adversary $\mathcal{A}$ playing D-AKA game. We construct an adversary $\mathcal{B}$ simulating the D-AKA game and playing the eCK game. $\mathcal{B}$ receives all the public keys in the eCK game. $\mathcal{B}$ first picks two parties A and B. Then, he creates a session sid between them by sending the query $\texttt{Send}$(A,B, message) and he assigns the ephemeral public key of B as a nonce N. Then, he sends the query $\texttt{Test}(sid)$ and receives s_b. Finally, he sends $s_b, N, \mathsf{pk}_B, \mathsf{pk}_A$ to $\mathcal{A}$. Whenever $\mathcal{A}$ calls the oracle $O_B(\mathsf{pk}_{A'})$, $\mathcal{B}$ creates a new session sid' with A' on behalf of B as explained above. Similarly, he assigns the ephemeral public key of B as a nonce N'. After, he sends the query $\texttt{Reveal}(sid')$ and receives the session key s'. As a response of $O_B(\mathsf{pk}_{A'})$, he sends s', N' to $\mathcal{A}$. In addition, whenever $\mathcal{A}$ calls the oracle $O_A(\mathsf{pk}_{B'}, N'')$, first, $\mathcal{B}$ checks if $(\mathsf{pk}_{B'}, N'')$ equals (pk_B, N). If it is not equal, he creates a new session sid'' on behalf of B' with the ephemeral public key N'' and calls the oracle $\texttt{Reveal}(sid'')$ to receive the session key s''. Then, he responds to $\mathcal{A}$ with s''. In the end, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. The simulation of D-AKA game is perfect. So the advantage of $\mathcal{B}$ equals to the advantage of $\mathcal{A}$. Therefore, because the advantage of $\mathcal{B}$ is negligible, the advantage of $\mathcal{A}$ is negligible as well. □

As a result of Theorem B.1, we can conclude any eCK secure key agreement protocol can be used in Eff-pkDB. However, we suggest using D-AKA secure key agreement protocols as they may require less public-key operations.

Appendix **C**

Security implications in SHM and PM

First, let us define "Null conversion". It is a transformation of a protocol DB' with $(\mathcal{K}, V, P', B)$ in PM into another protocol DB with $(\mathcal{K}, V, P, B, H)$ in SHM where P and H are described below[1]:

$\underline{P}$	$\underline{H(K)}$
`ask` key	`send` K
`receive` K	
`run` $P'(K)$	

This conversion shows that, if we have a counterexample protocol in PM which is X-secure but not Y-secure (X,Y $\in$ {DF, DH, MiM, TF}), then the same counterexample applies for its null conversion. Hence, **any non-implication in PM is correct for SHM as well.**

We have already explained that TF-security implies DF, DH and MiM security in SHM and PM. Now, we show the other relations between these security notions. We give our counterexamples in PM for simplicity.

DH $\rightarrow$ DF: It is clear that DH-security implies DF-security in SHM and also in PM. But, there is no such relation between DF/DH-security and MiM-security as explained below.

DF $\nrightarrow$ DH and DF $\nrightarrow$ MiM: A simple counterexample for a DF-secure protocol which is neither MiM nor DH-secure is an 'echo' protocol.

In an 'echo' protocol, the prover authenticates itself and then the challenge phase begins. P receives a challenge(s) from V and P responds with the challenge(s) itself. If P replies with the same challenge(s), V computes the elapsed time between the sending the challenge and receiving the response. Thus, V can decide if the proximity of P is less than B. Clearly, this is DF-secure because P cannot correctly reply before seeing the challenge. So, P cannot show itself closer than its proximity. But, echo protocol is not DH and MiM-secure. It is not DH-secure because a far-away and malicious prover

[1] Remark that H leaks the key in Null conversion because its algorithm is designed like this. This exemplifies that in our model we do not have any restriction (e.g., no leakage of a key) about H's algorithm.

$$
\begin{array}{lcr}
\underline{V}(\mathsf{sk}_V, \mathsf{pk}_V) & & \underline{\mathsf{P}}(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V) \\
 & \textbf{initialization phase} & \\
 & & N \leftarrow D(1^n) \\
A(\mathsf{sk}_V, \mathsf{pk}_V, \mathsf{pk}_P, N) \to s & \xleftarrow{\mathsf{N}, \mathsf{pk}_\mathsf{P}} & B(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V, N) \to s \\
\text{pick } m \in \{0,1\}^{2n} & \xrightarrow{m} & a = s \oplus m \\
 & \xLeftarrow{\mathsf{a}} & \\
 & \textbf{challenge phase} & \\
 & \text{for } i = 1 \text{ to } n & \\
\text{pick } c_i \in \{0,1\}, \quad \text{start } \mathsf{timer}_i & \xrightarrow{c_i} & r_i = a_{2i+c_i-1} \\
\text{stop } \mathsf{timer}_i & \xleftarrow{r_i} & \\
 & \textbf{verification phase} & \\
a = s \oplus m, & & \\
\mathsf{check\,timer}_i \leq 2B, r_i = a_{2i+c_i-1} & \xrightarrow{\mathsf{Out}_V} & \\
\text{output } \mathsf{pk}_P & &
\end{array}
$$

Figure C.1 – An example DB protocol in PM which is DH-secure but not MiM-secure

can authenticate itself and let the close and honest prover respond to the challenge(s). So, V decides that the malicious prover is close. It is not MiM-secure, because a MiM-adversary responds the challenge(s) itself in the challenge phase and, in the rest, he relays the messages between the (far-away) prover and the verifier.

MiM $\nrightarrow$ DF and MiM $\nrightarrow$ DH: MiM-symDB in Chapter 5 (Figure 5.4) is MiM-secure but not DF-secure. So, it is not DH-secure either.

DH $\nrightarrow$ MiM: We show a DB protocol which is DH secure in PM but not MiM-secure in PM in Figure C.1 below.

MiM $\nrightarrow$ TF and DH $\nrightarrow$ TF: Eff-pkDB [KV16], PrivDB [Vau15c] are MiM and DH secure. However, they are not TF-secure (they are not even TF'-secure in PM).

DH-secure but not MiM-secure DB protocol: We modify the Eff-pkDB protocol [KV16] as in Figure C.1. We let the prover send the secret a before challenge phase. The rest is the same.

This protocol is still DH-secure because of the following proof sketch: Eff-pkDB is DH secure as shown in [KV16]. In the proof, it has been shown that the secret s_1 generated by P' (close and honest) is independent from s generated by malicious P. It shows that P does not know anything about s_1 just after it is generated before a_1 is released. When we consider modified Eff-pkDB, we observe that even if P sends m himself, a that P has and a_1 that P' has will be independent. Therefore, with the same arguments of DH-proof of Eff-pkDB, we can show that modified Eff-pkDB is DH secure.

On the other hand, it is not MiM secure because an adversary knows the responses beforehand. When we convert it to SHM where P behaves as P_{dum}, we have the same security properties as well.

Appendix D

Full EMV

Here are some abbreviations used in the protocol:

- AID: Application Identifier. It defines each new execution of EMV protocol.
- AFL: Application File Locator. It is a list of files used in transaction. Application Expiry Date, PAN, CDOL1, CDOL2 are mandatory files.
- AIP: Application Interchange Profile. It indicates the supported features of the card (i.e., terminal risk management and authentication methods: SDA, DDA, CDA, card holder verification, issuer authentication, relay resistance protocol)
- ATC: Application Transaction Counter
- CDA: Combined Application Data. It is an authentication method combined in transaction phase.
- CDOL: Card Risk Management Object List. CDOL1 and CDOL2 lists the objects that the card needs to generate the first cryptogram and the second cryptogram, respectively.
- CidD: Cryptogram Information Data.
- DDA: Dynamic Application Data. It is a sign of dynamic data.
- SDA: Static Application Data. It is a sign of static data.
- PDOL: Processing Data Object List. It specify which information the card wants from the terminal.(e.g. terminal country code (TCC), amount (τ))

  EMV contactless session consists of four phases without card holder (user) verification method (i.e., Online PIN, Signature):

  - *Contact Establishment with NFC card:* $\boldsymbol{T}$ detects $\boldsymbol{C}$.

- *Transaction Initialization:* C picks an application identifier AID and sends it to T. Then, they exchange their data to continue to the next phase. C asks for some information of the terminal (PDOL). Then, T sends its PDOL data with a command (GET-PROCESSING-OPTIONS). C responds with its all supported features (AIP) and the card information needed for the transaction (AFL).
- *Transaction Initialization:* C picks an application identifier AID and sends it to T. Then, they exchange their data to continue to the next phase. C asks for some information of the terminal (PDOL). Then, T sends its PDOL data with a command (GET-PROCESSING-OPTIONS). C responds with its all supported features (AIP) and the card information needed for the transaction (AFL).
- *Relay Resistance Protocol [emvb]:* This protocol is executed if the card and the terminal supports it. Here, we assume that they support this feature. The terminal picks a random number R_1 and sends this to the card with a command EXCHANGE-RELAY-RESISTANCE-DATA. The card responses with another random number R_2. It also sends timing estimates ($timings$): Min Time For Processing Relay Resistance Protocol, Max Time For Processing Relay Resistance Protocol, Device Estimated Transmission Time For Relay Resistance Protocol. Then, the terminal checks if the total time passed after sending R_1 exceeds the limit. If the limit does not exceeds, then the next phase begins. Otherwise, the transaction is canceled.
- *Data Authentication:* There are three type of authentication methods in EMV: Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined Data Authentication (CDA). Because of some weaknesses in SDA and DDA (replay attacks and wedge attacks), in this paper, we consider CDA which is combined with the next phase.
- *Transaction:* T sends a command (GENERATE-AC) to C for the application cryptogram. This command includes: the type of cryptogram T requires, a CDA request, a list of values in CDOL1 that C needs from T to compute AC and a random number UN_T picked by T.

  In EMV, there are three type of cryptograms: Transaction Certificate (TC), Authorization Request Cryptogram (ARQC), Application Authentication Cryptogram (AAC). Here, we consider the online verification where T requests ARQC for online verification by the issuer. TC is used for the offline verification by the issuer and AAC is used to cancel the transaction.

  * *Online Verification:* C increases its counter ATC and generates a secret key SK_{AC} by using the counter and the master secret key MK_{AC}. Then, it generates the cryptogram: a MAC of $UN_T, AIP, ATC, \tau, TCC$ and some details of the transaction (See [emvc], Table 26) with using the secret key SK_{AC}. C sends the cryptogram ARQC to T and T relays it to I along

with card information. I verifies the MAC and possibly validate the information of C. If the cryptogram passes verification, then I generates a MAC of ARQC and ARC with the secret key SK_{AC}. This MAC is called as ARPC. After, it sends ARPC to T and T relays it to C with the second GENERATE-AC command for the generation of TC if ARC is true. Otherwise, it sends GENERATE-AC command for the generation of AAC to cancel the transaction.

C verifies ARPC. If the verification and ARC is true then C generates the second cryptogram which is TC. TC is a MAC of CDOL2's objects with SK_{AC} (See [emvc], Table 26)[1] in order to show transaction is complete. Additionally, it generates a signature of unpredictable numbers, ATC, TC,$timings$,R_1,R_2 and CDOL-PDOL data. C signs it with S_{IC}.

∗ Terminal checks if the sign and the data signed are valid. Later, the terminal contacts with the issuer to receive to reimbursement. In this case, the issuer verifies the signature and TC to execute the reimbursement.

[1] Even if CDOL1 and CDOL2 list the same objects, some terminal related objects change because the payment process continues (e.g., TVR) [Rad03].

$I(S_I, P_I, MK_{AC})$		T		$C(P_{IC}, S_{IC}, P_I, MK_{AC})$
			Initialization	
			$\xrightarrow{\text{SELECT}(AID)}$	$AID \leftarrow aid$
			$\xleftarrow{AID,[PDOL]}$	do a list $PDOL$
		retrieve data in $PDOL$ (i.e., TCC, τ)	$\xrightarrow{\text{GET-PROCESSING-OPTIONS}(TCC,\tau)}$	$AIP \leftarrow CDA$,
			$\xleftarrow{AIP,AFL}$	$AFL \leftarrow \{PAN, ED, CDOL1, CDOL2\}$
			Relay Resistance protocol	
		pick R_1	$\xrightarrow{\text{EXCHANGE-RELAY-RESISTANCE}(R_1)}$	pick R_2
		check RTT	$\xleftarrow{R_2,timings}$	
			Data Authentication and Transaction	
		pick UN		
		$lst_T = \{req_{ARQC}, req_{CDA}, UN_T\}$	$\xrightarrow{\text{GENERATE-AC}(lst_T)}$	$ATC \leftarrow ATC{+}{+}$
				$SK_{AC} \leftarrow Gen(MK, ATC)$
if balance and AC ok	$\xleftarrow{Cid,ARQC}$	$Cid = \{ED, PAN\}$	$\xleftarrow{ARQC}$	$ARQC = \mathsf{MAC}_{SK_{AC}}(UN, AIP, ATC, \tau, TCC)$
$MK_{AC} \leftarrow SK_{AC}$ $X = ARC \| 6 \times$ "00" $Y = AC \oplus X$				
$ARPC = \mathsf{DES3}_{SK_AC}(Y)$	$\xrightarrow{ARPC,ARC}$	If $ARC = True$		
		$lst2_T = \{req_{TC}, req_{CDOL2}, ARPC\}$	$\xrightarrow{\text{GEN-AC}(lst2_T)}$	$b \leftarrow \texttt{Verify}(ARPC)$
				if $b = True$ $TC = \mathsf{MAC}_{SK_{AC}}()$
		if $V = true$	$\xleftarrow{V}$ 2874	
			$\xrightarrow{\text{GEN-AC}(TC, CDA, \text{data in } CDOL2)}$	$TDHC = H(PDOL, CDOL1, CDOL2\text{data}, CidD, ATC, AC)$
		Verify $SDAD$	$\xleftarrow{CidD,ATC,SDAD}$	$SDAD = \mathsf{sign}_{S_{IC}}(UN_C, CidD, AC, TDHC, UN_T)$

Figure D.1 – The Full EMV protocol

Appendix E

Curriculum Vitae

Name:	Handan Kılınç Alper
E-mail:	handankilinc1@gmail.com
Citizenship:	Turkish

Education

2014 - 2018 **PhD in Computer, Communication and Information Sciences**
Area: Cryptography
Supervised by Prof. Serge Vaudenay
Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

2012 - 2014 **Master in Computer Science and Engineering**
Area: Cryptography
Supervised by Asst Prof. Alptekin Küpçü
Koç University, Turkey

2006 - 2011 **Bachelor in Mathematics**
TOBB University of Economics and Technology (TOBB ETU), Turkey

2006 - 2012 **Bachelor in Computer Science (Double Major)**

	TOBB University of Economics and Technology (TOBB ETU), Turkey
March 2012 - July 2012	**Erasmus student** AGH University of Science and Technology, Poland

Experience

Apr 2018	**Invited Talk at FutureDB workshop**
Apr 2018	**Invited Talk at Training School on Cryptanalysis of Ubiquitous Computing Systems**
Apr 2016	**Invited talk at CryptoAction Symposium**
2014 - 2018	**Teaching Assitant at EPFL** (courses: 'Cryptography and Security', 'Student Seminar: Security Protocols and Applications', 'Mathematical Analyzes')
2015 - 2018	**Supervised 5 Master and Bachelor semester projects and co-supervised one Master thesis**
2012 - 2014	**Research Assitant at Koç University** (research on achieving fairness in multi/two party computation resulted with 2 conference papers and 1 journal submission)
2012 - 2014	**Teaching Assitant at Koç University** (courses: 'Discrete Mathematics and Applications', 'Algorithms', 'Empirical and Quantitative Reasoning')
Sep 2011 - Feb 2012	**Part time software developer in SimTek Simulation and Information Technologies** (project: designing simulation of an educational device set in the military. Programming Language: Java and C++)

Academic Honors

2014	**One year fellowship from EDIC (Computer and Communication Sciences in EPFL) for the doctoral study.** EPFL, Switzerland
2012 - 2014	**Full scholarship from Koç University for the master study.** Koç University, Turkey
2011	**Ranked 2^{nd} in GPA among the class of 2011 Mathematics majors.** TOBB ETU, Turkey
2006 - 2012	**Full scholarship from TOBB ETU for the bachelor study.** TOBB ETU, Turkey

Projects

2016 - 2018 **SNF project on 'The Implication of Time and Position in Cryptography'**

2014 - 2018 **ICT COST Action IC1403 Cryptacus in the EU Framework Horizon 2020**

2012 - 2014 **Fair and Secure Computation**

Languages

Turkish (native), *English* (fluent), *French* (basic), *Spanish* (beginner)

Technical Skills

- **Programming Languages:** Python, C++, Java
- **Development Environments:** Eclipse, Visual Studio, MATLAB, Linux, MySQL

Extracurricular Activities

In TURQUIA 1912 – Turkish Students Association in Switzerland:

- President (2017-2018), Secretary (2016-2017), Vise President (2015-2016)
- Conducted meetings to introduce the association to new people and attracted members.
- Organized and found sponsors for the traditional annual reception each year from 2016 to 2018 for about 70-100 attendants including Turkish representatives in Switzerland.