PhD-FSTC-2019-33
The Faculty of Sciences, Technology and Communication

DISSERTATION

Defence held on 14/05/2019 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

José Miguel LÓPEZ BECERRA
Born on 28 September 1984 in Nezahualcoyotl (Mexico)

PROVABLE SECURITY ANALYSIS FOR THE PASSWORD AUTHENTICATED KEY EXCHANGE PROBLEM

Dissertation defence committee

Dr Peter Y. A. Ryan, dissertation supervisor
Professor, Université du Luxembourg

Dr. Jean-Sébastien Coron, Chairman
Professor, Université du Luxembourg

Dr. Dimiter Ostrev, Vice Chairman
Research Associate, Université du Luxembourg

Dr. Michel Abdalla
Professor, École normale supérieure

Dr. Steve Kremer
Inria senior research scientist
Université de Lorraine

Abstract

Password-based Authenticated Key-Exchange (PAKE) protocols allow the establishment of *secure communications* despite a human-memorable password being the only secret that is previously shared between the participants. After more than 25 years since the initial proposal, the PAKE problem remains an active area of research, probably due to the vast amount of passwords deployed on the internet as password-based still constitutes the most extensively used method for user authentication. In this thesis, we consider the computational complexity approach to improve the current understanding of the security provided by previously proposed PAKE protocols and their corresponding security models. We expect that this work contributes to the standardization, adoption and more efficient implementation of the considered protocols.

Our first contribution is concerning *forward secrecy* for the SPAKE2 protocol of Abdalla and Pointcheval (CT-RSA 2005). We prove that the SPAKE2 protocol satisfies the so-called notion of *weak forward secrecy*. Furthermore, we demonstrate that the incorporation of key-confirmation codes in the original SPAKE2 results in a protocol that provably satisfies the stronger notion of *perfect forward secrecy*. As forward secrecy is an explicit requirement for cipher suites supported in the TLS handshake, we believe our results fill the gap in the literature and facilitate the adoption of SPAKE2 in the recently approved TLS 1.3.

Our second contribution is regarding *tight security reductions* for EKE-based protocols. We present a security reduction for the PAK protocol instantiated over Gap Diffie-Hellman groups that is tighter than previously known reductions. We discuss the implications of our results for concrete security. Our proof is the first to show that the PAK protocol can provide meaningful security guarantees for values of the parameters typical in today's world.

Finally, we study the relation between two well-known security models for PAKE protocols. Security models for PAKEs aim to capture the desired security properties that such protocols must satisfy when executed in the presence of an adversary. They are usually classified into i) indistinguishability-based (IND-based) or ii) simulation-based (SIM-based), however, controversy

remains within the research community regarding what is the most appropriate security model that better reflects the capabilities that an adversary is supposed to have in real-world scenarios. Furthermore, the relation between these two security notions is unclear and mentioned as a gap in the literature. We prove that SIM-BMP security from Boyko et al. (EUROCRYPT 2000) implies IND-RoR security from Abdalla et al. (PKC 2005) and that IND-RoR security is equivalent to a slightly modified version of SIM-BMP security. We also investigate whether IND-RoR security implies (unmodified) SIM-BMP security.

Acknowledgments

There is a lot of people I am obliged to thank. Firstly, I would like to thank Prof. Peter Ryan for giving me the opportunity to join the Apsia group and pursue my Ph.D. under his supervision. I am truly grateful for his support over the last four years and for giving me the freedom to conduct various research projects without objection. I am obliged to express my gratitude to Dr. Dimiter Ostrev for his truly dedicated supervision, and for the countless meetings and fruitful discussions that inevitably influenced my research work. Dimiter is someone who is always a pleasure to work with. I would also like to thank Dr. Jean Lancrenon for patiently supervising me during the first year of my Ph.D. journey. His meticulous research stimulated my interest in the key-exchange problem. Without the advice and support of Peter, Dimiter and Jean, overcoming the obstacles and completing this thesis would not have been possible. I can summarize my experience by only saying that it has been a great pleasure collaborating with these people.

I express my sincere gratitude to Prof. Michel Abdalla, Prof. Jean-Sbastien Coron and Dr. Steve Kremer for being part of my defense as jury members without hesitation. My special gratitude goes to Steve, who additionally, took part in my CET committee from the very beginning.

I am also deeply thankful to all of those with whom I have had the opportunity to work with. I am grateful to Marjan Skrobot for all the support, advice and worthwhile discussions while working together. My sincere thanks goes to the Apsia team. Over the last four years, I have had the opportunity to meet exceptional people that make of Apsia a challenging, motivating and friendly group to work with.

Finally, I would like to thank my family for the immense support throughout my entire life. My deepest gratitude goes to my mother for always motivating me and supporting my decisions, my father and sister for always being there when needed help and my precious aunt Many her invaluable support when needed it the most. I am profoundly grateful to Antonio, his advice, support and aid over the last seven years is irreplaceable.

During my PhD studies, I was supported by the Luxembourg National Research Fund (FNR) under CORE project AToMS (Project ID 8293135).

Previous Publications

The research leading to this thesis has been previously published and presented in the following conferences:

1. Jose Becerra, Vincenzo Iovino, Dimiter Ostrev and Marjan Skrobot. On the Relation Between SIM and IND-RoR Security Models for PAKEs. 14th International Conference on Security and Cryptography - SECRYPT 2017. In Pierangela Samarati, Mohammad S. Obaidat and Enrique Cabello, editors, *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - SECRYPT. Madrid, Spain, July 2017*, pages 151-162. SciTePress, 2017.
   Available at `https://eprint.iacr.org/2017/470`

2. Jose Becerra, Vincenzo Iovino, Dimiter Ostrev and Marjan Skrobot. On the Relation Between SIM and IND-RoR Security Models for PAKEs with Forward Secrecy. In Mohammad S. Obaidat and Enrique Cabello, editors, *E-Business and Telecommunications - 14th International Joint Conference, ICETE 2017, Madrid, Spain, July 24-26, 2017, Revised Selected Paper*, pages 173-198. Springer, 2019.
   Available at `http://orbilu.uni.lu/handle/10993/38621`

3. Jose Becerra, Vincenzo Iovino, Dimiter Ostrev, Petra Sala and Marjan Skrobot. Tightly-Secure PAK(E). In *Cryptology and Network Security - 16th International Conference, CANS 2017, Hong Kong, China, November 30 - December 2, 2017, Revised Selected Papers*, pages 27-48. Springer, 2018.
   Available at `https://eprint.iacr.org/2017/1045`

4. Jose Becerra, Dimiter Ostrev and Marjan Skrobot. Forward Secrecy of SPAKE2. In *Provable Security - 12th International Conference, ProvSec 2018, Jeju, South Korea, October 25-28, 2018, Proceedings*, pages 366-384. Springer, 2018.
   Available at `https://eprint.iacr.org/2019/351`

Additionally, during his Ph.D. studies, the author collaborated in the following publications, however, the content of this thesis is not based on such publications.

1. Jose Becerra, Peter B. Rønne, Peter Y.A. Ryan, Petra Sala. HoneyPAKEs. In Vashek Matyás, Petr Svenda, Frank Stajano, Bruce Christianson, and Jonathan Anderson, editors, Security Protocols XXVI - 26th International Workshop, Cambridge, UK, March 19-21, 2018, Revised Selected Papers, volume 11286 of Lecture Notes in Computer Science, pages 63-77. Springer, 2018.
   Available at `http://orbilu.uni.lu/handle/10993/37937`

2. Jose Becerra, Peter Y.A. Ryan, Petra Sala and Marjan Skrobot. An Offline Dictionary Attack against zkPAKE Protocol. *To appear in* the 34th International Conference on Information Security and Privacy Protection - IFIP SEC 2019.

Contents

List of Figures

List of Abbreviations

AKE	Authenticated Key-Exchange
CA	Certification Authority
CDH	Computational Diffie-Hellman
CRS	Common Reference String
DDH	Decisional Diffie-Hellman
EKE	Encrypted Key Exchange
FtG	Find-then-Guess
I-D	Internet Draft
IETF	Internet Engineering Task Force
IND	Indistinguishability
IoT	Internet of Things
PAK	Password Authenticated Key exchange
PAKE	Password Authenticated Key-Exchange
PFS	Perfect Forward Secrecy
PIN	Personal Identification Number
PKI	Public-key Infrastructure
PPT	Probabilistic Polynomial Time

RoR	Real or Random
SIM	Simulation
SPAKE2	Simple Password-Based Encrypted Key Exchange
SRP	Secure Remote Password
TLS	Transport Layer Security
UC	Universally Composable
wFS	Weak Forward Secrecy

CHAPTER 1

Introduction

1.1 The Evolution of Cryptography

The most fundamental goal of cryptography is establishing *secure* communications in the presence of an adversary. From history, we have learned that the design of cryptographic protocols is far from being a trivial task: a large number of cryptographic mechanisms were proposed, believed to be secure and then broken. To break this vicious cycle, the cryptographic community developed a methodology to gain confidence in the security of a given protocol by performing a rigorous analysis. This approach is commonly referred to as *provable security* or *reductionist approach*.

1.1.1 Ancient Cryptography

The desire of communicating securely is believed to be as old as the invention of the writing: We know from history that cryptography started 4000 years ago, when ancient civilizations including Egypt, China, Mesopotamia, developed mechanisms to protect the *privacy* of messages exchanged so that only the *legitimate* recipient could read them – even if the message was intercepted by the enemy. Examples of ciphers in the realm of ancient cryptography include Caesar and Vigenère schemes and the well known Enigma machine.

Originally, cryptography was a discipline reserved exclusively for military and diplomatic purposes. It was during the early 20th century when the development of cryptography received a tremendous boost due to the invention and widespread of electronic communications, i.e. the telegraph and the radio, as they are inexpensive and broadcast communication channels where it is relatively simple to intercept the messages being transmitted [Riv, Kah96].

It is beyond doubt the influence that cryptography has had in shaping the history of humanity. A noticeable example is the Second World War (WW2): it is estimated that cryptanalysis lead

by Alan Turing on the Enigma machine – the enciphering machine used by the German armed forces – shortened the WW2 at least by two years saving countless lives.

The previously mentioned ciphers correspond to the so-called *ancient cryptographic methods*. During this era, cryptography was merely an art: constructing ciphers and breaking them relied on personal skills and creativity. Furthermore, a cipher was considered secure as long as no attacks were found on it. This approach led to the proposal of a countless number of ciphers which subsequently were found to be flawed. A remarkable example is the Vigenere cipher, which was considered unbreakable for 300 years until Friedrich Kasiski published an attack in 1863. Similarly, the german army considered the Enigma machine unbreakable. Moreover, with exception of the Enigma machine, the ciphers of this era were based on *security through obscurity*, as it was believed that keeping the mechanisms for encrypting and decrypting secret from the adversary would increase its security.

1.1.2 Modern Cryptography: A Computational Complexity Approach

We know from history that ad-hoc approach followed in ancient cryptographic methods is not convincingly successful – every cipher corresponding to that realm of ancient cryptography can be easily broken with modern technology or even by hand. The need of stronger security guarantees motivated the transition to *modern cryptography*: cryptography evolved from being the art of writing secret codes to a scientific discipline with fundamentals in mathematics, information theory and computational-complexity theory.

Modern cryptography considers the dimension of computational resources available to the adversary. Thus, a cryptographic protocol is secure if it is *infeasible* to break it for a computationally bounded adversary, i.e. it follows a computational complexity approach. This approach was firstly suggested by Shannon in his work entitled "Communication Theory of Secrecy Systems", for many considered the father of modern cryptography, as his work lead the transition of cryptography to a scientific discipline.

It is also remarkable that modern cryptography expanded the applications of cryptography: it transitioned from a discipline used only to protect the secrecy of military communication, to one that incorporates broader security requirements – authentication, message integrity, non-repudiation – with applications ranging from e-commerce, e-banking and instant messaging protocols to industrial secrets and even national security.

1.1.3 Provable Security

The development of cryptography has been largely influenced by the *provable security* approach due to Goldwasser and Micali [GM84a], which generally speaking, is about the security that can be proven.[1] To demonstrate that a protocol is *provably secure* the following is needed:

[1] In fact, Katz and Lindell [KL07] consider that modern cryptography is characterized by its emphasis on precise definitions, rigorous proofs of security and the fact that needed assumptions are clearly stated, i.e. they consider that modern cryptography is about the security that can be proved.

1. Define the adversarial capabilities. The only assumption one can make on the adversary is about its computing capabilities, particularly, no assumption regarding the adversarial strategy should be made [Gol06, Chapter 1].[2]
2. Define what it means for the protocol to be secure, or put differently, which adversarial actions constitute a break on the protocol.
3. Finally, prove that protocol in question satisfies the given security definition – this is usually done via the reductionist approach. This way, one shows that the only way to break the protocol in question is by solving a problem assumed to be computationally hard.

Provable security is the approach that we followed in this work to gain confidence in the security of the studied cryptographic protocols. Obviously, when considering this approach, the hope is that the security definition is *meaningful* and that the underlying hardness assumption holds, i.e. one should only rely on well studied hardness assumptions.

1.2 Session Key Generation Protocols

1.2.1 Motivation

The adoption of the internet has revolutionized the way we work, socialize and do business. Nowadays, the world faces an impressive demand for internet-based solutions in the manner of e-banking, information sharing, the Internet of Things (IoT), etc., whose demand is only expected to grow over time. Nevertheless, together with the need to transmit data over the internet comes the responsibility of protecting it. Then it is fundamental to guarantee the *secrecy*, *authenticity* and *integrity* of online communications, where cryptography is at the core of the solution.

Encryption is the process of encoding information in such a way that only *authorized* parties can read it. This authorization is done through the knowledge of a *secret key*, in such a way that only those entities holding the decryption key can read the content of an encrypted message. However, encryption by itself is useless without proper *key distribution*, i.e. an attacker who manages to obtain the decryption key could read the entire communication. Session key generation protocols provide an answer to the fundamental problem of key distribution.

A session key generation protocol is a cryptographic building block where two parties exchange messages to jointly compute a high-entropy *session key* over an insecure network – the goal is to use the established session key to build secure communications. Let us consider the following scenario for the sake of clarity: suppose Bob uses a mobile banking application on a regular basis to pay his bills. In an attempt to communicate securely with his Bank, a *naive* Bob locally generates a session-key and sends it to the Bank through a public network. The problem with this approach is that an adversary could easily capture the session key while it is in transit. A

[2] Computing capabilities usually refer to computational time and space. However, in the context of (Password) Authenticated Key-Exchange protocols, it is also important to define the attack interfaces that the adversary has, for instance, she can manipulate the network traffic, she can learn the established session keys or she may corrupt users.

better approach is when Bob and his Bank jointly execute a session key generation protocol, which allows them to agree on a shared strong cryptographic key *per-session* only known to the two of them.

Finally, we would like to highlight that session key generation protocols play a fundamental role in today's communication. Probably the most well-known example is the TLS standard, which stands as the most widely used cryptographic protocol – it is estimated that over 80% of the internet traffic is TLS-encrypted [moz].

1.2.2 Set-up Assumptions

To authenticate legitimate parties, session key generation protocols assume that some initial configuration is available, otherwise, there is always the chance of an adversary masquerading as an honest user without being detected. These set-up assumptions come in the following fashion:

- **Something the user is.** This corresponds to bio-metric authentication, such as fingerprint or facial recognition.
- **Something the user has.** This corresponds to Authenticated Key-Exchange (AKE) protocols, where the users hold a high-entropy long-term key – either symmetric or public-key. The long-term key is usually stored in an electronic device, such as a smart card or a token.
- **Something the user knows.** This is a human memorable secret, such as a password or a PIN code. This category corresponds to Password Authenticated Key-Exchange (PAKE) protocols and is the topic in this thesis.

1.3 An Overview of PAKEs

Password Authenticated Key Exchange protocols (PAKEs) are cryptographic building blocks to allow two users, who already share a password, to agree on a *cryptographically strong* session key by exchanging messages over a hostile network. The fundamental requirements that these protocols must satisfy are the following:

1. Secrecy of the session key: The established session key should be known only to legitimate parties who participated in the protocol.
2. Entity Authentication: When the protocol finishes, legitimate parties should know who they – presumably – share the session key with. This authentication could either be *explicit* or *implicit* (we refer to Section 3.3 for in depth discussion).
3. Offiline dictionary attacks resistance: The execution of the protocol should not leake any information that could allow an adversary to verify password guesses in an *offline* manner.

The first PAKE protocol is due to Bellovin and Merrit in 1992 [BM92], since then, it has been an active research topic within the cryptographic community, resulting in the proposal of

numerous PAKEs with enhanced efficiency and security properties. The development of PAKEs was motivated by the need to establish secure communications – authenticated and encrypted – in the setting where the participants previously share a password, for instance most of today's web authentication.

1.3.1 Security considerations for PAKEs

The design of PAKE protocols is a delicate task. Over the years, we have witnessed the repetition of the same phenomenon: PAKEs were proposed, believed to be "secure" and broken later on. The protocols which survive over the years are most frequently those which provide enough evidence to satisfy a sound *security definition*. This approach permits to gain confidence regarding the security guarantees offered by the protocol in question, however, it also opens the question of what constitutes a "good" security definition.

A security model for PAKE protocols is a framework which allows us to define what it means for a protocol to be secure. It first needs to define the setting in which a given protocol is to be executed, for instance a PAKE is a 2-party protocol, executed in the multi-user setting and with multiple user instances running simultaneously. It also needs to specify what the adversarial capabilities are: certainly, the model should not assume anything about the adversarial strategy, however, it can still describe what she is capable to fulfill, for instance she may be simply an eavesdropper or she may actively interfere with the communication. Finally, the security model needs to define what constitutes a break in the protocol.

The ultimate goal of a security model is to provide a *meaningful* definition of security by considering *reasonable* adversarial capabilities. Nevertheless, the design of such models is far from being a trivial task. Over the last two decades we have witnessed the evolution of security models for PAKEs: it started in the year 2000 with the Find-then-Guess (FtG) model of Beralle et al. [BPR00] and in parallel, the simulation-based model of Boyko et al. [BMP00], followed by the Real-or-Random (RoR) model [AFP05], the Universally Composable (UC) extended to the PAKE setting [CHK$^+$05] and more recently, the strong augmented PAKE functionality of Jarecki et al. [JKX18]. The basis of all the aforementioned security models is to capture resistance to *offline dictionary attacks* under an adversary-controlled network, however, there are important differences between them that result in different notions of security being modeled.

The design of security models is a continuous process. As the research community gains a clearer understanding of security in the PAKE setting, new security models are proposed to incorporate the new requirements. For example, the following security properties were not considered in initial works:

- Forward secrecy: This property asks for the security of sessions which are established before compromise of the user password. It went from being an optional property to an explicit requirement. For instance, the newly accepted TLS 1.3 standard removed support for cipher suites that do not satisfy this property (see Chapter 5).

- Composition with symmetric-key encryption schemes: In practice, PAKEs are commonly executed together symmetric-key schemes. The idea is to run a PAKE to establish a session key and then use it to symmetrically encrypt the communication between the involved parties. Thus, PAKEs need to compose securely with such encryption schemes [SL18].
- Pre-computation attacks resistance [JKX18]: Whenever the password file gets exposed due to a security breach, the best one can expect is that it does not *inmediately* allows the adversary to masquerade as a client. Put differently, in order to break a single password, the adversary must perform an exhaustive search linear in the size of the dictionary and this computation must occur only after the leakage of the password file.

1.4 Motivation and Research Objectives

Password-based authentication, despite its well-known vulnerabilities [AS99, Bon12], constitutes still the most commonly used method for user authentication over the internet – probably because it is simple to deploy while still providing an intuitive user experience. Therefore, passwords are extensively used in *login* scenarios and the most common implementation is the so-called *password-over-TLS* approach, where a client C authenticates to server S by sending its username and password to the server. The communication is done over a TLS channel to protect the password while it is in transit, however, this approach is known to be vulnerable to phishing attacks [CHVV03, EKSS09]. Regardless of how the passwords are stored at the server – *cleartext*, *hashed* or *hashed and salted* – the previously described method of authentication requires the password to be transmitted from the client to the server, which could potentially result in compromise of the password. Fortunately, one can achieve with PAKEs the same login functionality of authentication, while intrinsically protecting the user password as it is never exposed during the protocol execution.

Given the large scale of passwords deployed over the internet, together with the increasing threat of phishing attacks, one could expect PAKEs to be widely deployed for website authentication and benefit from enhanced security guarantees compared to the traditional approach. Unfortunately, this is not the case. Most researchers and practitioners attribute this phenomenon to patent rights: Lucent Technologies patented the *method* described in EKE, [BM92, BM93],[3] while SPEKE was patented by Phoenix Technologies [Jab96]. Luckily for PAKEs, the patents owned by Lucent Technologies expired in 2011 and in 2013, for the EKE protocol and its augmented variant respectively, while the patent rights on SPEKE2 expired in 2017. We believe that the patent expiration should motivate the deployment of PAKEs, considering that these protocols fit amazingly in internet-based applications such as the Internet of Things (IoT), e-banking, e-commerce, where password-based authentication mechanism is common and the establishment

[3]Moreover, the patents owned by Lucent Technologies seemed to also cover subsequently deployed protocols including AuthA, PPK, PAK and SPAKE2 [BR00, Mac02a, AP05b], as they are variants of the original EKE protocol but considering different instantiations of the encryption function. In fact, one of the motivations in J-PAKE [HR10] was to circumvent the patent issue.

of secure communication channels is required.

We consider the computational-complexity approach to analyze the security of the PAKE protocols considered in this thesis. Our aim is to facilitate the adoption of PAKEs in real-world applications. Particularly, our the research objectives are:

1. Investigate whether the SPAKE2 protocol [AP05b] provably satisfies some meaningful notion of forward secrecy. This is a necessary condition for its adoption as cipher suite in the recently approved TLS 1.3 standard.
2. Investigate the notion of tight security reductions for PAKE protocols and its relevance for *concrete security*. In particular, we consider the instantiation of the PAK protocol [Mac02a] over Gap Diffie Hellman groups.
3. Examine whether the simulation-based [BMP00] and indistinguishability-based [AFP05] security notions for PAKEs are equivalent.

More concretely, when a protocol is to be used in real-world applications, it is fundamental to consider i) its computational efficiency and ii) whether it satisfies a meaningful notion of security. That being said, we believe that the previously mentioned objectives impacts positively on the deployment of PAKEs in real-world applications:

- In our first result, we prove that the SPAKE2 protocol satisfies a meaningful notion of forward secrecy. We believe this result fills the gap in the literature and facilitates the adoption of the protocol in the TLS 1.3 standard as pre-shared key mechanism.
- In our second result, we provide a security reduction for the PAK protocol that is tighter than previously known ones. Tight reductions are relevant when considering the *concrete security* provided by the protocol in question. The reason is that non-tight reductions induce a *security degradation* factor, which in practice has to be compensated by instantiating the protocol with larger security parameters, resulting in less efficient implementations.
- For our final result, we examine the well-known SIM-based model [BMP00] and IND-RoR model [AFP05]. Despite its technical differences, it seems that both models capture a reasonable notion of security. However, a protocol designer intending to construct a security proof for a new PAKE protocol, has to choose which model is more appropriate. Then, establishing how these two security notions relate to one another, should guide the protocol designer on how to make the choice.

1.5 Outline

The outline of this thesis the following:

Chapter 2: Preliminaries.
In this chapter we provide the computational complexity theoretic and cryptographic preliminaries to make this thesis self-contained.

Chapter 3: Password Authenticated Key-Exchange Protocols.
In this chapter, we provide an introduction to the PAKE problem followed by a discussion on the strengths and weaknesses when passwords are used as authentication mechanisms.

Chapter 4: Security Models for Password Authenticated Key-Exchange Protocols.
In this chapter, we recall the widely known Find-then-Guess security model due to Bellare, Pointcheval and Rogaway [BPR00]. This is the security model that we consider to construct the security proofs for the PAK, SPAKE2 and PFS-SPAKE2 protocols considered in this work.

Chapter 5: Forward Secrecy for SPAKE2.
We consider the adoption of PAKE protocols in the recently approved TLS 1.3 standard. In this TLS version, it is a requirement to provide *forward secrecy* for the established session keys.[4] The IETF working group has recently considered the deployment of the SPAKE2 protocol as pre-shared key mechanism in the TLS 1.3 standard, however, while the SPAKE2 protocol comes with a security proof in the well-known Find-then-Guess model [BPR00], the security proof does not consider any notion of forward secrecy. Our first contribution is to prove that the original SPAKE2 protocol satisfies the so-called notion of weak forward secrecy. Additionally, we demonstrate that when the original SPAKE2 protocol is enhanced with explicit authentication, the resulting protocol provably satisfies the stronger notion of perfect forward secrecy.
The contents of this chapter have appeared in [BOS18].

Chapter 6: Tightly-Secure PAK(E).
We consider the PAK protocol [Mac02a] and present a tight security reduction when the protocol is instantiated over Gap Diffie-Hellman groups. This has direct implications on the *concrete security* provided by the protocol: Non-tight reductions induce a *security degradation* which must be compensated by choosing a larger security parameter when the protocol is to be instantiated – innevitably resulting in less efficient implementations.
The contents of this chapter have appeared in [BIO+18].

Chapter 7: On the Relation between SIM and IND-based models.
Security models for PAKEs are usually classified into i) indistinguishability-based (IND-based) or ii) simulation-based (SIM-based). However, the relation between these two security notions is unclear and mentioned as a gap in the literature. We prove that SIM-BMP security [BMP00] implies IND-RoR security [AFP05] and that IND-RoR security is equivalent to a slightly modified version of SIM-BMP security. We also investigate whether IND-RoR security implies (unmodified) SIM-BMP security.
The contents of this chapter have appeared in [BIOv17, BIOS19].

Chapter 8: Concluding Remarks and Future Directions.
We conclude our work and suggest open problems for future plans.

[4] Forward secrecy guarantees that session keys derived from long-term keys – a password in this thesis – remain secret to the adversary even if the long-term key material gets later compromised. It is a highly desired security property particularly in PAKEs, considering that compromise of the password is unfortunately becoming more and more common due to security breaches.

CHAPTER 2

Preliminaries

2.1 Introduction

When evaluating the security of a protocol in the computational model, the typical approach is to demonstrate that breaking the protocol in question also solves some mathematical problem believed to be hard. That mathematical problem is commonly referred to as *computational hardness assumption*. In this chapter, we recall the necessary background concepts and definitions that make this thesis self-content, taken from different sources including [KL07, Gol06, Gol08, KKS13, BR05, Bai07, Poi05]. Particularly, we describe the mathematical definitions, cryptographic primitives and computational hardness assumptions required to formally describe the protocols and security analysis presented in subsequent chapters.

2.2 Mathematical Background

Next, we introduce the notation that we use throughout this thesis.

Notation. Let A, B be two sets. We write $A \cap B$ to denote their intersection, $\bar{A}$ for the compliment of A and $\Pr[A]$ to denote the probability that A occurs. We write $d \xleftarrow{\$} D$ for sampling uniformly at random from set D and $|D|$ to denote its cardinality. The output of a probabilistic algorithm $\mathcal{A}$ on input x is denoted by $y \leftarrow \mathcal{A}(x)$, while $y := F(x)$ denotes a deterministic assignment of the value $F(x)$ to the variable y. Let $\{0,1\}^*$ denote the bit string of arbitrary length while $\{0,1\}^l$ stands for those of length l. When we sample elements from $\mathbb{Z}_q$, it is understood that they are viewed as integers in $[0 \ldots q-1]$, and all operations on these are performed $\mod q$. Finally, let κ be the security parameter, $negl(\kappa)$ denote a negligible function and $\pi[A, B]$ denote the password shared between principals A and B.

Definition 2.1. *A group $\langle \mathbb{G}, * \rangle$ is a set $\mathbb{G}$ equipped with a binary operation $*$, such that the following properties are satisfied:*

1. *Closure. $\forall a, b \in \mathbb{G}$, then it is the case that $a * b \in \mathbb{G}$.*
2. *Associativity. $\forall a, b, c \in \mathbb{G}$, we have $(a * b) * c = a * (b * c)$.*
3. *Identity. There exists an element $e \in \mathbb{G}$ which is neutral with respect to the operation, i.e. $\forall a \in \mathbb{G}$ we have $a * e = e * a = a$.*
4. *Invertibility. $\forall a \in \mathbb{G}$ there exists $\tilde{a} \in \mathbb{G}$ such that $a * \tilde{a} = \tilde{a} * a = e$.*

Definition 2.2. *(Cyclic Group). A group is cyclic with generator $g \in \mathbb{G}$ if and only if the set of elements comprising $\mathbb{G}$ is the set:*

$$\{g^a : a \in \mathbb{Z}\}$$

Definition 2.3. *A cryptographic hash function $H : \{0,1\}^* \to \{0,1\}^n$ is a deterministic function satisfying the following requirements in the presence of a computationally bounded adversary:*

1. *Pre-image resistance: Given $y = H(m)$, output m.*
2. *Collision resistant: Find two messages m_1, m_2 s.t. $H(m_1) = H(m_2)$.*
3. *Second pre-image resistant: Given m_1, find m_2 s.t. $H(m_1) = H(m_2)$.*

Lemma 2.4. *(Birthday Paradox Bound). Fix a positive integer N and sample q elements $y_1, ..., y_q$ uniformly and independently at random from a set of size N. Then the probability that there exists different i, j such that $y_i = y_j$, i.e. at least one collision occurs, is upper bounded by:*

$$coll(q, N) \leq \frac{q^2}{2N}$$

In cryptography, it is a good approach to define the adversary's advantage $\epsilon(\cdot)$ on breaking the protocol as a function of the security parameter κ. At the same time, we want this advantage to be too small to matter. More formally, we require $\epsilon(\kappa)$ to grow smaller than the inverse of any polynomial, i.e. to be a negligible function of the security parameter. The underlying idea is that an event that occurs with negligible probability, would be very unlikely to occur even when the experiment is repeated a polynomial number of times.

Definition 2.5. *A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for every constant $c \geq 0$ there exists an integer N such that for every $n > N$ it holds that $\epsilon(n) < n^{-c}$.*

Definition 2.6. *(Asymptotic Notation). Let $f(n)$ and $g(n)$ be functions that map from positive integers to positive reals. Then $f(n) = \mathcal{O}(g(n))$ means that there exists positive integers c and n' such that for all $n > n'$ it holds that $f(n) \leq c \cdot g(n)$.*

2.3 Complexity-based Cryptography

The foundations of modern cryptography rely on computational complexity theory. In this section, we provide the basic notions of the latter that we consider essential for the completeness of this thesis.

Computational complexity is the study of the resources needed by a machine to solve computational problems. The considered resources are often time and memory. Generally speaking, it aims to distinguish between those problems which are *easy* to compute from the *hard* ones. Then, as modern cryptography is based on the gap between efficient algorithms given to legitimate users and the computational infeasibility of breaking them, computational complexity provides a foundation for the study of modern cryptography.

Running Time. This is the most studied resource in computational complexity. In this context, the running time of an algorithm is a function of the length of its *input*, that indicates the number of elementary operations – or bit operations – that have to be performed for each run of the algorithm. Given an algorithm, its running time is a function of its input length expressing the running time of the best known algorithm for solving the problem.

Probabilistic Algorithm. It is an algorithm $\mathcal{A}$ that has access to a *random tape* which is used on its computation. The random tape corresponds to a random bit string – or coin tosses – that can be modeled in the following way:

1. External coin tosses. $\mathcal{A}$ gets as additional input a random tape r. Then $y := \mathcal{A}(x, r)$, denotes the output of A on input x and fixed coin tosses r.
2. Internal coin tosses: $\mathcal{A}$ has the ability to make internally coin tosses. Then the output of $\mathcal{A}$ on input x, is denoted by $y \leftarrow \mathcal{A}(x)$. If we consider an algorithm running in polynomial time, then it can toss at most a polynomial number of coins.

Efficient Algorithm. An algorithm is considered efficient if there exists a polynomial $\mathrm{p}(\cdot)$ such that its running time on input $\kappa \in \{0,1\}^*$ is at most $\mathrm{p}(|\kappa|)$. This is typically adopted as the criterion of efficiency in computational complexity. On the other hand, algorithms that do not run in polynomial time are considered to be infeasible or intractable. We then use the term *efficient* algorithm to refer to those running in probabilistic polynomial time.
Thus a probabilistic algorithm running in polynomial time is denoted as PPT.

Security Parameter. It is desired to design cryptographic protocols in a way that one can configure its security strength. This is achieved by introducing some security parameter κ, intuitively, the larger the κ, the bigger the security of the underlying protocol. Then, we can consider the running time of an protocol to be a function of the security parameter κ, which is fixed during the setup of the protocol. Thus, when we say that an algorithm runs in polynomial time, we mean that its execution is bounded by some polynomial function in κ.[1]

[1] In this thesis, we consider the described protocols and adversaries as algorithms and we assume that they receive as input the security parameter in unary encoding during its initialization, even if it is not made explicit. Then the adversary is required to run in polynomial time.

We recall now the fundamental notion of *computational indistinguishability.* Loosely speaking, two distributions are computationally indistinguishable is no efficient algorithm can tell them appart. The definition of computational indistinguishability is formulated by considering infinite sequences of distributions. Such sequences are called probability ensembles.

Definition 2.7. *(Probability Ensemble). Let I be a countable set. A probability ensemble* ***indexed by*** *I is a collection of random variables $\{X_i\}_{i\in I}$.*

Let κ be the security paramter. Then an ensemble $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ is a sequence of random variables $X_1, X_2 \cdots$, one for each value of the security parameter, where the random variable X_κ may corrrespond to the output of a cryptographic protocol, e.g. the session key for fixed κ.

Definition 2.8. *(Computational Indistinguishability). Two probability ensembles $X = \{X_\kappa\}_{\kappa\in\mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are computationally indistinguishable, if for all polynomial-time distinguishers D there exists a negligible function negl(·) such that:*

$$|\Pr[\, D(1^\kappa, X_\kappa) = 1\,] - \Pr[\, D(1^\kappa, Y_\kappa) = 1\,]| \leq negl(\kappa)$$

Asymptotic Security. From the security perspective, the goal in cryptography is to design protocols which run in polynomial time, but have super-polynomial time security. In other words, we require that for a sufficiently large security parameter, the chances of breaking the scheme are too small to matter, i.e. a negligible function of the security parameter.

Concrete Security. When a cryptographic protocol is to be implemented, it is necessary to provide a specific value to the security parameter considering i) the desired *security* that the protocol aims to provide and ii) the *efficiency* of the protocol. The concrete security approach quantifies the security guaranteed by a protocol by providing an upper bound on the success probability of any adversary running for at most some specified amount of time.

The notion of *reduction* between two problems is essential when evaluating the security of a protocol. Borrowed from computational complexity theory, a reduction from problem A to problem B is an algorithm R that transforms A's inputs into B's inputs with the goal of solving A, i.e. given x as input to A, the reduction produces an input $\mathrm{R}(x)$ to B – we expect R and B to run in polynomial time. Then, R and B constitute a polynomial time algorithm for solving A. Intuitively, this captures the idea that solving A cannot be harder than solving B.

Definition 2.9. *(Reduction). A reduction from problem A to problem B, denoted by $A \leq_R B$, is an efficiently computable algorithm which transforms every instance of problem A into an instance of problem B, and which can use the solution to the instance of B to obtain a solution to the original instance of A.*

Definition 2.10. *(Random Self-Reducibility). Let problem P take as input $x \in S$, where S is the space of inputs. P is random self-reducible if there exists an efficient algorithm that transforms every instance of P with input x, into an instance of P with input $r \leftarrow S$, such that the answer to $P(x)$ can be derived in polynomial time from the answer $P(r)$.*

2.4 Provable Security Approach

When evaluating whether a cryptographic protocol is secure, the fact that no attacks have been found is considered as a first condition. However, there exists protocols that may take long time – some of them even years – before being broken, for instance the Chor-Rivest public-key scheme [CR85, CR88], based on the knapsack problem, survived for more than 10 years until it was broken [Vau98]. Another remarkable example is the attack on the Needham-Schroeder protocol [Low95]. Therefore, the lack of attacks should never be considered as the only security validation. To gain confidence in the security of a protocol, the cryptographic community proposed the so-called *provable security* approach. It is a research direction which aims to mathematically demonstrate that a property of a given protocol cannot be broken by the type of considered adversaries. Next, we recall the approaches considered in provable security:

1. Symbolic model: Often called the Dolev-Yao model [DY83], it is an abstract model where the underlying cryptographic primitives are assumed to be perfect and represented by function symbols considered as black boxes, messages are *terms* on these primitives and the adversary can compute only on these primitives. We refer to [CKW11] for a survey on symbolic models.
2. Unconditional security: Aims to provide security for computationally unbounded adversaries. A remarkable example in this category is the one-time pad encryption scheme, which satisfies Shanon's theorem of perfect-secrecy [Sha49].
3. Computational security: Introduced by Goldwasser and Micali [GM84a], the objective of this approach is to provide security for all adversaries running in polynomial time – more precisely, the adversary is an algorithm modeled as a probabilistic Turing Machine.

In this thesis we consider the computational model. It is rooted in the framework of complexity theory and security proofs in this model naturally follow the *reductionist* approach. A security proof consists of providing an *efficient* reduction from a well-studied computational problem to an attack against the protocol in question. If we conjecture that the chances of an efficient algorithm in solving the well-studied problem are a negligible function of the security parameter, then the chances of an efficient adversary in breaking the protocol are also a negligible function of the security parameter.

Remark 2.11. *Probably, the major drawback of the reductionist approach is that there could be the case that a security proof has not practical impact on the exact security provided by the protocol. For instance, even when there exists a polynomial-time reduction, it could be the case that the protocol is broken withing few hours, while the reduction, which leads to an algorithm solving the hard problem, requires a hundred years to run. The reason is that such reductions are only meaningful when sufficiently large parameters are used, furthermore, the security in the computational model does not answer the question of how large is sufficiently large. On the other hand, when a protocol is to be deployed, it is necessary to provide a concrete value for the security*

parameter taking into consideration the desired security level and the efficiency of the protocol. This motivates the need of tight security reductions (see Chapter 6).

Frequently, it is the case that security in the computational model follows either the game-based or the simulation-based approach:

- Game-based: Security is define as game played between the adversary and some challenger. The game explicitly defines a winning condition for the adversary, e.g. in the PAKE setting, it is to distinguish real session keys from random strings. When proving security in a game-based definition, the goal is to show that the advantage of the adversary on winning the game is negligible. Security models in this approach include [BPR00, AFP05].
- Simulation-based: Also referred as real world - ideal world paradigm, security is defined as being infeasible for an adversary to distinguish whether she interacts with the real protocol or with an idealized protocol that is secure by definition. Security models following this approach include [CHK+05, Can01, Sho99, BMP00].

2.4.1 Proof Structure using Sequences of Games

In this section we recall the so-called sequence of games approach. It is a well-known technique for organizing computational proofs introduced by Shoup [Sho04] and revisited in [BR06], it is specially relevant when considering game-based definitions. One of the strengths of this approach is that it is widely applicable, for instance it may be used when considering the random oracle model, standard model, common reference string (CRS) model, in the public-key or symmetric-key setting – even when the shared-key is a password – thus it provides an unifying methodology for various proofs.

The notion of security is defined via a security game played between a *challenger* and an *adversary*, both of them are modeled as probabilistic processes and the whole experiment is modeled as a probability space. The security of a protocol is associated with a particular event S, which corresponds to an adversary winning the game. Then a protocol is secure if *for all* adversaries $\Pr[S]$ is negligibly close to some target probability, typically 0 or $1/2$. [2].

This approach introduces a sequence of games $G_0, \cdots G_i \cdots G_n$ together with an associated event S_i for each game. The initial game G_0 corresponds to the original security experiment with respect to some protocol and adversary, then by definition $\Pr[S] = \Pr[S_0]$. The final game G_n defines a secure and idealized execution environment where $\Pr[S_n]$ is exactly the target probability and should be straight forward to compute. Then, intermediate games G_i are obtained from previous ones by *transitions* such that $\Pr[S_i]$ and $\Pr[S_{i+1}]$ are negligibly close. Thus it follows that $\Pr[S]$ is negligible close to the target probability and security is proved. As noted in [Sho04], *transitions* between games fall in the categories described in Figure 2.1.

[2] This is slightly different in PAKEs, where one has to incorporate the password defect into the security definition (see Chapter 4).

Transitions based on indistinguishability. Two consecutive games, G_i and G_{i+1}, are defined in such a way, that if the adversary is able to distinguish between them, then it is possible to obtain an algorithm that is able to distinguish between two distributions assumed to be computationally indistinguishable, i.e. a contradiction.

Transitions based on failure events. Two consecutive games, G_i and G_{i+1}, are identical unless some *bad* event F occurs, then it follows that $S_i \cap \bar{F} \iff S_{i+1} \cap \bar{F}$. From Shoup's Difference Lemma, in order to show that $|\Pr[S_i] - \Pr[S_{i+1}]|$ is negligible, it it is sufficient to demonstrate that $\Pr[F]$ is negligible.

Bridging step. This is corresponds to a transition where the game is simply formulated in a different way, such that $\Pr[S_i] = \Pr[S_{i+1}]$. Reasons for making a bridging step include i) obtaining an equivalent game but simpler to analyse and ii) preparation for one of the above type of transitions.

Figure 2.1: Transitions when using sequences of games.

Lemma 2.12. *(Difference Lemma.) Let A, B and F be events defined in some probability space such that $A \cap \bar{F} = B \cap \bar{F}$. Then $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

2.4.2 Idealized Models

The aim of cryptography is to develop protocols that can be used in practice. Nowadays, a cryptographic protocol is more likely to be standardized if i) it is provably secure and i) it can be efficiently implemented in the real world. Furthermore, it would be ideal to provide proofs of security which rely *only* on well-studied hardness assumptions. Unfortunately, most of the times such proofs can only be formulated for complex protocols which usually cannot be efficiently implemented. As solution, the cryptographic community has proposed some *ideal models* which permit to simplify the construction of protocols and its proof of security. More concretely, it places additional assumptions on the adversary by demanding that attacks on certain primitives are only generic. A generic attack against a primitive is one that runs independently of the details of how that primitive is implemented.

Standard Model

This is the model of computation where the only accepted assumption corresponds to the adversary being computationally bounded, where the security of a protocol relies only on some hardness assumption without any trusted setup. While ideal from the security perspective, protocols proven secure in this model tend to be inefficient in practice [GL01a].

Random Oracle Model

Formalized by Bellare and Rogaway [BR93b], the random oracle model is an idealized model where one assumes that a hash function is replaced by a random function that is publicly available

– known as the random oracle. In this model, the adversary cannot compute hash values herself, instead, she must query the random oracle. This strategy allows to provide security proofs for protocols which otherwise would be difficult to prove in the standard model.

When the security of a protocol is analyzed in the ROM, one has to assume that attacks on the protocol – if any – are independent of any particular instantiation of the hash function. Then, if there exists an attack to the protocol due to a weakness in the chosen hash function, one could simply replace the flawed one by a stronger candidate. There exists criticisms to the ROM [CGH04], mostly because it seems very difficult to build a truly random oracle. However, this model allows to provide security arguments for protocols which would be hard to prove in the standard model. In defense of the ROM, Katz and Lindell make the following statement: "A proof in the random oracle model is significantly better than no proof at all" [KL07, Chapter 13].

2.4.3 Computational Complexity Assumptions

Let $\mathcal{G}$ be an algorithm which on input a security parameter κ, outputs the description of a multiplicative group $\mathbb{G}$ of prime order q along with a generator $g \in \mathbb{G}$, i.e. $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(\kappa)$. We assume that the description of the group $\mathbb{G}$ and its generator g is public information which the adversary receives as input. Let $\mathbb{G}_T$ be a multiplicative group of prime order q.

Definition 2.13. *(Discrete Logarithm (DL) Problem). Given g^x compute x. Let the advantage of an algorithm $\mathcal{A}$ in solving the DL problem be:*

$$\mathrm{Adv}^{\mathtt{DL}}_{\mathbb{G}}(\mathcal{A}) = \Pr\left[x \xleftarrow{\$} \mathbb{Z}_q \,:\, x = \mathcal{A}(g^x) \right]$$

DL assumption: There exist sequences of cyclic groups $\mathbb{G}$, indexed by a security parameter κ, such that for all $\mathcal{A}$ running in time t polynomial in κ, $\mathrm{Adv}^{\mathtt{DL}}_{\mathbb{G}}(\mathcal{A})$ is a negligible function.

Definition 2.14. *(Computational Diffie-Hellman (CDH) Problem). Given (g^x, g^y) compute g^{xy}, where $\{g^x, g^y, g^{xy}\} \in \mathbb{G}$. Let the advantage of an algorithm $\mathcal{A}$ in solving the CDH problem be:*

$$\mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\mathcal{A}) = \Pr\left[(x, y) \xleftarrow{\$} \mathbb{Z}_q^2 :\, g^{xy} = \mathcal{A}(g^x, g^y) \right].$$

CDH assumption: There exist sequences of cyclic groups $\mathbb{G}$, indexed by a security parameter κ, such that for all $\mathcal{A}$ running in time t polynomial in κ, $\mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\mathcal{A})$ is a negligible function.
Note: In this thesis, we usually write $DH(\cdot)$ to the denote the solution to the CDH problem, i.e. $DH(g^x, g^y) = g^{xy}$.

Definition 2.15. *(Computational Square Diffie-Hellman (CSDH) Problem). Given g^x compute g^{x^2}, where $\{g^x, g^{x^2}\} \in \mathbb{G}$. Let the advantage of an algorithm $\mathcal{A}$ in solving the CSDH problem be:*

$$\mathrm{Adv}^{\mathtt{CSDH}}_{\mathbb{G}}(\mathcal{A}) = \Pr\left[x \xleftarrow{\$} \mathbb{Z}_q :\, g^{x^2} = \mathcal{A}(g^x) \right].$$

Under the *CSDH assumption* there exist sequences of cyclic groups $\mathbb{G}$ indexed by κ s.t. $\forall \mathcal{A}$ running in time t polynomial in κ, $\mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\mathcal{A})$ is a negligible function.

The equivalence between the CDH and CSDH problems is shown in [BCP04].

Claim 2.16. $\mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\mathcal{B}) \geq \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}^{\mathcal{A}})$*, where* $\mathcal{B}$ *and* $\mathcal{B}^{\mathcal{A}}$ *are PPT algorithms running in time* t*.*

Claim 2.17. $\mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}) \geq (\mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\mathcal{B}^{\mathcal{A}}))^2$*, where* $\mathcal{B}$ *and* $\mathcal{B}^{\mathcal{A}}$ *are PPT algorithms running in time* t *and* $t' = 2t + \mathcal{O}(t_{exp})$ *respectively, and* t_{exp} *denotes the time for exponentiation in* $\mathbb{G}$*.*

Definition 2.18. *(Decision Diffie-Hellman (DDH) Problem). Distinguish* (g^x, g^y, g^{xy}) *from* (g^x, g^y, g^z)*, where* $\{g^x, g^y, g^{xy}, g^z\} \in \mathbb{G}$*. Let the advantage of a algorithm* $\mathcal{A}$ *in solving DDH problem be:*

$$\mathrm{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\mathcal{A}) = \left|\Pr\left[(x,y) \xleftarrow{\$} \mathbb{Z}_q^2 : 1 = \mathcal{A}(g^x, g^y, g^{xy})\right] - \Pr\left[(x,y,z) \xleftarrow{\$} \mathbb{Z}_q^3 : 1 = \mathcal{A}(g^x, g^y, g^z)\right]\right|.$$

DDH assumption: There exist sequences of cyclic groups $\mathbb{G}$, indexed by a security parameter κ, such that for all $\mathcal{A}$ running in time t polynomial in κ, $\mathrm{Adv}_{\mathbb{G}}^{\mathtt{DDH}}(\mathcal{A})$ is a negligible function.

Definition 2.19. *(Bilinear map). A bilinear map is a function* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *such that the following properties are satisfied:*

- *Bilinear:* $\forall\ u, v \in \mathbb{G}$*,* $a, b \in \mathbb{Z}_q$*,* $e(u^a, v^b) = e(u, v)^{ab}$*.*
- *Non-degenerate:* $e(g, g)$ *generates* $\mathbb{G}_T$*.*
- *Computable:* $\forall\ u, v \in \mathbb{G}$*,* $a, b \in \mathbb{Z}_q$*, there is an efficient algorithm to compute* $e(u^a, v^b)$*.*

Definition 2.20. *(Bilinear Group).* $\mathbb{G}$ *is a bilinear group if there exists group* $\mathbb{G}_T$ *and a bilinear map* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$*.*

Gap Diffie-Hellman (Gap-DH) groups are those where the DDH problem can be solved in polynomial time but no PPT algorithm can solve the CDH problem with advantage greater than negligible, e.g. bilinear groups from Def. 2.20. More formally:

Definition 2.21. *(Gap-Diffie-Hellman (Gap-DH) Problem). Given* (g, g^x, g^y) *and access to a Decision Diffie-Hellman Oracle (DDH-O) compute* g^{xy}*.*

$$\mathrm{Adv}_{\mathbb{G}_1}^{\mathtt{gap-DH}}(\mathcal{A}) = \Pr\left[(x,y) \xleftarrow{\$} \mathbb{Z}_q^2 : DH(g^x, g^y) = \mathcal{A}^{\text{DDH-O}}(g^x, g^y)\right].$$

Gap-DH assumption: There exists sequences of *bilinear groups* $\mathbb{G}$ indexed by κ, such that for all PPT $\mathcal{A}$, $\mathrm{Adv}_{\mathbb{G}_1}^{\mathtt{gap-DH}}(\mathcal{A}) \leq negl(\kappa)$, where κ is the security parameter.

CHAPTER 3

Password Authenticated Key-Exchange Protocols

3.1 Introduction

It is undeniable that the internet has transformed the way we communicate and interact with the rest of the world. On a regular basis, internet users consume services such as e-mail, e-banking, e-commerce, social media and cloud storage. Nevertheless, these services are only possible if mechanisms to establish *secure communications* are available: It is necessary to keep the messages transmitted *secret* to potential adversaries, and to warranty that users and service providers are who they claim to be. For instance, a user who uses his credit card to buy a smart phone from Amazon, should be guaranteed that only Amazon has access to his credit card details. Fortunately, the cryptography community has developed over the last three decades the appropriate mechanism to tackle the described need.

3.2 Password Authenticated Key-Exchange protocols

A Password Authenticated Key-Exchange (PAKE) protocol is a cryptographic primitive that allows two entities, who only share a password, to establish a high entropy secret key by exchanging messages over a hostile network. The motivation is the usage of the established *session key* to protect the subsequent communication by building a *secure channel* between the parties involved. Theoretically, they are fascinating, because of their ability to use a weak secret – such as a password or a pin – to produce a strong cryptographic key in a *provably secure* way over an insecure communications network.

Similarly, Authenticated Key-Exchange (AKE) protocols allow the establishment of session keys over insecure networks. The distinction between PAKEs and AKEs is given by the *long-term* secret material: AKE protocols assume the existence either of shared high entropy symmetric keys or public-key infrastructure (PKI), which in practice, places additional requirements regarding the storage of such secret material or the need of trusted third parties (TTP) issuing the public-key certificates. On the other hand, PAKEs have *minimal requirements* with respect to the infrastructure required and the long-term secrets that users need to hold in order to succeed – as only a shared password is needed to execute the protocol. Particularly, no assumption is made regarding the quality of the shared secret, in fact, it could be as simple as a four-digit pin number.

The fact that passwords constitute the most extended method for authentication together with the minimal requirements in infrastructure, make PAKEs a promising primitive to be widely deployed for the establishment of secure communications. The seminal work in this area is the Encrypted Key-Exchange (EKE) protocol of Bellovin and Merritt in 1992 [BM92]. Their proposal was the first to show that it is possible to design a password authentication mechanism that can withstand offline dictionary attacks. Since then, numerous PAKE protocols have been proposed. Among them, only a handful have been considered for use in real-world applications: EKE [BM92], SPEKE [Jab96],[1] SRP [Wu98], KOY [KOY01], Dragonfly [Har08], SPAKE2 [AP05b], PPK and PAK [BMP00, Mac01b, Mac02a] and J-PAKE [HR10]. The PAK, J-PAKE, SRP and Dragonfly protocols have been standardized in the form of RFC5683, RFC8236, RFC2945 and RFC7664 respectively by the Internet Engineering Task Force (IETF). In parallel, prominent complexity-theoric security models for PAKEs have been proposed to get assurance on the claimed security properties by performing a rigorous analysis of the protocol in question [BPR00, AFP05, BMP00, CHK$^+$05].

Need of session keys. The PAKE setting assumes the existence of a shared password between two parties. Then one could question the need to execute a PAKE protocol to agree on a session key, given the fact that there is already a shared secret, i.e. one could naively consider that the shared password could be used directly to encrypt the conversation between the two parties, say Alice and Bob. The notion of a *session* helps us to understand why the naive approach is a bad idea. A session is a time frame for communication between two or more devices that occurs during the span of a single connection. For instance, a session is established everytime Bob inputs his PIN in an ATM; later on, another session is established when Bob pays with his credit card. That being said, we can provide several reasons why PAKEs are a better choice:

- The encryption of a message using a password as secret key leads to off-line dictionary attacks.
- It is desirable to limit the amount of ciphertext encrypted under the same key to prevent

[1]The SPEKE protocol [Jab96] is one of the most well-known PAKE designs. It has been proposed by Jablon in 1996 and proven secure in the SIM-BMP model under the Random Oracle (RO) assumption by MacKenzie [Mac01a]. SPEKE is practically relevant as it is specified in the ISO/IEC 11770-4 [ISO09] and IEEE P1363.2 [IEE02] standards.

cryptanalysis.

- It is desired that each communication session is established with a different session key.
  - This ensures independence across communication sessions or applications.
  - It confines the exposure caused by the compromise of a session key.
  - Session keys are created when required and deleted when the corresponding session expires, i.e. there is no need to store them.
- Some security properties cannot be satisfied in the naive approach, e.g. forward secrecy.

3.2.1 Passwords and Dictionary Attacks

A password is a human memorable string typically used to authenticate a user to some server. The advantages associated to passwords in authentication scenarios include:

- Usability: Passwords are simple to use, to set up and to remember by a user. Also, users are very much used to passwords in login scenarios, where a server needs to verify that a user is who he claims to be.
- No need of public-key infrastructure, which is a very good thing, since we know from recent history that sometimes certification authorities are not as trusted as they should. For instance, it was found that the certification authority Symantec had issued numerous certificates that did not comply with the industry standards.

On the other hand, the main disadvantage of passwords is given by the limitation of the human memory to remember long and random-looking strings, which forces users to choose passwords which are not only straightforward to remember but also easy to guess [YBAG04] and frequently re-used (highly correlated). Despite these notable disadvantages, passwords constitute still the most widely used mechanism for human-computer authentication, and they are extensively used in scenarios including simple e-mail access, internet banking or even national security.

The nature of passwords makes PAKE protocols vulnerable to *dictionary attacks*. In such attacks, an adversary tries to break the security of the protocol by exhaustively enumerating all possible passwords until a correct guess is found. This strategy might not be very successful on AKE schemes where the legitimate entities share a high-entropy key as long-term secret. However, in the PAKE setting the long-term secrets come from a small set of values, i.e. a dictionary, posing a genuine security threat. We distinguish between two types of possible dictionary attacks: *offline* and *online* dictionary attacks.

- In an offline dictionary attack, an adversary – who may be active or simply an eavesdropper – obtains information about the password that allows her to launch an exhaustive offline search. For example, consider an insecure protocol that leaks the hash of the user's password $\mathrm{H}(\pi)$ during its execution, where $H(\cdot)$ is a standard hash function. Then, an eavesdropper

who manages to obtain the $H(\pi)$, could go *offline* and try different password candidates $\{\pi_1, \pi_2, ...\pi_n\}$ until she finds the correct one by verifying that $H(\pi) = H(\pi_i)$.

- In an online dictionary attack, an attacker takes a candidate password from the password dictionary, *interacts* with a legitimate party by running the protocol and verifies whether the key-exchange succeeds for the candidate password or not. This verification occurs either as part of the protocol execution or in higher layer applications that uses the established session key. Since passwords come from a small set, this attack has a non-negligible chance of success.

Online dictionary attacks cannot be entirely prevented, but their damaging effect can be mitigated to some extent, for example by requiring users to choose strong passwords, limiting the number of unsuccessful login attempts before the user account is blocked, or even using machine learning to detect a pattern in login attempts that suggests an online dictionary attack might be in progress. On the other hand, an offline dictionary attack occurs when the execution of the protocol allows an adversary to launch an exhaustive offline search of the password. The intuition of security requires PAKEs to be only vulnerable to online dictionary attacks, while offline dictionary attacks have a devastating effect and must be prevented.

We would like to note that there exists a tremendous effort made by researchers and practitioners to enhance the security provided by password-based authentication, either in client-side or server-side. Recent works in this direction include: i) two-factor authentication mechanisms, where a user who wishes to authenticate to some server needs to prove the knowledge of the password and the possession of an auxiliary device [JKSS18], ii) password managers allow users to store and retrieve high-entropy passwords based on a single master-password which the user needs to remember [SJKS17], and iii) Juels and Rivest [JR13] propose the use of an auxiliary server, called the Honey-Checker, to detect whenever the password file – stored at the server – gets compromised, while the authors in [BRRS18] propose the adoption of a PAKE protocol into the Honey-Cheker to protect the password while in transit to the server.

3.3 Security Properties in PAKE Protocols

A PAKE protocol is a two-party protocol that is executed in a concurrent scenario. This means that each party is allowed to run multiple instantiations of the protocol simultaneously, i.e. multiple *sessions*. To complicate things, it is a two-party protocol for honest users but executed in a multy-party setting, which makes *authentication* and *consistency* essential requirements.

Authentication: At the end of the protocol execution, each principal should be able to verify who they shared the session key with. PAKEs provide two kinds of authentication:

1. Implicit authentication: After completing a protocol run, a party U holds a session key sk. Implicit authentication guarantees that only its *intended partner* could have computed the same sk, or put differently, if U has not conversed with its intended partner, whoever it

ran the protocol with will fail at computing the same session key sk that U holds. The key confirmation typically occurs in higher application layers whenever the session key is used.

2. Explicit authentication: After successfully completing a protocol run, a party U gets the guarantee that it conversed with its intended partner and that its partner has also computed the correct session key. This is usually achieved by the incorporation of key confirmation codes in the protocol description.

A *session* is simply one protocol run being attempted by an honest party. When a party U completes one protocol execution, he is not only expected to hold a session key, it is crucial that he knows who he shares the session key with and for which particular session. Then, U outputs (sk, sid, pid), where sk denotes the established session key, sid is an unique session identifier and pid is its partner identity, i.e. the user that U thinks he shares the session key with.

Consistency: At the end of the protocol, if two principals establish a common session key sk, then both need a consistent view of who is their partner for that particular session [Kra03]. Concretely, if A outputs $(sk_A, sid_A, pid_A = B)$ and B outputs (sk_B, sid_B, pid_B), if $sk_A = sk_B$ and $sid_A = sid_B$, then it must be the case that $pib_B = A$.

Additionally, PAKE protocols need to satisfy the following properties [HR10]:

- Eavesdropping resistance: An adversary observing the execution of the protocol should not gain any useful information from the communication.
- Offline Dictionary Attacks resistance: An adversary should not be allowed to verify password guesses without actively interacting with the honest users.
- Online Dictionary Attacks bound: An adversary should be limited to test *at most* one password per session during an active attack.
- Known session key security: Compromise of a session key should not compromise other session keys.
- Forward Secrecy: Compromise of long-term secret material should not compromise previously established session keys.

We distinguish between the so called perfect forward secrecy (PFS) and weak forward secrecy (wFS). The notion of *weak forward secrecy* (wFS) protects session keys after compromise of long-term key material, but only for those sessions created *without the active participation of the attacker* [Kra05]. On the other hand, the stronger notion of perfect forward secrecy (PFS) protects all session keys established before the compromise of the long-term keys, even if created with the active intervention of the adversary (see Chapter 5 for a detailed discussion).

3.3.1 Augmented vs Balanced PAKEs

Dictionary attacks are certainly at the center of the discussion when evaluating the security of PAKE protocols. Due to the nature of passwords, it is important to question whether some

security can still be preserved even if the user password gets compromised – one could immediately think of forward secrecy as an example, however, that is not the aim in this section. Obviously, there is no hope of preventing an adversary, who manages to obtain the password in *clear* text, from masquerading as a client to the server *immediately after* the password leakage and agreeing on the same session key as the server. The question is whether compromise of some party immediately reveals the corresponding passwords in clear text.

We consider the client - server setting where the server stores a *password file* which contains the necessary password information for all clients. During the execution of a PAKE protocol, the server retrieves the necessary password information from this file. Having such amount of secret and valuable information concentrated in a single file makes it an attractive target for adversaries, for instance, in 2012, 68 millions hashed passwords where stolen from Dropbox and in the same year, 6 million passwords where stolen from Linkedin. In both cases, the strategy followed by the companies was to ask the affected users to reset their password. Thus, it is favorable to design PAKEs that still preserve some degree of security in case of compromise of the password file. More concretely, it is highly desirable that compromise of the password file does not *immediately* allow the adversary to masquerade as a user, i.e. the adversary should be expected to perform an exhaustive offline dictionary attack in order to masquerade as a client to the server during a PAKE execution.

In this section we consider the so-called *resilience to server compromise*, introduced by Bellovin and Merrit [BM93]. This security property requires that, even if the password file has been compromised, in order to masquerade as a user, the adversary needs to extract the clear text password from the file and the cost of this computation *per user* should be linear in the size of possible passwords – assuming passwords are uniformly distributed. For the sake of clarity on this property, we consider a common login scenario (no PAKE involved): in case the password file stores the passwords in clear text, once it is leaked, the adversary obtains the necessary information to immediately masquerade as a client to the server. However, if the password file stores salted passwords, the adversary is forced to perform an exhaustive offline dictionary attack and the cost of such attack per user is linear in the size of the password dictionary, with the hope that the compromise is detected by then and users can be asked to change their passwords. Unluckily, for PAKEs, guaranteeing resilience to server compromise is not as straight forward as simply storing salted passwords.

A PAKE protocol is said to be *augmented* if it is resilient to server compromise, and *balanced* otherwise. In a balanced PAKE, all the information that a client needs to run the protocol can be retrieved from the password file and *directly* inserted in the protocol execution, even if hashed and salted passwords are stored in the file. Example of balanced PAKEs includes the SPAKE2, PFS-SPAKE2, PPK and PAK protocols that we consider in this work. On the other hand, in augmented PAKEs, the server does not store the password itself, but only a *verifier* whose purpose is to check that the client uses the correct password. Example of augmented PAKEs includes the PAK-Z [Mac02a] and VTBPEKE [PW17] protocols. Additionally, Gentry et al. [GMR06] propose a general technique to transform any balanced PAKE into an augmented

one with security in the UC framework.

Resistance to pre-computation attacks. So far, all the augmented PAKEs we have mentioned, are vulnerable to pre-computation attacks that may lead to the instantaneous compromise of user passwords once the server gets compromised, i.e. while they do force the adversary to compute an offline dictionary attack to extract the *clear* password from the password file, this computation could be performed *before* the compromise of the server. More recently, Jarecki, Krawczyk and Xu introduced the notion of *strong* augmented PAKE [JKX18], which is an augmented PAKE with the extra requirement that pre-computation attacks are not allowed, i.e. the exhaustive offline dictionary attack is only possible *after* the password file has been leaked. They present a compiler that transform any augmented PAKE into a strong augmented PAKE with security in the UC model.

3.4 Potential Application of PAKE protocols

We recall that the remarkable property provided by any PAKE protocol is the protection of the user's password. That being said, in the following sections we consider two scenarios where such cryptographic protocols could be applied.

3.4.1 Establishment of secure channels

Probably, the most natural application of PAKEs is in building secure channels between two principals who share a password. Considering that passwords constitute still the most widely used method of authentication, where typically some user U shares a password with some server S, the potential of PAKEs is fascinating. As a motivating example, let us consider the scenario of a customer who uses his credit card in a card-present transaction. To validate the transaction, the customer introduces his credit card and PIN in the payment terminal. The credit card information, details of transaction and PIN are transmitted securely to the bank so it can accept or reject the operation. One requirement to accept the transaction is the verification of the PIN: the bank needs to verify that, whoever makes the transaction knows the PIN associated to the credit card. This is where one could benefit from adopting PAKEs in such transactions: Since the customer and bank share the PIN, they could run a PAKE protocol to establish a secure session and then use that secure channel to transmit the necessary data. An interesting advantage of such approach is given by the *explicit* authentication provided by PAKEs: the bank could verify whether the user knows the correct PIN while intrinsically protecting it from potential adversaries. In this case, the verification of the PIN occurs as part of the PAKE protocol in such a way that it is not necessary to transmit it to the bank i.e. the PIN never leaves the payment terminal.

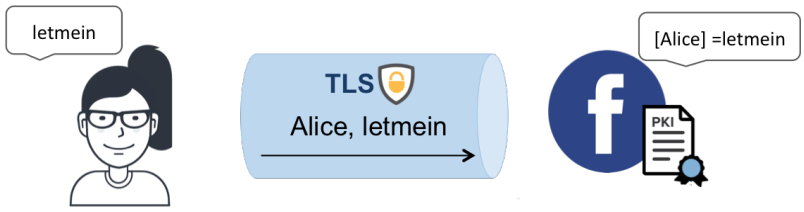

Figure 3.1: Traditional approach to login.

3.4.2 Login scenarios

Another setting where one could benefit from the deployment of PAKEs is in login systems. To understand the advantage of using PAKEs for such purpose, we first look at the traditional login approach which relies on a TLS connection.

Suppose some user U wants to authenticate to server S to access resources in S. The server stores all password related information – for all clients – in the *password file*.[2]. Whenever the user desires to authenticate himself to some server, he inputs his password directly into the server's web page. Concretely, the following occurs (see Figure 3.1):

1. User U visits the Server web page and a TLS connection is established. Frequently, the TLS configuration that is used provides only *unilateral* authentication, as the client authenticates the server using the server's public-key certificate but not the other way around.
2. To verify that the user is who he/she claims to be, the user sends his credentials – user name and password – to the server encrypted over the TLS-channel. This is meant to protect the password while it is in transit.
3. The server decrypts the message and verifies that the received password matches with its records.

Generally speaking, the concern with the aforementioned approach is that it relies on the user to determine whether it is safe to input his password in the web page or not. For instance, let us consider two realistic PKI failures that lead to compromise of the user's password:

- A certification authority could issue bogus certificates. This is something that was already observed in the Symantec scandal.
- In a phishing attack, an adversary may obtain a valid certificate and then trick a user to visit the attackers web page.

The problem with traditional login is that the user sends his password to the server in order to authenticate. The password is usually sent TLS-encrypted, however, due to real-world PKI failures, there exist ways in which the adversary could capture the user's password, for instance

[2]In order to avoid the *immediate* password leakage upon server compromise, it is recommended not to store the passwords in clear text but rather salted and hashed, i.e. H(s, π), where π is the user password, s is a unique per-password salt and $H(\cdot)$ is a hard-memory function like Scrypt [Per].

Figure 3.2: The user password is exposed to the adversary in phishing attack.

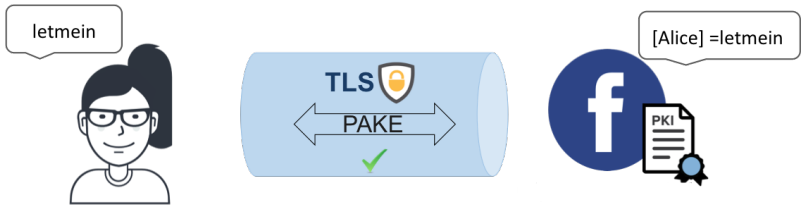

Figure 3.3: The adoption of TLS + PAKE as login mechanism guarantees that the user password is never exposed. If the user attempts to authenticate himself to a phisher, the PAKE execution fails and the fisher does not learn the password.

phishing attacks. In fact, the user may observe in his browser "secure connection", even if the TLS connection was established with the adversary (see Figure 3.2).

Fortunately, PAKE protocols allow two principals to authenticate each other without exposing the *password*. This property is what makes PAKEs suitable for login scenarios [OWT09, EKSS09]. The idea is as follows: First, establish a TLS-connection between client and server using the server's public-key certificate. Then, run a PAKE protocol inside the TLS-channel, this should allow the server to authenticate the client with the advantage that the password is intrinsically protected (see Figure 3.3).

Remark 3.1. *Despite the clear advantages offered by PAKEs in login scenarios, there exists usability concerns that prevent their adoption. The TLS + PAKE approach requires the integration of PAKE protocols in the browser [EKSS09]. More concretely, this approach requires the design of a secure area within the browser where the user inputs his password. This is probably the biggest concern from the usability perspective, as it requires the users to change from typing their credentials directly in the web page into an easy to identify secure area embedded in the browser. Similarly, the TLS + PAKE approach would restrict internet companies from handling events such as forgotten or mistyped passwords. Usually, each company has its own custom-made means of handling such event, for instance, blocking the account after a number of failed attempts, asking security questions to recover the password, resetting the password, etc. These policies seem difficult to implement if the authentication mechanism is executed directly with the browser.*

CHAPTER 4

Security Models for Password Authenticated Key-Exchange Protocols

4.1 Introduction

The design of PAKE protocols has proved to be a non-trivial task. In the past, protocols were proposed and assumed to be secure if some people trying to break them did not succeed. We know from history that the success rate of this approach is not impressive, consequently, PAKEs where broken sometimes years after they were published [AP05a, Szy06, BŠŠ17, CH14, MPS00a, Pat97]. Thus, *security models* for (Password) Authenticated Key Exchange protocols emerged to get assurance on the claimed security properties by performing a rigorous analysis.

Bellare and Rogaway introduced the first *complexity-theoretic treatment* of the notion of security for AKE protocols [BR93a], commonly referred as the BR model. This approach requires to define the capabilities of the adversary and what constitutes a break in the protocol, or put in other words, the security properties that the protocol must satisfy when it is executed in the presence of an adversary. Following the definition is a security proof to demonstrate that the protocol in question satisfies it. The proof follows the complexity-theoretic methodology [GM84b], where a reduction is demonstrated from a problem believed to be hard to the problem of breaking the protocol.

The original work of Bellare and Rogaway considers the realistic scenario of concurrent sessions running on a network fully controlled by the adversary. It focuses on the case where users previously share a symmetric key as means of authentication. Subsequent works [BM97, BR95] consider the public key and three party setting. The BR model was also extended to the password setting in [BPR00, AFP05].

4.1.1 Two Flavors of Security Models

There exists two main approaches to capture the security of protocols. Security models which follow the original BR model are referred as *indistinguishability* based. The second approach is the *simulation* based model, introduced for AKE protocols by Shoup [Sho99] and later extended to the password setting in [BMP00]. We provide a sketch of both approaches:

- IND-based: Also known as game-based security. A protocol is secure, if under the *allowed adversary actions*, it is computationally infeasible for an attacker $\mathcal{A}$ to distinguish an established *session key* from a random string. Two examples of this approach are the IND-FtG and IND-RoR security models [BPR00, AFP05].

- SIM-based: This approach defines two worlds: i) an *ideal world*, secure by definition, which describes the service that is to be provided and ii) the *real world* which is the real protocol execution against an attacker. Security in the SIM setting asks for the computational indistinguishablity between ideal world and real world executions. Two examples of this approach are the SIM-BMP and UC security models [BMP00, CHK$^+$05].

In IND-based definitions, security is defined via a game played between the adversary and a challenger and the goal of the adversary is to distinguish session-keys from random strings. We have already described the methodology to construct security proofs for IND-based definitions in Section 2.4.1. On the other hand, security proofs for SIM-based security definitions follow a different approach: for all adversaries in the real world, one has to construct a simulator in the ideal world that generates an execution computationally indistinguishable from the real one. We refer to [Lin17] for an in-depth description of the simulation paradigm.

It is generally accepted that simulation-based models better capture the real-world security requirements in cryptographic protocols. For instance, we now briefly look at the password distribution and refer to Chapter 7 for an in-depth discussion.

Password Distribution. We know that users have the tendency to choose weak passwords. Furthermore, i) some passwords are more likely to be chosen than others and ii) users tend to re-use passwords – use the same password across multiple websites [WRBW16]. Usually, simulation-based models do not make any assumption regarding the password distribution. On the other hand, indistinguishability-based definitions assume that i) passwords are uniformly distributed and ii) passwords are not correlated, which might not reflect the way passwords are distributed in the real world.

While simulation based models may be better at capturing some security requirements, indistinguishability based models are easier to work with and construct security proofs in. Therefore, for the rest of this chapter and Chapters 5 and 6 we focus on indistinguishability-based models. We discuss simulation based models again in more detail in Chapter 7.

4.2 IND-based Find-then-Guess Security Model

Next we describe the well-known security model of Bellare, Pointcheval and Rogaway [BPR00], which we use to prove the security of the SPAKE2 and PFS-SPAKE2 protocols in Chapter 5, and of the PAK protocol in Chapter 6.

Frequently referred to as the Find-then-Guess (FtG) model, it is an extension of the Bellare and Rogaway model [BR93a] to the password setting. It guarantees semantic security for the targeted session key under the realistic scenario of an adversary having full control of the network, concurrent executions of the protocol, users losing their session keys and forward secrecy. It also incorporates the essential requirements that PAKE protocols must satisfy: i) an eavesdropper adversary should not learn any information about the password, and ii) an adversary can verify at most one password guess per session in an active attack.

In the FtG model, security is defined via a security experiment played between a challenger $\mathcal{CH}$ and some adversary $\mathcal{A}$. The task of $\mathcal{CH}$ is to administrate the security experiment while keeping the appropriate secret information outside from $\mathcal{A}$'s view. Roughly speaking, $\mathcal{A}$ wins the experiment if she is able to distinguish the established session key from a random string.

We will start by formally defining PAKE protocols. This will be followed by an in-depth description of the FtG security model.

PAKE protocol. A PAKE protocol is defined as a pair of algorithms $(Gen, \mathcal{P})$. Gen is the password generation algorithm. It takes as input the password dictionary D, a probability distribution $\mathcal{Q}$ and initializes the protocol participants with passwords in D drawn from $\mathcal{Q}$. $\mathcal{P}$ is a probabilistic algorithm that defines how users respond to signals from the environment.

Participants and passwords. Each participant is either a client $C \in \mathcal{C}$ or a server $S \in \mathcal{S}$. Let $\mathcal{U} = \mathcal{C} \cup \mathcal{S}$ denote the set of all (honest) users. Each client C holds a secret password π_C and server S holds a vector of passwords for all clients i.e. $\pi_S =< \pi_C >_{C \in \mathcal{C}}$ such that for all clients C then, $\pi_S[C] = \pi_C$. These passwords are assigned to users during the initialization of the protocol by running the Gen algorithm. We consider the client - server scenario where there is a single server S. For simplicity, the passwords are assumed to be independent and uniformly distributed.

User Instances. The users might have more than one protocol execution running simultaneously, this is modeled by allowing each user an unlimited number of *instances* with which to execute the protocol. Specifically, let Π_U^i denote the i-th instance of user $U \in \mathcal{U}$. In cases where distinction matters, let Π_C^i and Π_S^j denote the i-th and j-th instance of client $C \in \mathcal{C}$ and server S respectively.

Protocol execution. We assume the presence of a PPT adversary $\mathcal{A}$ with full control of the network. This means that principals solely communicate through the adversary, and she may delay, reorder, modify, drop messages sent by honest principals, or inject messages of her choice in order to attack the protocol.

The goal of $\mathcal{A}$ in the security experiment is to forge the semantic security of the established session keys. More formally, we consider the interaction of two probabilistic algorithms: the adversary $\mathcal{A}$, and the challenger $\mathcal{CH}$. The challenger internally simulates as many user instances as A requests. The interaction between $\mathcal{A}$ and $\mathcal{CH}$ takes place via the following queries:

- **Send**(U, i, m): A message m is sent to instance Π_U^i. The instance processes the input message according to the protocol description $\mathcal{P}$ and outputs a response which is given to $\mathcal{A}$. To instruct client C to initiate a session with server S, the adversary sends a message containing the name of the server to an unused instance of C, i.e. $\mathsf{Send}(C, i, S)$. In addition, $\mathcal{A}$ is notified about any change on the instance's state which include:
  - continue: The instance Π_U^i is ready to receive and process another message.
  - reject: Π_U^i aborts the protocol execution without computing the session key. This is due to receiving an unexpected message, for instance an invalid confirmation code.
  - accept: Π_U^i knows its partner identity pid_U^i, the session identifier sid_U^i and session key sk_U^i.[1] However, Π_U^i still expects to receive another message to fulfill the protocol specification, usually a confirmation code.
  - terminate: Π_U^i holds pid_U^i, sid_U^i and sk_U^i. It has completed the protocol execution and will not send nor receive any other message.
- **Execute**(C, i, S, j): This query causes an honest run of protocol $\mathcal{P}$ between Π_C^i and Π_S^j. The transcript of the execution is given to $\mathcal{A}$.
- **Reveal**(U, i): The session key sk_U^i held at Π_U^i is given to $\mathcal{A}$. For this query to be valid, sk_U^i must be already computed, i.e. Π_U^i must be in *accept* or *terminate* state.
- **Corrupt**(U). The adversary obtains the password of user U. If $U = C \in \mathcal{C}$, then $\mathcal{A}$ receives π_C, and if $U = S \in \mathcal{S}$, then $\mathcal{A}$ receives $\pi_S =< \pi_C >_{C \in \mathcal{C}}$.
- **Test**(U, i): $\mathcal{CH}$ flips a bit b and answers the query as follows: if $b = 1$ $\mathcal{A}$ gets the session key sk_U^i, otherwise she receives a random string $r \xleftarrow{\$} \{0,1\}^\kappa$, where $\{0,1\}^\kappa$ denotes the session key space. The adversary is allowed to ask this query only once.

The previously described queries model reasonable capabilities that the adversary may have in the real world. The underlying motivation is that the protocol in question should remain secure even in the presence of such a powerful adversary. Next we mention the security properties captured by each query:

1. Send query. It models the fact that the network may be under full control of the adversary – or shortly, the adversary is the network. Certainly, this is powerful query that covers numerous attacks that the adversary may try to perform, for instance impersonation, replay and man-in-the-middle attacks, just to mention a few.

[1] The partner identity pid_U^i, the session identifier sid_U^i and session key sk_U^i, were introduced and discussed previously in Section 3.3.

2. Execute query. This query models eavesdropper adversaries, who may passively observe honest executions of the protocol. This query can certainly be modeled via Send queries, however, it seems to be a better approach to model passive adversaries via Execute queries for PAKE protocols. The reason is that, the non-negligible term in the advantage function models online-dictionary attacks. This term increases linearly with the number of Send queries asked by the adversary (see Definition 4.1 and 4.2). Clearly, there cannot be an online dictionary attacks if the adversary is merely eavesdropping the conversation. Then modeling honest executions via Send queries would unnecessary increase the non-negligible term in the advantage function.

3. Reveal query. It models the compromise or poor management of established session keys. The idea is that leakage of session keys should not result in the compromise of other established session keys, also known as *known session-key* independence.

4. Corrupt query. This query models the compromise of the long-term secret, which due to security breaches, is reasonable to consider it as a legitimate adversarial capability. This query, together with the *freshness* condition, are necessary to model the security property of *forward secrecy*.

5. Test query. This is the only query that does not correspond to an adversarial capability in the real world. Instead, this query measures the success of the adversary on breaking the protocol in question. As we will see shortly, this query is meaningful only if certain constraints are placed on the target instance of the query.

Accepting and terminating. In the FtG model from [BPR00], an instance Π_U^i *accepts* whenever it holds a session key sk_U^i, a session ID sid_U^i and a partner ID pid_U^i.[2] An instance Π_U^i *terminates* if it holds sk_U^i, sid_U^i, pid_U^i and additionally, it is not expected to send nor to receive any more messages. The difference between both states is determined by the protocol design. For instance, in the PAK protocol (see Figure 6.1), once a server instance sends the second message, it is his duty to compute the sk_U^i, sid_U^i, pid_U^i and *accept* – therefore it is eligible to a Reveal query, nevertheless, it still needs to receive a last message to *terminate*. On the other hand, when considering the PFS-SPAKE2 protocol (see Figure 5.5), the server is not asked to compute the session key unless it has received the last message.

Next we provide definitions that will help to provide a meaningful definition of security. Concretely, the *partnering* and *freshness* definitions permit to appropriately define the advantage function by removing the scenarios where the adversary could trivially win the security experiment by using the previously described queries. In addition, the *partnering* definition helps us to incorporate the *consistency* property into the model (see Section 3.3).

Partnering. Two instances, Π_C^i and Π_S^j, are partnered if both *accept*, each one has computed the session key, session identifier and partner identity. Specifically, the client and server instance hold $(sk_C^i, sid_C^i, pid_C^i)$ and $(sk_S^j, sid_S^j, pid_S^j)$ respectively such that:

[2]Note that the meaning of "accept" in this context is different from the usage of "accept" in other settings such as computational complexity.

1. $sk_C^i = sk_S^j$, $sid_C^i = sid_S^j$, $pid_C^i = S$, $pid_S^j = C$ and
2. no other instance accepts with the same session id sid, except with negligible probability.

As said before, there exists scenarios where an adversary could trivially win the security experiment, for instance, by making a Test query to some instance Π_U^i followed by a Reveal query targeting the same instance or its partner. The notion of freshness – together with the definition of partnering – allow us to prevent such simplistic scenarios from happening. The idea is to give credit to $\mathcal{A}$ only if the instance target of the Test query is *fresh*.

We define two notions of freshness depending on weather perfect forward secrecy (PFS) of weak forward secrecy (wFS) is required.

PFS-Freshness. An instance Π_U^i is *PFS-fresh* unless:

- A Reveal query was made to Π_U^i or its partner or
- There was a Corrupt(U') and a Send(U, i, m) query, Π_U^i does not have a partner and the corruption of *any user* U' occurs **before** the Test query.

wFS-Freshness. An instance Π_U^i is *wFS-fresh* unless:

- A Reveal query was made to Π_U^i or its partner or
- There was a Corrupt(U') and a Send(U, i, m) query, Π_U^i does not have a partner and the corruption of any user U' occurs at *any* time.

The first flavour of *freshness* models perfect forward secrecy. The intuition is to consider as legitimate target of a Test query those instances whose session keys were negotiated *before* the corruption of any principal. This definition is similar to that of [BPR00] and is typically used in the client-server setting [Mac02a, ACP05], where compromise of the server results in compromise of the passwords for all clients. The only difference to [BPR00] is that our definition also considers an instance to be fresh if it has a *partner* (regardless of any corruption), i.e. whenever the session key is the result of untampered communication between honest instances.

The second variant of freshness models weak forward secrecy, which does not guarantee the secrecy of those sessions keys which were negotiated with an *active* intervention of the adversary (determined via *partnering*) and somebody has been corrupted either *before or after* the establishment of the session key.

It was previosly noted in [HR10], that a passive adversary does not benefit from corrupting a user. A similar reasoning is considered in our definitions of PFS-Freshness and wFS-Freshness, particularly, we consider a session fresh if it has a partner, i.e. regardless of any Corrupt query (obviously, the condition regarding Reveal queries must be preserved).

After asking a number of queries as described in the protocol execution paragraph, the adversary $\mathcal{A}$ outputs a bit b'. We say that $\mathcal{A}$ wins the security experiment if the single Test query was directed to a *PFS-fresh* instance and $b' = b$. Let $\text{Succ}_P^{\text{PFS-FtG}}$ be the event that this occurs.

Advantage of the adversary. The advantage of $\mathcal{A}$ on attacking the protocol P is defined as follows:

$$\mathrm{Adv}_P^{\text{PFS-FtG}}(\mathcal{A}) = 2 \cdot \Pr\left[\,\mathrm{Succ}_P^{\text{PFS-FtG}}(\mathcal{A})\,\right] - 1 \tag{4.1}$$

Definition 4.1. *(PFS-FtG security). Protocol P is FtG secure and satisfies perfect forward secrecy if for all PPT adversaries making at most n_{se} Send queries, there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\mathrm{Adv}_P^{\text{PFS-FtG}}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + \epsilon(\kappa),$$

where D is the password dictionary and κ is the security parameter.

We similarly define FtG security with *weak* forward secrecy, the only change is that the adversary $\mathcal{A}$ must ask the the single Test query to a *wFS-fresh* instance.

Definition 4.2. *(wFS-FtG security). Protocol P is FtG secure and satisfies weak forward secrecy if for all PPT adversaries making at most n_{se} Send queries, there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\mathrm{Adv}_P^{\text{wFS-FtG}}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + \epsilon(\kappa),$$

where D the password dictionary and κ is the security parameter.

While online dictionary attacks cannot be entirely prevented, the adversary should be limited to test at most one password per session during an active attack. Taking this into consideration, the non-negligible term in the advantage function is an upper bound on the password guesses the adversary is allowed to make during an active attack. It is a function on the number of Send queries the adversary makes, as it takes at least one Send query to test one password per session.

It is easy to see that PFS-FtG $\rightarrow$ wFS-FtG security. *Weak forward secrecy* (wFS) protects session keys established before the compromise of long-term key material, but only for those sessions created *without the active participation of the attacker* [Kra05], while perfect forward secrecy (PFS) protects all session keys established before the compromise, even if created with the active intervention of the adversary.

Fact 4.3. *This fact can be easily verified.*

$$\Pr\left[\mathrm{Succ}_P^{\text{FtG}}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}_{P'}^{\text{FtG}}(\mathcal{A})\right] + \epsilon \;\Leftrightarrow\; \mathrm{Adv}_P^{\text{FtG}}(\mathcal{A}) = \mathrm{Adv}_{P'}^{\text{FtG}}(\mathcal{A}) + 2\epsilon. \tag{4.2}$$

CHAPTER 5

Forward Secrecy for SPAKE2

5.1 Introduction

The Simple Password-based Encrypted Key-Exchange (SPAKE2) protocol, proposed by Abdalla and Pointcheval [AP05b], is an efficient cryptographic protocol that allows two users to establish a secure channel using only their shared password for authentication purposes. At the moment of writing, it is being considered by the IETF working group for standardization and possible integration in the TLS 1.3 standard. Although it has been proven secure in the Find-then-Guess model of Bellare, Pointcheval and Rogaway [BPR00], whether it satisfies some meaningful notion of *forward secrecy* remains an open question.

In this chapter, we prove that the SPAKE2 protocol satisfies the so-called *weak forward secrecy* introduced by Krawczyk [Kra05]. Furthermore, we demonstrate that the incorporation of key-confirmation codes in SPAKE2 results in a protocol that provably satisfies the stronger notion of *perfect forward secrecy*. As forward secrecy is an explicit requirement for cipher suites supported in the TLS handshake, we believe this work could fill the gap in the literature and facilitate the adoption of SPAKE2 in the recently approved TLS 1.3.

5.1.1 SPAKE2 Protocol

The SPAKE2 protocol, proposed by Abdalla and Pointcheval [AP05b], is a one-round PAKE protocol proven secure in the Find-then-Guess (FtG) model of Bellare et al. [BPR00] without considering any notion of *forward secrecy*. It is a simple, yet efficient protocol that, in addition to the pre-shared password, requires the protocol participants to share two Common Reference Strings (CRS) prior to the protocol execution. The adoption of the CRS yields to an elegant construction that does not require full domain hash functions, which are hard to implement

efficiently in practice. On the other side, the CRS requires extra security assumptions that might be easy to satisfy in some scenarios but very restrictive in others [GL01b]. Also, as it is a one-round protocol, only *implicit authentication* can be satisfied. Fortunately, the incorporation of *key-confirmation codes* allows the protocol participants to explicitly authenticate each other. This is a well-known technique described in more detail in [KJP06] and in Chapter 40 of [Vac13].

More recently, the Internet Engineering Task Force (IEFT) community has revisited the deployment of the SPAKE2 protocol: i) as a stand alone specification [LK18], ii) as pre-authentication mechanism in the Kerberos protocol [MSHH18] and iii) its adoption in the TLS 1.3 protocol, specifically in the *handshake* stage when *pre-shared* keys for authentication are available [BF18, Res18] (a shared password in this setting). The discussion of forward secrecy in SPAKE2 has been a common factor in the aforementioned Internet Drafts (I-D).

5.1.2 PAKEs adoption in TLS

Nowadays, the TLS protocol is the *de-facto* standard to protect internet communications. It consists of two stages: first the *Handshake* protocol allows the two parties to agree on a session key, secondly, the *Record* protocol uses the previously negotiated session keys to protect the messages exchanged between the parties. Most of the TLS implementations provide only *unilateral* authentication, where client C authenticates server S by means of public-key infrastructure (PKI) during the handshake, usually identity disclosure of client to server is not supported.

While the server-authenticated approach might be sufficient for scenarios like internet surfing, it is certainly inadequate for email access, internet banking and social media login applications, where a client C has to authenticate to a server S to gain access to resources in S. The naive approach of client authentication requires C to send his/her user name and password – either in clear or hashed – to S which it verifies against its records. This approach lets the client's password exposed and vulnerable to a number of attacks: eavesdropping, offline dictionary, and phishing attacks. The first two kinds of attacks can be prevented if the client sends his user/password through a server-authenticated TLS channel, however, it seems harder to counter the latter attack even if a TLS channel is available. The problem with phishing attacks is the following: an adversary can clone a legitimate website and fool the client to visit the fake website. To make things worse, the adversary can manage to obtain a valid public-key certificate from a certification authority (CA) for her illegitimate web page. Indeed, the client may see on his web browser "secure connection" as a TLS connection may be established between the client and the cloned website controlled by the adversary (a typical client should not be expected to verify the certificate details). Then the client inputs his credentials to login and the adversary simply learns the user/password combination.

Fortunately, PAKEs stand as a strong candidate for scenarios where two parties require to mutually authenticate each other while intrinsically protecting their shared password. In fact, the Secure Remote Password (SRP) protocol [Wu98] has been incorporated into previous versions of TLS and standardized in the form of RFC5054 [TWMP07]. Specifically, the SRP protocol was

made available as a cipher suite for the key negotiation phase i.e. the TLS handshake. Similarly, the IETF is currently considering the adoption of SPAKE2 in TLS 1.3 [BF18], in particular in the TLS handshake, for scenarios where authentication is made using pre-shared keys available between the Client and Server, i.e. a password in this particular case.

In the recently approved TLS 1.3, it has explicitly been a design goal to provide *forward secrecy* for the session keys used to construct the TLS channel. Explicitly, *static* RSA and Diffie-Hellman cipher suites were removed to favor public-key based key-exchange mechanisms that guarantee forward secrecy. Therefore, formally proving that SPAKE2 satisfies some significant notion of forward secrecy would increase its possibilities of acceptance into TLS 1.3.

5.1.3 Forward Secrecy

Forward secrecy is a security property that may be required in both AKE and PAKE protocols. Over the last two decades, we have witnessed how the perception of forward secrecy has changed: it evolved from being merely an optional security property – which was extra to the fundamental properties of user authentication and secrecy of the session keys – into a highly desired property which frequently is an explicit requirement.

Roughly speaking, forward secrecy ensures the protection of session keys even if the long-term secret of the participants gets later compromised [DVOW92]. In practice, there are numerous ways in which the long-term secret material could get compromised: i) the password file at the server could get compromised, ii) via phishing attacks a client could reveal his password to some malicious entity. Furthermore, users tend to choose similar passwords for different servers, say a user U might choose similar looking or even the same password π for servers $S_1 \cdots S_n$. Then, it is fair to assume that compromise of the password $\pi[U, S_i]$ might also compromise $\pi[U, S_j]$. Therefore, forward secrecy is a highly advisable property which has been explicitly a design goal in relevant AKE and PAKE protocols [LMQ$^+$03, Kra05, HR10, Mac02a] and more recently in TLS 1.3 [Res18].[1]

The notion of forward secrecy appeared first in [DVOW92] and was later formalized in [Sho99, CK01, Kra05, LLM07] for AKE and in [BPR00, KOY02] for PAKE protocols. It is indisputable that this formalization enhanced the understanding of forward secrecy by identifying: i) means in which a principal can get compromised and ii) the information leaked in such a case. It is usually assumed that the adversary has full control of the network. However, the notion of *weak forward secrecy* (wFS) protects session keys, which may be established before or after the compromise of long-term key material, but only for those sessions created *without the active participation of the attacker* [BPR00, Kra05].[2] On the other hand, perfect forward secrecy (PFS) protects all session keys established *before* the compromise of the long-term material, even if created by the active intervention of the adversary.

[1] However, in TLS 1.3, there still remain some configurations that do not satisfy forward secrecy, for instance in pre-shared key mode.

[2] To the best of our knowledge, the notion of wFS was first mentioned in [BPR00] and it was later formalized in [Kra05], when Krawczyk acknowledged that the well-known HMQV protocol, a one-round AKE protocol, cannot satisfy PFS but only wFS.

It is generally accepted that PFS is difficult to satisfy in one-round protocols – which inevitably guarantee only *implicit authentication*. For instance, Krawczyk [Kra05] states that PFS cannot be satisfied by one-round protocols using *public-key* encryption as authentication mechanism. Therefore Krawczyk proposed the notion of *weak Forward Secrecy* (wFS) as an attempt to satisfy some notion of security in the HMQV protocol [Kra05] when the long-term material is compromised – but only for those sessions established without the active participation of the adversary. Krawczyk's argument was later generalized in [GKR10] as "PFS is impossible for one-round protocols when the information transmitted between the parties is computed without access to the parties's long-term secrets". In separate work, Bellare et al. [BPR00] followed by Boyd and Nieto [BN11] demonstrate that 2-flow protocols cannot satisfy PFS if the *internal state* and the long term key is leaked as a result of a corruption.

It seems that the arguments in [BPR00, Kra05] have lead researchers to mistakenly assume that PFS cannot be satisfied for "any" one-round key-exchange protocol. For instance, when LaMacchia et al. introduced the so-called eCK model [LLM07], the authors recalled Krawczyk's argument as "no two-round AKE protocol can achieve PFS", while in fact, Krawczyk's argument refers only to one-round AKEs where authentication is achieved by encrypting the protocol messages under the public-key of the recipient.

PFS and key-confirmation: The authors in [BPR00, Mac02a, Sho99] demonstrated that PFS can be satisfied when *explicit authentication* is added to protocols that initially satisfy only wFS. The idea is the following: Suppose P is a 2-flow PAKE protocol satisfying only implicit authentication. The adversary sends the first message to Bob masquerading as Alice, Bob computes the session key, sends back the second message and finishes his protocol execution. Then the adversary waits for the leakage of the long-term key and that could *possibly* help her to compute the same session key as Bob. For this scenario, the notion of PFS requires the adversary not to learn Bob's session key, which can be easily avoided by requiring key-confirmation, since then Bob will not accept the session key before he authenticates his communication partner.

5.1.4 Our contribution

We propose a new version of SPAKE2 which we name PFS-SPAKE2. This is essentially SPAKE2 but incorporating key-confirmation codes inspired by the work of [Mac02a]. This well-known approach allowed us to meet the PFS requirement in a provably secure way, even in the case of active adversaries, making it a suitable candidate for standardization and adoption in the TLS 1.3 protocol. In addition, we prove that the original SPAKE2 satisfies weak forward secrecy.

5.2 The SPAKE2 Protocol

SPAKE2 is an efficient one-round PAKE protocol by Abdalla and Pointcheval [AP05b] and proven secure in the Find-then-Guess model of Bellare, Pointcheval and Rogaway [BPR00].

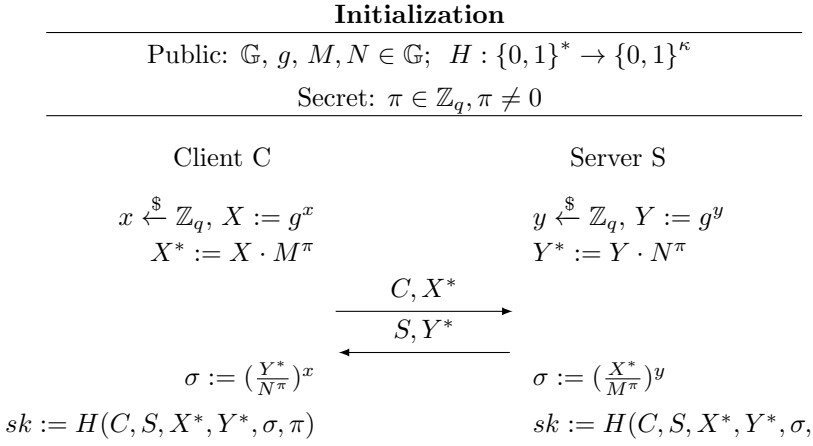

Figure 5.1: SPAKE2 protocol.

5.2.1 Protocol description

In Figure 5.1 we recall the technical description of the SPAKE2 protocol.

The protocol is a conversation between two principals, usually a Client and a Server, who wish to agree on a high-entropy session key. Before the protocol is run, there is an initialization phase where a password π is assigned to each pair (C, S) of Client and Server. This constitutes the only secret that protocol participants are required to share in advance. Additionally, before any protocol execution, public parameters must be published and accessible to protocol participants. These parameters include the description of group $\mathbb{G}$, hash function H and two common reference strings $M, N \in \mathbb{G}$. The secret password of the participants is encoded in $\mathbb{Z}_q$, which can be easily achieved by standard encoding techniques: an 8-character password can be represented in 56 bits while an element in $\mathbb{Z}_q$ is usually 160-bits long.

When the protocol starts, the client outputs a group element X^* and sends it to the server. The term X^* is derived from the product of two group elements, $X \in_R \mathbb{G}$ and $M^\pi \in \mathbb{G}$, this construction guarantees that for an observer, the term X^* looks like a random group element. In *parallel*, the server outputs a group element Y^*, constructed in a similarly to the client's X^* term, and sends it to the client. Upon reception of the corresponding Y^* and X^*, C and S proceed to compute σ, which will constitute the *shared secret of the session*, and subsequently to compute the session key sk. The session keys will match for C and S provided that they share the same password π.

We stress that SPAKE2 has optimal round complexity [KV11]. It is a one-round protocol as each party sends only one message and they could be sent *simultaneously*. More concretely, the server computes his Y^* term without waiting for the reception of the client's X^* term as they are independent. The same reasoning applies when the client outputs his X^* term. Nevertheless, in practice, both parties need some type of signal that triggers the protocol execution; usually for the client it an user instruction while for the server the signal is typically the reception of the

client's message.

5.2.2 Security of SPAKE2

The SPAKE2 protocol is already proven secure in the FtG model [AP05b]. However, the original proof does not provide any security on previously established session keys in case of password leakage. Favorably, as the following theorem states, the SPAKE2 protocol satisfies *weak* forward secrecy in the FtG model (see Definition 4.2 from Chapter 4) assuming the CDH and CSDH problems are hard in $\mathbb{G}$.

Theorem 5.1. *(Security in the wFS-FtG Model). Let P be the protocol specified in Figure 5.1 instantiated in group $\mathbb{G}$ and with passwords uniformly distributed over dictionary D. Let $\mathcal{A}$ be an adversary that runs in time t polynomial in κ, makes at most n_{ex}, n_{se}, n_{ro} queries of type Execute, Send and random oracle. For all such adversaries $\mathcal{A}$:*

$$\mathrm{Adv}_P^{\text{wFS-FtG}}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + \mathcal{O}\left(\frac{(n_{se}+n_{ex})(n_{se}+n_{ex}+n_{ro})}{q} + n_{ro}\cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}^{\mathcal{A}}) + n_{se}n_{ro}\cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\hat{\mathcal{B}}^{\mathcal{A}}) + (n_{ro})^2 \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\tilde{\mathcal{B}}^{\mathcal{A}})\right),$$

where $\mathcal{B}^{\mathcal{A}}$, $\hat{\mathcal{B}}^{\mathcal{A}}$ are CDH-solvers and $\tilde{\mathcal{B}}^{\mathcal{A}}$ is a CSDH-solver algorithm, running in time $t' = \mathcal{O}(t + (n_{se}+n_{ex}+n_{ro})\cdot t_{exp})$ and where t_{exp} is the time for an exponentiation in $\mathbb{G}$.

We prove Theorem 5.1 using the sequence of games approach described by Shoup in [Sho04]: We introduce a sequence of protocols $\mathrm{P}_0 \ldots \mathrm{P}_7$, where P_0 is the original protocol and P_7 is a protocol that allows only online dictionary attacks, while showing that the transition between P_i and P_{i+1} is indistinguishable for a computationally bounded adversary. In Figure 5.2, we provide an overview of the required games for the security proof of SPAKE2.

We proceed to detail the previously described games and show that P_i and P_{i+1} are computationally indistinguishable. In particular, we show that the success probability of $\mathcal{A}$ in G_i is at most negligible more than that of G_{i+1}. To simplify the notation, we write $\mathrm{Succ}_{P_i}^{\text{FtG}}$ instead of $\mathrm{Succ}_{P_i}^{\text{wFS-FtG}}$ to denote the success probability of $\mathcal{A}$ winning in game G_i.

Due to similarities with the PPK security proof, we borrow from [Mac02a] the proof structure and necessary nomenclature that will allow us to prove the security of the protocol by sequence of games. We say "in a CLIENT ACTION k to Π_C^i" to refer to "in a Send query directed to the client instance Π_C^i that results in CLIENT ACTION k procedure being executed" and "in a SERVER ACTION k" to refer to "in a Send query directed to the server instance Π_S^j that results in SERVER ACTION k procedure being executed". A client instance Π_C^i is *paired with* server instance Π_S^j if there was a CLIENT ACTION 0 to Π_C^i with output $\langle C, X^*\rangle$, a SERVER ACTION 1 to Π_S^j with input $\langle C, X^*\rangle$ and output $\langle S, Y^*\rangle$ and a CLIENT ACTION 1 to Π_C^i with input $\langle S, Y^*\rangle$. A server instance Π_S^j is *paired with* client instance Π_C^i if there was a CLIENT ACTION 0 to Π_C^i with output $\langle C, X^*\rangle$ and a SERVER ACTION 1 to Π_S^j with input $\langle C, X^*\rangle$ and output $\langle S, Y^*\rangle$.

Game $\mathbf{G_0}$: Execution of original protocol.
This game corresponds to the original security experiment.

Game $\mathbf{G_1}$: Force uniqueness of honest instances.
An instance outputs random values X^* and Y^* which hopefully have not been previously observed, otherwise it draws random values again until the previous condition is satisfied.

Game $\mathbf{G_2}$: Simulation without using the password.
From this game, we give credit to $\mathcal{A}$ on winning the experiment only if she asks the random oracle to compute the session key. Additionally, honest instances' response to Send and Execute queries is independent of the user's password unless some bad event happens.

Game $\mathbf{G_3}$: Randomized session keys for passive adversaries.
Consider a session that is honestly established via an Execute query. From this game, $H(\cdot)$ queries – targeting a session established via an Execute query – are answered with a random string independently of any previously observed messages.

Game $\mathbf{G_4}$: Check for successful password guesses.
Before any Corrupt query, if the adversary guesses the correct password and successfully impersonates either a client or server instance, then the protocol stops and the adversary automatically wins.

Game $\mathbf{G_5}$: Randomized session keys for paired instances.
Consider a session that is established by the adversary faithfully forwarding messages between the involved instances. From this game, $H(\cdot)$ queries – targeting such honestly established session – are answered with a random string independently of any previously observed messages.

Game $\mathbf{G_6}$: Prevent testing more than one password per client or server instance.
In this game we restrict the adversary from testing two different passwords, say π_1 and π_2, during active attack against a server or client instance.

Game $\mathbf{G_7}$: Internal password oracle.
This game considers an ideal protocol P_7 that admits only online dictionary attacks.

Figure 5.2: Sequence of games for proving SPAKE2 security.

The following events correspond to an adversary $\mathcal{A}$ testing password candidates in client and server instances. We hope the relation between the games described in Figure 5.2 with their matching event becomes clear, in particular, G_3 corresponds to **testexecpw**, G_4 to **correctpw**, G_5 to **pairedpwguess**, and G_6 to **doublepwclient** and **doublepwserver** event. We recall that we usually write $DH(X, Y)$ to denote g^{xy}, where $X = g^x, Y = g^y$ and $g^{xy} \in \mathbb{G}$.

- **testpw**(C, i, S, π): Adversary $\mathcal{A}$ makes i) an $H(C, S, X^*, Y^*, \sigma, \pi)$ query, ii) a CLIENT ACTION 0 to Π^i_C with output $\langle C, X^* \rangle$ and iii) a CLIENT ACTION 1 to Π^i_C with input $\langle S, Y^* \rangle$, where $\sigma = DH(X^*/M^\pi, Y^*/N^\pi)$. The associated value to this event is either the sk^i_C or the output of the $H(\cdot)$ query, whichever is set first.
- **testpw**(S, j, C, π): $\mathcal{A}$ makes an $H(C, S, X^*, Y^*, \sigma, \pi)$ query and a SERVER ACTION 1 to Π^j_S with input $\langle C, X^* \rangle$ and output $\langle S, Y^* \rangle$, where $\sigma = DH(X^*/M^\pi,\ Y^*/N^\pi)$. The associated value to this event is either the sk^j_S or the output of the $H(\cdot)$ query, whichever is set first.
- **testexecpw**(C, i, S, j, π): $\mathcal{A}$ makes both i) an $H(C, S, X^*, Y^*, \sigma, \pi)$ query, where $\sigma = DH(X^*/M^\pi,\ Y^*/N^\pi)$ and ii) an $\mathsf{Execute}(C, i, S, j)$ which produces X^*, Y^*. The associated value to this event is the established sk.
- **pairedpwguess**: A **testpw**(S, j, C, π_c) event occurs for a server instance Π^j_S that is *paired* with a client instance Π^i_C. The associated value to this event is the established sk.
- **doublepwclient**: Before any $\mathsf{Corrupt}$ query, **testpw**(C, i, S, π_1) and **testpw**(C, i, S, π_2) events both occur, for some C, i, S, π_1 and π_2, where $\pi_1 \neq \pi_2$.
- **doublepwserver**: Before any $\mathsf{Corrupt}$ query, both **testpw**(S, j, C, π_1) and **testpw**(S, j, C, π_2) occurs, for some S, j, C, π_1 and π_2, where $\pi_1 \neq \pi_2$.
- **correctpw**: Before any $\mathsf{Corrupt}$ query, either a **testpw**(C, i, S, π_c) or a **testpw**(S, j, C, π_c) event occurs, for some C, i, S, j, where π_c is the correct password.

Throughout the proof, unless the contrary is explicitly stated, the challenger – sometimes referred to as the simulator – sets the CRS as follows as follows $M := g^m$, $N := g^n$, where $M, N \in \mathbb{G}$ and $(m, n) \xleftarrow{\$} \mathbb{Z}_q^2$, i.e. we assume the simulator knows the discrete logarithm of the CRS to base g. We are now ready to detail the security proof of the SPAKE2 protocol.

Game G_0 : Execution of original protocol.
As part of the security experiment, the challenger *simulates* honest users and instances to the adversary. In this particular game, the simulated instances behave according to the original protocol description.

This game is identical to the original security experiment and we assume the random oracle model. The challenger *simulates* to the adversary i) honest instances and ii) the hash function $H(\cdot)$, which $\mathcal{A}$ is given oracle access to. We maintain a list Δ_{ro} in order to answer $H(\cdot)$ oracle queries consistently. See Figure 5.3.

Rule H: For a hash query $H(q)$, such that q is found in Δ_{ro}, look for the (q, r) record and answer with r. Otherwise, define r according to the following rule and add the record (q, r) to Δ_{ro}:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^{\kappa}$.

Figure 5.3: Simulation of $H(\cdot)$ random oracle queries.

Game $\mathbf{G_1}$: Force uniqueness of honest instances.
The challenger *simulates* honest instances to the adversary and generates the X^* and Y^* terms according to the description of the protocol. Let G_1 be as G_0, except that if any of the X^* or Y^* collide with previously generated ones, then the challenger draws random values again until he arrives at X^* and Y^* that have not been seen before. It is easy to see that the probability of collision is bounded by the birthday paradox, then:

$$\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_0}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_1}(\mathcal{A})\,\right] + \mathcal{O}\left(\frac{(n_{se}+n_{ex})(n_{se}+n_{ex}+n_{ro})}{q}\right)$$

P_1 guarantees that for any client instance Π^i_C that accepts with some session identifier *sid*, there exists *at most* one server instance Π^j_S that accepts with the same *sid*. The implication is that P_1 rules out the possibility of an instance Π^i_C sharing the same *sid* and *sk* with two instances Π^j_S and $\Pi^{j'}_S$, which would allow the adversary to trivially win the security experiment since by the second requirement of the partnering definition would not be satisfied: $\mathcal{A}$ could fairly make a Reveal query to Π^j_S or $\Pi^{j'}_S$ and win the game. A similar reasoning applies when considering a server instance Π^j_S accepting with some *sid*.

Game $\mathbf{G_2}$: Simulation without password.
The idea of this game is that, unless some *bad event* occurs, the protocol can be simulated without requiring any password. In particular, there is an internal oracle H^* – not available to the adversary – which instances use to compute their session keys. See Figure 5.4. A bad event is either a $\mathbf{testpw}(C, i, S, \pi_c)$, a $\mathbf{testpw}(S, j, C, \pi_c)$ or a $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event. For simplicity, we can think of the first two events as the scenario where $\mathcal{A}$ correctly guesses the password π_c and derives the correct session key by successfully impersonating a principal in an online attack. The third event corresponds to the case where $\mathcal{A}$ knows π_c and is able to compute a session key that was established via Execute queries (more details in G_3).

Let P_2 be a protocol identical to P_1, except that, unless backpatching is required, honest instances compute the session key by making a query of the form $\mathrm{H}^*(C, S, X^*, Y^*)$, i.e. the resulting session key is independent of the password π_c and the shared secret σ. Whenever $\mathcal{A}$ asks a $H(\cdot)$ oracle query, the simulator checks if any of the previously mentioned bad events occurs, and in such a case, it does *backpatching* to provide consistent views to the adversary.

- In an $\mathsf{Execute}(C, i, S, j)$ query set $X^* := g^{\tau[C,i]}$ and $Y^* := g^{\tau[S,j]}$, with $\tau[\cdot] \xleftarrow{\$} \mathbb{Z}_q$ and

Rule H*: For a hash query $\mathrm{H}^*(q)$, such that q is found in Δ^*_{ro}, look for the (q, r) reccord and answer with r. Otherwise define r according to the following rule and add the record (q, r) to Δ^*_{ro}:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^\kappa$.

Figure 5.4: Simulation of internal oracle $\mathrm{H}^*(\cdot)$.

$sk^i_C \leftarrow sk^j_S := H^*(C, S, X^*, Y^*)$.

- In a CLIENT ACTION 0 to Π^i_C, set $X^* = g^{\tau[C,i]}$, for $\tau[C, i] \xleftarrow{\$} \mathbb{Z}_q$.
- In a SERVER ACTION 1 to Π^j_S, set $Y^* = g^{\tau[S,j]}$, where $\tau[S, j] \xleftarrow{\$} \mathbb{Z}_q$ and $sk^j_S := H^*(C, S, X^*, Y^*)$.
- In a CLIENT ACTION 1 to Π^i_C proceed as follows:
  - If Π^i_C is *paired* with Π^j_S then set $sk^i_C \leftarrow sk^j_S$.
  - Else if this query triggers a **testpw**(C, i, S, π_c) event, then set sk^i_C to the associated value of the event **testpw**$(C, i, S, \pi_c,)$.
  - Else set $sk^i_C := H^*(C, S, X^*, Y^*)$.
- In an $H(C, S, X^*, Y^*, \sigma, \pi)$ query made by the adversary $\mathcal{A}$, in case it triggers a **testpw** (C, i, S, π_c) for some unpaired Π^i_C, or if it triggers **testpw**(S, j, C, π_c) for some unpaired Π^j_S, or if it triggers **testexecpw**(C, i, S, j, π_c) event for some Π^i_C and Π^j_S instances, or if there is some Π^j_S paired with some Π^i_C and **pairedpwguess**(S, j, C, π_c) is triggered, then answer with the associated value of the corresponding event, i.e. the already established session key via the internal oracle. Otherwise answer according to **Rule H**.

Claim 5.2. *For all adversaries $\mathcal{A}$,*

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_1}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_2}(\mathcal{A})\right].$$

Proof. It follows from the construction that P_2 and P_1 are perfectly indistinguishable. We make the following observations:

1. In a SERVER ACTION 1 to Π^j_S, in P_1 the sk^j_S is generated at random and independent of any previous messages since the $H(\cdot)$ query that determines the session key is new, while in P_2 the sk^j_S is also generated at random and independent of previous messages using the internal oracle $H^*(\cdot)$.
2. In a CLIENT ACTION 1 to Π^i_C:

- If Π_C^i is paired with Π_S^j, then in P_1, $sk_C^i = sk_S^j$ as both instances query the random oracle $H(\cdot)$ with the same arguments to establish the session key, while in P_2, sk_C^i is copied from sk_S^j.
- If Π_C^i is not paired with a server instance, then either i) a $\mathbf{testpw}(C, i, S, \pi_c)$ event occurs, in which case sk_C^i is set identically in P_1 and P_2 using the $H(\cdot)$ oracle in both cases, or ii) in P_1, sk_C^i is independent from previous messages, and in P_2, sk_C^i is also independent but set using the internal oracle H^*.

3. For all $H(\cdot)$ queries, either:

- The queries are new, in which case they are answered according to Rule H.
- Either $\mathbf{testpw}(C, i, S, \pi_c)$, $\mathbf{testpw}(S, j, C, \pi_c)$ or $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event occurs, in which case the $H(\cdot)$ are answered according to the associated value of the event i.e. the previously established session keys which in fact were established using $H^*(\cdot)$.

Thus, in G_2, the simulator provides identical views to the adversary to those from G_1. Note that the simulator not only knows π_c but also the discrete logarithm of M, N, X^* and Y^*, and with this information he can determine whenever one of the previously mentioned *bad events* occurs. □

Game G_3 : Randomized session keys for passive adversaries.
Consider an adversary $\mathcal{A}$ that eventually asks an $\mathsf{Execute}(C, i, S, j)$ query, resulting in the X^* and Y^* messages being exchanged and sk established at Π_C^i and Π_S^j instances. From this game, whenever $\mathcal{A}$ makes a $H(C, S, X^*, Y^*, \sigma, \pi)$ oracle query, the simulator does not check whether the $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event occurs (no need to do backpatching as in G_2). It simply answers the $H(\cdot)$ query with Rule H from Figure 5.3, i.e. a random string independent of the established sk and of any previously exchange messages.

Let F_2 and F_3 denote the event that for some C, S, i, j, the $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event occurs in P_2 and P_3 respectively.

Claim 5.3. *For all adversaries $\mathcal{A}$,* $\left|\Pr\left[\,\mathrm{Succ}_{P_2}^{\mathrm{FtG}}(\mathcal{A})\,\right] - \Pr\left[\,\mathrm{Succ}_{P_3}^{\mathrm{FtG}}(\mathcal{A})\,\right]\right| \leq \Pr\left[\,F_3\,\right]$.

Proof. P_2 and P_3 behave identically, unless F_2, respectively F_3, occurs. The main observation is that the events F_2 and F_3 are triggered. Then, $\Pr\left[\,F_2\,\right] = \Pr\left[\,F_3\,\right]$ and to conclude the proof we apply Shoup's Difference Lemma [Sho04]. □

Next we will demonstrate that the probability of $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event happening is bounded by the CDH assumption.

Claim 5.4. *For all PPT adversaries $\mathcal{A}$ running in time t, there exists a CDH-solver $\mathcal{B}^{\mathcal{A}}$ with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\,\mathrm{Succ}_{P_2}^{\mathrm{FtG}}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}_{P_3}^{\mathrm{FtG}}(\mathcal{A})\,\right] + n_{ro} \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}^{\mathcal{A}}).$$

Proof. Let ϵ be the probability that F_3 occurs in P_3. We build an adversary $\mathcal{B}^{\mathcal{A}}$ whose goal is to solve the CDH problem using adversary $\mathcal{A}$ as a subroutine and whose success probability is at least ϵ/n_{ro}. On input $(A = g^{\alpha}, B = g^{\beta})$, $\mathcal{B}^{\mathcal{A}}$ *simulates* P_3 to $\mathcal{A}$ with the following changes:

1. For every $\mathsf{Execute}(C, i, S, j)$ query made by $\mathcal{A}$, set $X^* = A \cdot g^{r_{C,i}}$, $Y^* = B \cdot g^{r_{S,j}}$ and $sk_C^i \leftarrow sk_S^j := H^*(C, S, X^*, Y^*)$, where $r_{C,i}, r_{S,j} \xleftarrow{\$} \mathbb{Z}_q$ are known to the simulator.

2. For every $H(C, S, X^*, Y^*, \sigma, \pi_c)$ query, where the X^* and Y^* terms are generated via an $\mathsf{Execute}(C, i, S, j)$ query and π_c corresponds to the password shared between C and S, then add γ to the set S-DH, where:

$$\gamma = \left(\frac{\sigma \cdot B^{m\pi_c - r_{C,i}} \cdot A^{n\pi_c - r_{S,j}}}{DH(g^{r_{C,i} - m\pi_c}, g^{r_{S,j} - n\pi_c})} \right)$$

3. When $\mathcal{A}$ finishes, the set S-DH contains at most n_{ro} elements, where each item is a possible solution to $\mathrm{DH}(A, B)$. Then $\mathcal{B}^{\mathcal{A}}$ picks $\gamma \xleftarrow{\$}$ S-DH as its output and hopes that it is the correct one.

Conditional on F_3 occurring, $\mathcal{B}^{\mathcal{A}}$ has probability at least $1/n_{ro}$ to find the correct value of $\mathrm{DH}(A, B)$. Then, $\epsilon \leq n_{ro} \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\mathcal{B}^{\mathcal{A}})$. □

We make the assumption that even if $\mathcal{A}$ distinguishes P_2 from P_3, she still runs in time t. One can observe that G_3 guarantees forward secrecy for those sessions where $\mathcal{A}$ act as eavesdropper. It implies that an adversary who knows the password but passively observes the execution of the protocol, cannot derive the correct session key, even if she manages to corrupt some users.

Game $\mathbf{G_4}$: Check for successful password guesses.
The purpose of this game is to acknowledge successful online dictionary attacks, in such a case, the protocol stops and the adversary automatically wins. Let P_4 be defined exactly as P_3, except that if the **correctpw** event occurs, then the protocol stops and the adversary wins. It follows immediately that:

$$\mathrm{Succ}^{\mathrm{FtG}}_{P_3}(\mathcal{A}) \leq \mathrm{Succ}^{\mathrm{FtG}}_{P_4}(\mathcal{A})$$

Notice that from this game, unless the **correctpw** event occurs, session keys for unpaired instances will be established according to internal oracle H^*, i.e. independent of both the previous transcript of the protocol and $H(\cdot)$ queries. Following the same reasoning, $H(\cdot)$ queries for *unpaired* instances, will always be answered according to Rule H, i.e. with a fresh value and independently of previously made $H^*(\cdot)$ queries (because P_5 stops in case the **correctpw** event occurs). Obviously, this explanation holds before somebody gets corrupted.

Game $\mathbf{G_5}$: Randomized session keys for paired instances.
This game captures a similar idea to that of G_3. The difference is that here we consider a session that is honestly established via Send queries, by $\mathcal{A}$ faithfully forwarding messages between the

involved instances. The underlying notion is that for such sessions, $\mathcal{A}$ cannot distinguish the established session key from a random string.

Let P_5 be identical to P_4 except that, in a $H(C, S, X^*, Y^*, \sigma, \pi_c)$ query, in the case when X^*, Y^* corresponds to some Π_S^j *paired* with some Π_C^i, the simulator does not verify if **pairedpwguess** occurs. Let F_4 and F_5 denote the event that, during the execution of G_4, respectively G_5, the following conditions are met: there are a client instance Π_C^i outputting X^*, a server instance receiving X^* and outputting Y^*, and, before any successful online dictionary attack to Π_C^i, there is a $H(C, S, X^*, Y^*, DH(X^* \cdot M^{-\pi_c}, Y^* \cdot N^{-\pi_c}), \pi_c)$ query. Then:

Claim 5.5. *For all adversaries $\mathcal{A}$, $\left|\Pr\left[\,\mathrm{Succ}_{P_4}^{\mathrm{FtG}}(\mathcal{A})\,\right] - \Pr\left[\,\mathrm{Succ}_{P_5}^{\mathrm{FtG}}(\mathcal{A})\,\right]\right| \leq \Pr\left[\,F_5\,\right]$.*

Proof. Identical to that of Claim 5.3. □

Claim 5.6. *For all PPT adversaries $\mathcal{A}$ running in time t, there exists a CDH-solver $\hat{\mathcal{B}}^{\mathcal{A}}$ with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\,\mathrm{Succ}_{P_4}^{\mathrm{FtG}}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}_{P_5}^{\mathrm{FtG}}(\mathcal{A})\,\right] + n_{se} \cdot n_{ro} \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\hat{\mathcal{B}}^{\mathcal{A}}).$$

Proof. Let ϵ be the probability of F_5 happening in P_5. We build a CDH-solver algorithm $\hat{\mathcal{B}}^{\mathcal{A}}$, which gets as input $(A, B) \in \mathbb{G}^2$ and outputs $\mathrm{DH}(A, B)$ with probability at least $\epsilon/(n_{ro} \cdot n_{se})$. When the simulation starts, $\hat{\mathcal{B}}^{\mathcal{A}}$ sets $M = g^m$ and $N = g^n$. Then she chooses at random a client instance Π_C^d, which she hopes will participate in the triggering of event F_5, and simulates P_5 to $\mathcal{A}$ with the following changes:

1. In a CLIENT ACTION 0 query to Π_C^d, set $X^* := A$.
2. In a SERVER ACTION 1 query to Π_S^j with input X^*, where there was a previous CLIENT ACTION 0 query to Π_C^d with output X^*, set $Y^* = B \cdot g^{r_{S,j}}$, with $r_{S,j} \xleftarrow{\$} \mathbb{Z}_q$, and set $sk_S^j = H^*(C, S, X^*, Y^*)$.
3. Test for $\mathbf{testpw}(C, d, S, \pi_C)$ are not made.
4. In CLIENT ACTION 1 query to Π_C^d with input Y^*, if it is paired with Π_S^j, set $sk_C^d \leftarrow sk_S^j$. Else set $sk_C^d = H^*(C, S, X^*, Y^*)$.
5. In a $H(C, S, X^*, Y^*, \sigma, \pi_C)$ query made by $\mathcal{A}$, where i) X^* was generated by the Π_C^d instance, ii) Y^* was generated by an Π_S^j instance and iii) Π_C^d is paired with Π_S^j, then answer with Rule H.
6. When $\mathcal{A}$ finishes, for every $H(C, S, X^*, Y^*, \sigma, \pi_C)$ query she made, where i) X^* was generated by the Π_C^d instance, ii) Y^* was generated by an Π_S^j instance and iii) Π_S^j is paired with Π_C^d, add γ to the set S-DH, where:

$$\gamma = \frac{\sigma \cdot A^{n\pi_c - r} \cdot B^{m\pi_c}}{DH(g^{m\pi_c}, g^{n\pi_c - r})}. \tag{5.1}$$

7. When $\mathcal{A}$ finishes, the set S-DH contains at most n_{ro} elements, where each item a possible solution to $\mathrm{DH}(A, B)$. Then $\hat{\mathcal{B}}^{\mathcal{A}}$ picks $\gamma \xleftarrow{\$}$ S-DH as its output and hopes that it is the correct one.

We use Shoup's convention, and think of G_6 and the simulation as occurring on the same probability space. Then, conditional on F_5, $\hat{\mathcal{B}}^{\mathcal{A}}$ finds the correct value of $\mathrm{DH}(A, B)$ with probability at least $1/n_{se}n_{ro}$. Then, $\mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\hat{\mathcal{B}}^{\mathcal{A}}) \geq \Pr\left[\,(F_5)\,\right]/n_{se}n_{ro}$. □

Game $\mathbf{G_6}$: Prevent testing more than one password per instance.
In PAKE protocols, the security intuition asks to limit the number of passwords that an adversary may test per session. Ideally, she should be limited to verify *at most* one guess of a password per session in an *active attack*. In this game we consider the aforementioned requirement.

Let P_6 be a protocol identical to P_5, except that if either **doublepwserver** or **doublepwclient** event occurs, the protocol stops and the adversary fails. Let $F_5 =$ **doublepwserver** $\cup$ **doublepwclient** occurring in P_5 and let F_6 be defined similarly but for P_6. It is easy to verify by construction that $\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A}) \wedge \neg F_5 \Leftrightarrow \mathrm{Succ}^{\mathrm{FtG}}_{P_6}(\mathcal{A}) \wedge \neg F_6$.

Let P_6 be a protocol identical to P_5, except that if either the **doublepwserver** or **doublepwclient** event occurs, the protocol stops and the adversary fails. Obviously, it follows by construction that $\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A}) \wedge \neg F_5 \Leftrightarrow \mathrm{Succ}^{\mathrm{FtG}}_{P_6}(\mathcal{A}) \wedge \neg F_6$.

Claim 5.7. *For all adversaries $\mathcal{A}$,* $\left|\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right] - \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right]\right| \leq \Pr\left[\,F_6\,\right]$.

Proof. Identical to that of Claim 5.3. □

Now we have to demonstrate that $F_6 =$ **doublepwserver** $\cup$ **doublepwclient** can occur with only small probability. Following the reductionist approach, we first demonstrate that for all adversaries $\mathcal{A}$, the probability of the event **doublepwserver** occurring is bounded by the CSDH assumption (see Def. 2.15).

Claim 5.8. *For all PPT $\mathcal{A}$ running in time t, there exists a CSDH-solver $\tilde{\mathcal{B}}^{\mathcal{A}}$ running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_4}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right] + (n_{ro})^2 \cdot \mathrm{Adv}^{\mathtt{CSDH}}_{\mathbb{G}}(\tilde{\mathcal{B}}^{\mathcal{A}}),$$

Proof. Let ϵ be the probability that the *doublepwserver* event occurs in P_5. One can build a CSDH solver $\tilde{\mathcal{B}}^{\mathcal{A}}$ using adversary $\mathcal{A}$ as a subroutine and has success probability $\epsilon/(n_{ro})^2$. On input $A = g^{\alpha}$, $\tilde{\mathcal{B}}^{\mathcal{A}}$ simulates P_6 to $\mathcal{A}$ with the following changes:

1. Set $M \leftarrow g^{-\alpha}$ and $N \leftarrow g^{-\alpha - r}$, where $r \xleftarrow{\$} \mathbb{Z}_q$ is known to $\tilde{\mathcal{B}}^{\mathcal{A}}$.

2. On a SERVER ACTION 1 to Π^j_S with input $\langle C, X^* \rangle$, output $Y^* = g^y$, for $y \xleftarrow{\$} \mathbb{Z}_q$.

3. Checks for **correctpw** event are not made.

4. When $\mathcal{A}$ terminates, the simulator selects at random a pair of random oracle queries $H(C, S, X^*, Y^*, \sigma_1, \pi_1)$ and $H(C, S, X^*, Y^*, \sigma_2, \pi_2)$, where $\pi_1, \pi_2 \in \mathbb{Z}_q^*$ and $\pi_1 \neq \pi_2$, there was a SERVER ACTION 1 to Π_S^j with input $\langle C, X^* \rangle$ and output Y^*. Then, the simulator computes γ as possible solution to CSDH(A) as follows:

$$\gamma = \left(\sigma_1^{\pi_2} \cdot \sigma_2^{-\pi_1} \cdot (X^*)^{y(\pi_1 - \pi_2)}\right)^{\phi} \cdot A^{-r},$$

where ϕ is the multiplicative inverse of $\pi_1 \cdot \pi_2(\pi_1 - \pi_2)$ in $\mathbb{Z}_q^*$. □

We now demonstrate that for all adversaries $\mathcal{A}$, the probability of the event **doublepwclient** occurring is bounded by the CSDH assumption.

Claim 5.9. *For all PPT $\mathcal{A}$ running in time t, there exists a reduction from the CSDH problem with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\mathrm{Succ}_{P_5}^{\mathrm{FtG}}(\mathcal{A})\right] \leq \Pr\left[\mathrm{Succ}_{P_6}^{\mathrm{FtG}}(\mathcal{A})\right] + (n_{ro})^2 \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\tilde{\mathcal{B}}^{\mathcal{A}}),$$

where $\mathrm{Adv}_{\mathbb{G}}^{\mathtt{CSDH}}(\tilde{\mathcal{B}}^{\mathcal{A}})$ is the advantage of a CSDH-solver algorithm running in time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{ex})$.

Proof. Similar to that of Claim 5.8 □

Game $\mathbf{G_7}$: Internal password oracle.
In this game we estimate the probability of the **correctpw** event taking place, i.e. the probability of the adversary on guessing the correct password π_c during an active attack.

Let G_7 be identical to G_6, except that, in G_7, there is an internal password oracle $\mathcal{O}_{pw}$ that handles all passwords. In particular, $\mathcal{O}_{pw}$ is used to:

1. Generate all passwords during the initialization phase.
2. Answer Corrupt(U) queries.
3. Determine whether the **correctpw** event occurs.

On the initialization phase, the challenger uses $\mathcal{O}_{pw}$ to assign a password π to each pair of client C and server S, uniformly and at random from the password dictionary. When $\mathcal{A}$ makes a Corrupt(U) query, the challenger forwards it to $\mathcal{O}_{pw}$, which returns a single password π_c if U is client and a vector of passwords $\pi_s =< \pi_c >_{C \in \mathcal{C}}$ if U is the server. Finally, to determine if the **correctpw** occurs, the challenger first makes a TestPw(π, C) to $\mathcal{O}_{\pi}$ to determine if the adversary has guessed the password π_C of client C, it returns 1 if $\pi = \pi_c$ and 0 otherwise.

Claim 5.10. *For all adversaries $\mathcal{A}$,* $\Pr\left[\mathrm{Succ}_{P_6}^{\mathrm{FtG}}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}_{P_7}^{\mathrm{FtG}}(\mathcal{A})\right]$.

Proof. By construction, G_7 and G_6 are statistically indistinguishable. □

Claim 5.11. *For all adversaries $\mathcal{A}$,*

$$\Pr\left[\mathrm{Succ}_{P_7}^{\mathrm{FtG}}(\mathcal{A})\right] \leq \frac{1}{2} + \frac{n_{se}}{2 \cdot |D|}.$$

Proof.

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \mathrm{correctpw}\right] \cdot \Pr\left[\mathrm{correctpw}\right] + \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\mathrm{correctpw}\right] \cdot \Pr\left[\neg\mathrm{correctpw}\right] \quad (5.2)$$

From G_6, we know that $\mathcal{A}$ can test at most one password per session during an active attack, then it follows that $\Pr[\,\mathrm{correctpw}\,] \leq n_{se}/|D|$. We recall from P_4 that $\mathcal{A}$ automatically wins the experiment whenever the **correctpw** event occurs.

We now inspect the second term of Equation 5.2: if **correctpw** does not happen, then $\mathcal{A}$ wins the security experiment if she makes a Test query to a wFS-fresh instance Π^i_U and guesses the hidden bit b used to answer the Test query. We show that the view of $\mathcal{A}$ is independent of the bit b and thus $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\mathrm{correctpw}\right] = 1/2$:

1. The freshness condition prevents $\mathcal{A}$ from asking a Reveal query directed to Π^i_U or its partner (if it has one). Additionally, P_1 guarantees the uniqueness of the session identifier *sid* for honest instances. Otherwise, if two client instances accept with the same *sid*, then there is a trivial attack using Reveal queries. Then, the output of Reveal queries is independent of both sk^i_U and the bit b.

2. We look at the output of $\mathrm{H}(\cdot)$ queries. From G_3 and G_5, it follows that the output of $\mathrm{H}(\cdot)$ is independent from sk^i_U for passive adversaries and *paired* instances, respectively. Also, as consequence of G_5 and unless **correctpw** occurs, the session key of an *unpaired* instance is set using the internal oracle not accessible to $\mathcal{A}$ and independently of any $H(\cdot)$ query. It then follows that, provided that **correctpw** does not happen, then sk^i_U is independent of $\mathrm{H}(\cdot)$ queries.

Back to Equation 5.2 we obtain:

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] = \frac{1}{2} + \frac{n_{se}}{2 \cdot |D|}$$

Finally, we obtain:

$$\mathrm{Adv}^{\mathrm{FtG}}_{P_7} \leq \frac{n_{se}}{|D|}.$$

□

■

5.3 PFS-SPAKE2

Inspired by MacKenzie's work [Mac02a], we propose the adoption of key confirmation codes in the original SPAKE2 protocol [AP05b] to achieve PFS in a provable secure manner.

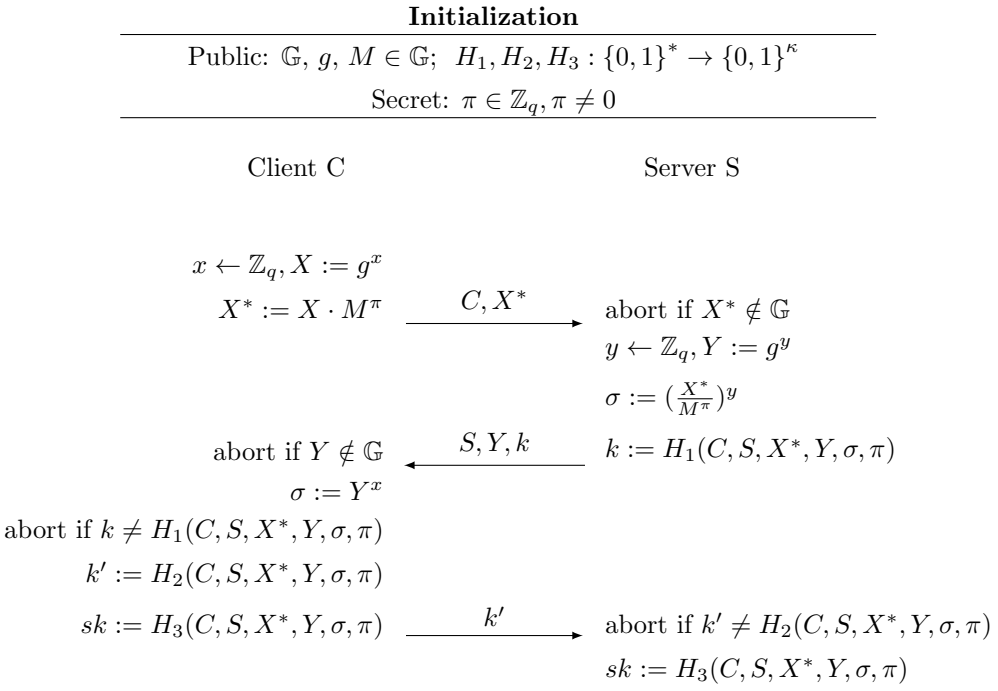

Figure 5.5: PFS-SPAKE2 protocol.

5.3.1 Protocol Description

In Figure 5.5 we provide the technical description of the proposed PFS-SPAKE2 protocol. Before the protocol is executed, public parameters must be choosen and published. These parameters include the description of group $\mathbb{G}$, hash functions H_1, H_2, H_3 and a single CRS M – which we require to be choosen at random from $\mathbb{G}$ and its discrete logarithm to be kept secret. These constraints on the CRS can be achieved either by having a third trusted party or by assuming a public source of randomness to publicly derive M. Our protocol is instantiated over group $\mathbb{G}$ of order q, a subgroup of $\mathbb{Z}_p^*$, where CDH assumption holds and p, q are safe prime numbers. The protocol requires that passwords are encoded in $\mathbb{Z}_q$.

The session keys and key-confirmation codes are computed a function of: the identity of users involved in the conversation, the conversation itself, the shared-secret agreed though the conversation i.e. the Diffie-Hellman term, and finally the shared-password. This is a standard yet powerful technique which first appeared in [BPR00] and was probably motivated by the fact that at the end of the protocol, each user should know i) who they share the session key with and ii) for which particular session. Then this technique permits, a user U to link the session key with the identity of its partner for some particular session. Moreover, it is relevant when proving the security in the random oracle model as it allows the simulator to observe which password the adversary is testing and for which particular session.

Table 5.1: Comparison with existing PAKEs for Client-Server scenarios.

Protocol	Communication[a]	Computation[b]	Rounds / Flows	Hardness Asm.[c]	Forward Secrecy	Key Confirm.
EKE [BM92, BPR00]	$2 \times \mathbb{G}$	4 exp., 2 enc.	1 / 2	CDH	wFS	No
PPK [Mac02a]	$2 \times \mathbb{G}$	6 exp.	1 / 2	CDH	-	No
PAK [Mac02a]	$2 \times \mathbb{G} + 2\kappa$	5 exp.	3 / 3	CDH	PFS	Yes
J-PAKE[d] [HR10]	$12 \times \mathbb{G} + 6 \times \mathbb{Z}_q$	28 exp.	2 / 4	DSDH	PFS	No
J-PAKE* [HR10]	$12 \times \mathbb{G} + 8 \times \mathbb{Z}_q$	28 exp.	3 / 6	DSDH	PFS	Yes
SPAKE2 [AP05b]	$2 \times \mathbb{G}$	6 exp.	1 / 2	CDH	wFS	No
PFS-SPAKE2	$2 \times \mathbb{G} + 2\kappa$	5 exp.	3 / 3	CDH	PFS	Yes

[a] Communication. $\mathbb{G}$ denotes a group element, $\mathbb{Z}_q$ a scalar and κ a κ-bit string.
[b] Computation. Exp. denotes an exponentiation in $\mathbb{G}$ and enc. an encryption and decryption operation.
[c] Hardness Assumption. DSDH stands for Decision Square Diffie-Hellman.
[d] J-PAKE* is simply J-PAKE but with an extra round for key confirmation.

Comparison to existing PAKEs. Usually the efficiency of a PAKE protocol is defined by i) the number of communication rounds until the protocol terminates, ii) the total number of messages exchanged and iii) the computational cost of the protocol. Compared to the original SPAKE2 (see Figure 5.1), the proposed PFS-SPAKE2 protocol benefits from *explicit* authentication and strong security guarantees for PFS. This improvement usually comes at the cost of increasing the number of rounds and messages flows required and unfortunately our protocol is not an exception [Mac02a, Kra05].

Additionally, the PFS-SPAKE2 protocol is slightly computationally more efficient than the original version. The reason is that it requires the client to compute only three exponentiations instead of four, particularly, in the new protocol the client does not need to compute $N^{\pi} \in \mathbb{G}$. One could argue that a similar reasoning can be applied to the server side too, however, we consider that this is not the case. For efficiency reasons, the server could legitimately store for every client the terms M^{π} and N^{π} together with the password in the database, then the previously mentioned terms do not need to be computed for every protocol execution. Nevertheless, the client is usually required to input his password for every protocol execution, so we cannot assume that the terms M^{π} and N^{π} are stored at the client side.

In Table 5.1 we summarize a comparison of the PFS-SPAKE2 to other relevant PAKE protocols with full security proofs (we consider the Client-Server setting for this purpose).[3] Notably J-PAKE satisfies PFS and requires only two communication rounds, though, it is computationally more expensive than the PFS-SPAKE2 as the former requires 28 exponentiations while the later only 5. Furthermore, J-PAKE with key confirmation requires the same number of commu-

[3] In this setting, the server usually stores some function $f(\cdot)$ of the password. In some protocols, this creates a difference in the computational cost between the client and the server. The reason is that the server may store $f(\pi)$ in a way that can be inserted in a protocol execution, while the client needs to compute $f(\pi)$ in every protocol run. This difference is noticeable in i) PPK, PAK and ii) SPAKE2 and PFS-SPAKE2, as $f(\cdot)$ requires hashing into groups for i) and group exponentiation for ii).

Game $\mathbf{G_0}$: Execution of original protocol.
Game $\mathbf{G_1}$: Force uniqueness of honest instances.
Game $\mathbf{G_2}$: Prevent lucky guesses on hash outputs.
Game $\mathbf{G_3}$: No need to backpatch $H_l(\cdot)$ queries against Execute queries.
Game $\mathbf{G_4}$: Check for successful password guesses.
Game $\mathbf{G_5}$: Randomized session keys for paired instances.
Game $\mathbf{G_6}$: Prevent testing more than one passwords per server instance.
Game $\mathbf{G_7}$: Internal password oracle.

Figure 5.6: Sequence of Games for PFS-SPAKE2

nication rounds as PFS-SPAKE2. Aditionally, PAK and PFS-SPAKE2 are similar in terms of efficiency, PFS and key-confirmation guarantees.

5.3.2 Security of PFS-SPAKE2

Next we prove the security of the PFS-SPAKE2 protocol in the FtG model and considering perfect forward secrecy (see Definition 4.1 from Chapter 4).

Theorem 5.12. *(Security in the PFS-FtG Model). Let P be the protocol specified in Fig. 5.5, instantiated in group $\mathbb{G}$ and with passwords uniformly distributed over dictionary D. Let $\mathcal{A}$ be an adversary that runs in time t polynomial in κ, makes at most n_{ex}, n_{se}, n_{ro} queries of type Execute, Send and random oracle. Then:*

$$\mathrm{Adv}_P^{\text{PFS-FtG}}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + \mathcal{O}\left(\frac{(n_{se}+n_{ex})(n_{se}+n_{ex}+n_{ro})}{q} + n_{ro}\cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}^{\mathcal{A}}) + n_{se}n_{ro}\cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\hat{\mathcal{B}}^{\mathcal{A}}) + (n_{ro})^2\cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\tilde{\mathcal{B}}^{\mathcal{A}})\right),$$

where $\mathcal{B}^{\mathcal{A}}$, $\hat{\mathcal{B}}^{\mathcal{A}}$ and $\tilde{\mathcal{B}}^{\mathcal{A}}$ are CDH-solver algorithms running in time $t' = \mathcal{O}(t + (n_{se}+n_{ex}+n_{ro})\cdot t_{exp})$, where t_{exp} is the time for an exponentiation in $\mathbb{G}$.

To prove the security of PFS-SPAKE2, we introduce a sequence of protocols $\mathrm{P}_0 \ldots \mathrm{P}_7$, where P_0 is the original protocol and P_7 allows only online dictionary attacks, while showing that the transition from P_i to P_{i+1} is computationally indistinguishable. Let $\mathrm{G_i}$ be the security game associated to $\mathrm{P_i}$. In Figure 5.6, we provide an overview of the required security games.

The security proof of PFS-SPAKE2 has some similarities with the PAK security proof. Then, we borrow from [Mac02a] the proof structure and necessary nomenclature for our proof, which we recall here:

We say "in a CLIENT ACTION k to Π_C^i" to refer to "in a Send query directed to the client instance Π_C^i that results in CLIENT ACTION k procedure being executed" and "in a SERVER ACTION k" to refer to "in a Send query directed to the server instance Π_S^j that results in SERVER ACTION k procedure being executed".

A client instance Π_C^i is *paired with* server instance Π_S^j, if there was a CLIENT ACTION 0 to Π_C^i with output $\langle C, X^* \rangle$, a SERVER ACTION 1 to Π_S^j with input $\langle C, X^* \rangle$ and output $\langle S, Y, k \rangle$ and a CLIENT ACTION 1 to Π_C^i with input $\langle S, Y, k \rangle$. A server instance Π_S^j is *paired with* client instance Π_C^i if there was a CLIENT ACTION 0 to Π_C^i with output $\langle C, X^* \rangle$ and a SERVER ACTION 1 to Π_S^j with input $\langle C, X^* \rangle$ and output $\langle S, Y, k \rangle$, additionally, if there is a SERVER ACTION 2 with input k', then there was a previous CLIENT ACTION 1 to Π_C^i with input $\langle S, Y, k \rangle$ and output k'.

Next we define the events that will allow us to prove the security of the protocol. As in [Mac02a], we make use of such events to formally define the *bad* scenarios which would increase the chances of $\mathcal{A}$ in winning the game, or, put it differently, the scenarios which we aim to show that can occur at most with negligible probability, for instance i) $\mathcal{A}$ testing more than one password per server instance, ii) $\mathcal{A}$ computing the session key for sessions where $\mathcal{A}$ was merely an eavesdropper. The events also help to identify whenever $\mathcal{A}$ has successfully impersonated an honest party.

- **testpw**(C, i, S, π, l): Adversary $\mathcal{A}$ makes i) an $H_l(C, S, X^*, Y, \sigma, \pi)$ query for some $l \in \{1, 2, 3\}$, ii) a CLIENT ACTION 0 to Π_C^i with output $\langle C, X^* \rangle$ and iii) a CLIENT ACTION 1 to Π_C^i with input $\langle S, Y, k \rangle$, where $X^* = X \cdot M^\pi$ and $\sigma = DH(X, Y)$. The associated value to this event is the output of the $H_l(\cdot)$ query, or the k, k', sk_C^i values, respectively for $l = 1, 2, 3$, whichever is set first.

- **testpw**(S, j, C, π, l): $\mathcal{A}$ makes an $H_l(C, S, X^*, Y, \sigma, \pi)$ for some $l \in \{1, 2, 3\}$ and a SERVER ACTION 1 to Π_S^j with input $\langle C, X^* \rangle$ and output $\langle S, Y, k \rangle$, where $X^* = X \cdot M^\pi$ and $\sigma = DH(X, Y)$. The associated value to this event is the output of the $H_l(\cdot)$ query, or the k, k', sk_S^j values, respectively for $l = 1, 2, 3$, whichever is set first.

- **testpw!**(C, i, S, π): A CLIENT ACTION 1 with input $\langle S, Y, k \rangle$, causes a **testpw** $(C, i, S, \pi, 2)$ event to occurs, with associated value k.

- **testexecpw**(C, i, S, j, π): $\mathcal{A}$ makes i) an $H_l(C, S, X^*, Y, \sigma, \pi)$ for some $l \in \{1, 2, 3\}$, where $X^* = X \cdot M^\pi$ and $\sigma = DH(X, Y)$ and ii) previously an $\mathsf{Execute}(C, i, S, j)$ query which produces the X^*, Y values. The associated value to this event is the k, k', sk_S^j values, respectively for $l = 1, 2, 3$.

- **correctpw**: Before any $\mathsf{Corrupt}$ query, either a **testpw!**(C, i, S, π_c) event occurs, for some C, i, S, or a **testpw**(S, j, C, π_c, l) event occurs for some S, j, C and $l \in \{1, 2, 3\}$, where π_c is the correct password.

- **pairedpwguess**: For some client and server instance Π_C^i and Π_S^j respectively, then events **testpw**(C, i, S, π_c, l) and **testpw**(S, j, C, π_c, l) both occurs for $l \in \{1, 2, 3\}$, where Π_C^i is paired with Π_S^j, and Π_S^j is paired with Π_C^i after its SERVER ACTION 1.

- **doublepwserver**: Before any $\mathsf{Corrupt}$ query, both a **testpw**(S, j, C, π_1, l) and a **testpw** (S, j, C, π_2, l) event occurs, for some S, j, π_1 and π_2, with $\pi_1 \neq \pi_2$ and $l \in \{1, 2, 3\}$.

Rule H: For a hash query $H_l(q)$, such that q and l are found in Δ_{ro}, look for the (l, q, r) row and answer with r. Otherwise, define r according to the following rule and add the record (l, q, r) to Δ_{ro}:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^{\kappa}$.

Figure 5.7: Simulation of $H(\cdot)$ random oracle queries.

In cases where it is clear enough, we write $H_l(\cdot)$ to refer to an oracle query of the form $H_l(C, S, X^*, Y, \sigma, \pi)$. For easiness of the proof we assume that for each $H_l(C, S, X^*, Y, \sigma, \pi)$ query made by $\mathcal{A}$, with $l \in \{1, 2, 3\}$, then the corresponding $H_{l'}(\cdot)$ and $H_{l''}(\cdot)$ are also made, with $l', l'' \in \{1, 2, 3\} \backslash \{l\}$ and $l' \neq l''$. The simulator sets $M := g^m \in \mathbb{G}$, where $m \xleftarrow{\$} \mathbb{Z}_q$, i.e. the simulator knows the discrete logarithm of the CRS M to base g. In the following games, we simply write $\text{Succ}_{P_i}^{\text{FtG}}$ instead of $\text{Succ}_{P_i}^{\text{PFS-FtG}}$ to denote the event that $\mathcal{A}$ wins in game G_i. We are now ready to detail the security proof of the PFS-SPAKE2 protocol.

Game $\mathbf{G_0}$: Execution of original protocol.

This game is identical to the original experiment. The challenger *simulates* to the adversary i) honest instances and ii) hash function H_l, with $l \in \{1, 2, 3\}$, which $\mathcal{A}$ is given oracle access to. As usual when relying on the random oracle model, we maintain a list Δ_{ro} in order to answer $H_l(\cdot)$ oracle queries consistently. See Figure 5.7.

Game $\mathbf{G_1}$: Uniqueness of honest sessions.

During the interaction with adversary $\mathcal{A}$, the challenger needs to simulate honest instances and generate the X^* and Y terms according to the protocol description. Let F_1 be the event where there is a collision between either an X^* or Y value, with previously seen X^* or Y values. If F_1 occurs, the challenger draws random values again until he arrives at a X^* or Y term that has not been previously seen. It is easy to show that the probability of F_1 occurring is bounded by the birthday paradox. Then for all $\mathcal{A}$:

$$\Pr\left[\,\text{Succ}_{P_0}^{\text{FtG}}(\mathcal{A})\,\right] \leq \Pr\left[\,\text{Succ}_{P_1}^{\text{FtG}}(\mathcal{A})\,\right] + \mathcal{O}\left(\frac{(n_{se}+n_{ex})(n_{se}+n_{ex}+n_{ro})}{q}\right).$$

Game $\mathbf{G_2}$: Prevent lucky guesses on hash outputs.

This game forces the adversary to query the random oracle whenever she needs to compute an $H_l(\cdot)$ function. As a result, this game rules out the possibility of $\mathcal{A}$ to output correct values k, k' or sk without calling the corresponding $H_l(\cdot)$ random oracle.

A second intention in this game is to demonstrate that the protocol can be simulated without requiring any password –unless *some bad* event occurs. There is an internal oracle H^* which instances use to compute their session keys independently of any password, see Figure 5.8. A bad event is either a $\mathbf{testpw}(C, i, S, \pi_c, l)$, a $\mathbf{testpw}(S, j, C, \pi_c, l)$ or a $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event. For simplicity, we can think of the first two events as the scenario where $\mathcal{A}$ correctly

Rule H*: For a hash query $H_l^*(q)$, such that l, q is found in Δ_{ro}^*, look for the (l, q, r) record and answer with r. Otherwise, define r according to the following rule and add the record (l, q, r) to Δ_{ro}^*:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^\kappa$.

Figure 5.8: Simulation of internal oracle $\mathrm{H}^*(\cdot)$.

guesses the correct password π_c and derives the correct session key by successfully impersonating a principal in an online attack. The third event corresponds to the case where $\mathcal{A}$ knows π_c and is able to compute a session key that was established via Execute queries (details follow in G_3).

Let P_2 be a protocol identical to P_1, except that honest instances compute the session key by making a query of the form $\mathrm{H}^*(C, S, X^*, Y^*)$, i.e. the resulting session key is independent of the password π_c and the shared secret σ. Whenever $\mathcal{A}$ asks for a $H_l(\cdot)$ oracle query, the simulator checks if any of the previously mentioned bad events occurs, in such a case, it does *backpatching* to provide consistent views to the adversary. Next we detail the changes in P_2.

- In an $\mathsf{Execute}(C, i, S, j)$ query set $X^* = g^{\tau[C,i]}$ and $Y = g^{\tau[S,j]}$, where $\tau[\cdot] \xleftarrow{\$} \mathbb{Z}_q$. Set $k := H_1^*(v)$, $k' := H_2^*(v)$ and $sk_C^i \leftarrow sk_S^j := H_3^*(v)$, where $v = C||S||X^*||Y$.
- In a CLIENT ACTION 0 query to Π_C^i, set $X^* = g^{\tau[C,i]}$, where $\tau[C, i] \xleftarrow{\$} \mathbb{Z}_q$.
- In a SERVER ACTION 1 query to Π_S^j, set $Y = g^{\tau[S,j]}$ and $k := H_1^*(C, S, X^*, Y)$, where $\tau[S, j] \xleftarrow{\$} \mathbb{Z}_q$.
- In a CLIENT ACTION 1 query to Π_C^i proceed as follows:
  - If Π_C^i is *paired* with Π_S^j, set $k' := H_2^*(v)$ and $sk_C^i := H_3^*(v)$, for $v = C||S||X^*||Y$.
  - Else if this query triggers a $\mathbf{testpw}(C, i, S, \pi_c, l)$ event, for some $l \in \{1, 2, 3\}$, then set k' and sk_C^i to the associated value of the event $\mathbf{testpw}(C, i, S, \pi_c, 2)$ and $\mathbf{testpw}(C, i, S, \pi_c, 3)$ respectively.
  - Else Π_C^i aborts.
- In a SERVER ACTION 2 query to Π_S^j proceed as follows:
  - If Π_S^j is *paired with* Π_C^i *after* some CLIENT ACTION 1 to Π_C^i, then set $sk_S^j \leftarrow sk_C^i$.
  - Else this query triggers a $\mathbf{testpw}(S, j, C, \pi_c, l)$, with $l \in \{1, 2, 3\}$, set sk_S^j to the associated value of the event $\mathbf{testpw}(S, j, C, \pi_c, 3)$.
  - Else instance Π_S^j aborts.
- In an $H_l(C, S, X^*, Y, \sigma, \pi)$ query made by $\mathcal{A}$, if it triggers a $\mathbf{testpw}(C, i, S, \pi_C, l)$, $\mathbf{testpw}(S, j, C, \pi_C, l)$ or $\mathbf{testexecpw}(C, i, S, j, \pi_C)$ event, then output the associated value of the corresponding event. Otherwise, answer according to **Rule H**.

Claim 5.13. *For all adversaries $\mathcal{A}$,*

$$\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_1}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_2}(\mathcal{A})\,\right] + \frac{2 \cdot n_{se}}{2^{\kappa}}.$$

Proof. In CLIENT ACTION 1 to Π^i_C in P_1, the input value k determines whether the client instance should *terminate* or *abort*. Ideally, an instance Π^i_C terminates if i) it is paired with another instance Π^j_S or ii) a $\mathbf{testpw}(C, i, S, \pi_C, l)$ event occurs, for $l \in \{1, 2, 3\}$. However, in P_1, an instance Π^i_C could terminate if $\mathcal{A}$ produces the correct k value expected by Π^i_C without querying the random oracle. Let F_1 be the event where in a CLIENT ACTION 1 to Π^i_C, it terminates such that i) Π^i_C is not paired with Π^j_S and ii) $\mathbf{testpw}(C, i, S, \pi_C, l)$ event does not occur, for $l \in \{1, 2, 3\}$, i.e. $\mathcal{A}$ luckly guessed the correct k value. Then $\Pr[\,F_1\,] \leq n_{se}/2^{\kappa}$. Comparable to the previous scenario, in SERVER ACTION 2 to Π^j_S, the input k' determines whether the instance Π^j_S should *terminate* or *abort*. For P_1, let F_2 be the event where in a SERVER ACTION 2 to Π^j_S, it terminates such that i) Π^j_S is not paired with Π^i_C and ii) $\mathbf{testpw}(S, j, C, \pi_C, l)$ event does not occur, for $l \in \{1, 2, 3\}$, i.e. $\mathcal{A}$ luckily guessed the correct k' value. Then $\Pr[\,F_2\,] \leq n_{se}/2^{\kappa}$. □

Game $\mathbf{G_3}$: No need to backpatch $\boldsymbol{H_l(\cdot)}$ queries against Execute queries.
This game shows that there is no need to backpatch $H_l(\cdot)$ queries to maintain consistent views for sessions established via Execute queries. More formally, let P_3 be identical to P_2 except that, in a $H_l(C, S, X^*, Y, \sigma, \pi_C)$ query made by $\mathcal{A}$, the simulator does not verify whether the $\mathbf{testexec}(C, i, S, j, \pi_C)$ event occurs or not. Let F_2 and F_3 denote the event that for some C, i, S, j, the $\mathbf{testexec}(C, i, S, j, \pi_C)$ event occurs in P_2 and P_3 respectively.

Claim 5.14. *For all adversaries $\mathcal{A}$,* $\left|\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_2}(\mathcal{A})\,\right] - \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_3}(\mathcal{A})\,\right]\right| \leq \Pr[\,F_2\,]$.

Proof. P_2 and P_3 are identical until the $\mathbf{testexec}(C, i, S, j, \pi_C)$ event occurs. The observation is that the events F_2 and F_3 are triggered as result of some interaction $\mathcal{CH}_2$ vs $\mathcal{A}$ and $\mathcal{CH}_3$ vs $\mathcal{A}$ respectively, however by definition they are identical. Then it follows that $\Pr[\,F_2\,] = \Pr[\,F_3\,]$ and to conclude the proof we simply apply Shoup's Difference Lemma [Sho04]. □

Next we will demonstrate that the probability of the event F_3 happening is bounded by the CDH assumption.

Claim 5.15. *Given $\mathcal{A}$ running in time t, there exists a CDH-solver $\mathcal{B}^{\mathcal{A}}$ with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_2}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_3}(\mathcal{A})\,\right] + n_{ro} \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\mathcal{B}^{\mathcal{A}}).$$

Proof. Let ϵ be the probability that F_3 occurs in P_2. We build an adversary $\mathcal{B}^{\mathcal{A}}$ whose goal is to solve the CDH problem using adversary $\mathcal{A}$ as a subroutine and with success probability ϵ/n_{ro}. On input two group elements $(A, B) \in \mathbb{G}^2$, $\mathcal{B}^{\mathcal{A}}$ *simulates* P_3 to $\mathcal{A}$ with the following changes:

1. For every $\mathsf{Execute}(C, i, S, j)$ query made by $\mathcal{A}$, the simulator $\mathcal{B}^{\mathcal{A}}$ sets $X^* = A \cdot g^{r_1}$, $Y = B \cdot g^{r_2}$, where $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$ is known to the simulator. Then it sets $k := H_1^*(v)$, $k' := H_2^*(v)$ and $sk_S^j \leftarrow sk_C^i := H_3^*(v)$, where $v = C||S||X^*||Y$.

2. For every $H_l(C, S, X^*, Y, \sigma, \pi_C)$ query, where $l \in \{1, 2, 3\}$, X^* and Y are generated via an $\mathsf{Execute}(C, i, S, j)$ query, add γ to the set S-DH, where:

$$\gamma = \frac{\sigma \cdot B^{m \cdot \pi_C} \cdot M^{r_2 \cdot \pi_C}}{B^{r_1} \cdot A^{r_2} \cdot g^{r_1 r_2}}$$

3. When $\mathcal{A}$ finishes, the set S-DH contains at most n_{ro} elements, where each item a possible solution to DH(A,B). Then $\mathcal{B}^{\mathcal{A}}$ picks $\gamma \xleftarrow{\$}$ S-DH as its output and hopes that it is the correct one.

The adversary $\mathcal{A}$ can only distinguish P_2 from P_3 once F_3 has occurred, but this still happens with probability $\epsilon \leq n_{ro} \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathsf{CDH}}(\mathcal{B}^{\mathcal{A}})$. We make the assumption that even if $\mathcal{A}$ distinguishes P_2 from P_3, she still runs in time t. We make the observation that G_3 guarantees forward secrecy for session keys established via $\mathsf{Execute}$ queries. □

Game G_4 : Check for successful password guesses.
Let P_4 be identical to P_3, except that if **correctpw** event occurs, the protocol stops and the adversary automatically wins.

Claim 5.16. *For all PPT adversaries $\mathcal{A}$,* $\Pr\left[\mathrm{Succ}_{P_3}^{\mathrm{FtG}}(\mathcal{A})\right] \leq \Pr\left[\mathrm{Succ}_{P_4}^{\mathrm{FtG}}(\mathcal{A})\right]$.

Proof. Obvious. □

This game simply counts for an adversary who is successful in an online dictionary attack by impersonating either a Client or the Server. The implication is that from P_4, until either **correctpw** event or a $\mathsf{Corrupt}$ query occurs, no *unpaired* client or server instance will terminate.

Game G_5 : Randomized session keys for paired instances.
Let P_5 be identical to P_4 except that if the **pairedpwguess** event occurs the protocol stops and the adversary fails.

In this game we will demonstrate that an adversary $\mathcal{A}$ who i) may actively *corrupt* any Client or Server, i.e. $\mathcal{A}$ knows the corresponding *correct* password π_C and ii) manages to compute k, k' or sk for *paired* instances Π_C^i and Π_S^j, is also a CDH-solver. Let F_4 and F_5 denote the **pairedpwguess** event occurring in P_4 and P_5 respectively.

Claim 5.17. *For all adversaries $\mathcal{A}$,* $|\Pr\left[\mathrm{Succ}_{P_4}^{\mathrm{FtG}}(\mathcal{A})\right] - \Pr\left[\mathrm{Succ}_{P_5}^{\mathrm{FtG}}(\mathcal{A})\right]| \leq \Pr\left[F_4\right]$.

Proof. Identical to Claim 5.14 □

Next we will demonstrate that the **pairedpwguess** event can happen at most with some probability bounded by the CDH assumption.

Claim 5.18. *Given $\mathcal{A}$ running in time t, there exists CDH-solver $\hat{\mathcal{B}}^{\mathcal{A}}$ with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_4}(\mathcal{A})\,\right] \leq \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right] + n_{se} \cdot n_{ro} \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\hat{\mathcal{B}}^{\mathcal{A}}),$$

Proof. Let ϵ be the probability of **pairedpwguess** event happening. We build $\hat{\mathcal{B}}^{\mathcal{A}}$, a CDH-solver that has success probability $\epsilon/(n_{se} \cdot n_{ro})$. On input $(A = g^{\alpha}, B = g^{\beta})$, $\hat{\mathcal{B}}^{\mathcal{A}}$ first sets $M = g^{m} \in \mathbb{G}$ for $m \xleftarrow{\$} \mathbb{Z}_q$, then he chooses $d \in \{1...n_{se}\}$ at random – a session target of the Test query – and *simulates* P_4 to $\mathcal{A}$ with the following changes:

1. In a CLIENT ACTION 0 query to Π^d_C with input S, set $X^* \leftarrow A$, where Π^d_C is the client instance that $\hat{\mathcal{B}}^{\mathcal{A}}$ hopes it remains *PFS-fresh*.
2. In a SERVER ACTION 1 query to Π^j_S with input $\langle C, X^* \rangle$, where there was previously a CLIENT ACTION 0 query to Π^d_C with input S and output $\langle C, X^* \rangle$, set $Y = B \cdot g^{r_{S,j}}$, where $r_{S,j} \xleftarrow{\$} \mathbb{Z}_q$.
3. In a CLIENT ACTION 1 query to Π^d_C, if Π^d_C is *unpaired* then it aborts an also $\hat{\mathcal{B}}^{\mathcal{A}}$ stops the simulation.
4. In a SERVER ACTION 2 query to Π^j_S, if it was *paired* with Π^d_C after its SERVER ACTION 1 but now is *not paired*, then Π^j_S aborts. However, the simulation continues as the instance Π^d_C may still be target of the Test query.
5. For every $H_l(C, S, X^*, Y, \sigma, \pi_C)$, made by $\mathcal{A}$, with $l \in \{1, 2, 3\}$ and where i) X^* and Y were generated by Π^d_C and Π^j_S respectively, ii) Π^j_S was paired with Π^d_C after its SERVER ACTION 1 and iii) Π^d_C was paired with Π^j_S, then add γ to the set S-DH, where:
$$\gamma = \sigma \cdot B^{m \cdot \pi_C} \cdot M^{r_{S,j} \cdot \pi_C} \cdot A^{-r_{S,j}}$$
6. When $\mathcal{A}$ finishes, the set S-DH contains at most n_{ro} elements, where each one is a possible solution to $\mathrm{DH}(A, B)$. Then $\hat{\mathcal{B}}^{\mathcal{A}}$ picks $\gamma \xleftarrow{\$}$ S-DH as its output. □

In this reduction the simulator $\hat{\mathcal{B}}^{\mathcal{A}}$ has to guess the client instance target of the Test query, say Π^d_C. The freshness requirement guarantees that a Corrupt query is only possible after the Test query, directed to Π^d_C (or its partner), has been placed. Following the reductionist approach, we showed that the **pairedpwguess** event occurs at most with probability $\epsilon \leq n_{se} \cdot n_{ro} \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\hat{\mathcal{B}}^{\mathcal{A}})$.

Game $\mathbf{G_6}$: Prevent testing more than one password per server instance.
In P_6, we restrict an adversary, who tries to masquerade as a client, from testing two passwords per session, say π_1 and π_2, in an online dictionary attack. Concretely, let P_6 be identical to P_5 except that if **doublepwserver** event occurs, then the protocol stops and the adversary fails.

Let F_5 and F_6 denote the **doublepwserver** event occurring in P_5 and P_6 respectively. By definition, it follows that $\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A}) \wedge \neg F_5 \Leftrightarrow \mathrm{Succ}^{\mathrm{FtG}}_{P_6}(\mathcal{A}) \wedge \neg F_6$.

Claim 5.19. *For all adversaries $\mathcal{A}$,* $\left|\Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right] - \Pr\left[\,\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\,\right]\right| \leq \Pr\left[\,F_6\,\right]$.

Proof. Identical to Claim 5.14. □

Next we will demonstrate that for all adversaries $\mathcal{A}$, the probability of the event **doublepwserver** occurring is bounded by the CDH assumption.

Claim 5.20. *Given $\mathcal{A}$ running in time t, there exists a CDH-solver $\hat{\mathcal{B}}^{\mathcal{A}}$ with running time $t' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp})$ such that:*

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_5}(\mathcal{A})\right] \leq \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_6}(\mathcal{A})\right] + (n_{ro})^2 \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(\tilde{\mathcal{B}}^{\mathcal{A}}).$$

Proof. We construct an algorithm $\tilde{\mathcal{B}}^{\mathcal{A}}$ that solves the CDH problem probability $\epsilon/(n_{ro})^2$, where ϵ is the probability of **pairedpwguess** event occurring. On input $(A, B) \in \mathbb{G}^2$, $\tilde{\mathcal{B}}^{\mathcal{A}}$ simulates G_5 to $\mathcal{A}$ with the following changes:

1. Set $M := A$
2. In a SERVER ACTION 1 to Π^j_S with input $\langle C, X^* \rangle$ set $Y \leftarrow B \cdot g^y$, where $y \xleftarrow{\$} \mathbb{Z}_q$, and sends back $\langle S, Y, k \rangle$. From P_4 it holds that no unpaired instances can terminate. Specifically, unpaired client and server instances abort in CLIENT ACTION 1 and SERVER ACTION 2 respectively.
3. When $\mathcal{A}$ terminates, the simulator selects at random a pair of random oracle queries $H_l(C, S, X^*, Y, \sigma_1, \pi_1)$ and $H_l(C, S, X^*, Y, \sigma_2, \pi_2)$, where $\pi_1 \neq \pi_2$, there was a SERVER ACTION 1 with input $\langle C, X^* \rangle$ and output $\langle S, Y, k \rangle$, and $l \in \{1, 2, 3\}$. Then computes γ as possible solution to $DH(A, B)$, where:

$$\gamma = A^{-y} \cdot (\sigma_1/\sigma_2)^{\phi},$$

where ϕ is the multiplicative inverse of $(\pi_2 - \pi_1)$ in $\mathbb{Z}_q$. □

P_6 and P_5 are identical unless the **doublepwserver** event occurs, however, this only occurs with probability $\epsilon \leq n^2_{ro} \cdot \mathrm{Adv}^{\mathtt{CDH}}_{\mathbb{G}}(t')$. The quadratic degradation factor is due to the $\tilde{\mathcal{B}}^{\mathcal{A}}$ not having other option rather than guessing two queries $H_l(C, S, X^*, Y, \sigma_1, \pi_1)$ and $H_l(C, S, X^*, Y, \sigma_2, \pi_2)$ such that $\sigma_1 = \mathrm{DH}\ (X^*/M^{\pi_1}, Y)$ and $\sigma_2 = \mathrm{DH}\ (X^*/M^{\pi_2}, Y)$.

Game G_7 : Internal password oracle.
In protocol P_7, we consider an internal password oracle $\mathcal{O}_\pi$ who handles every password request and is only available to the challenger. Specifically, the challenger queries the $\mathcal{O}_\pi$ to i) assign passwords to users, ii) answer Corrupt queries and iii) determine if the **correctpw** event occurs.

Claim 5.21. *For all adversaries $\mathcal{A}$,* $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_6}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right]$.

Proof. It follows from inspection. □

Claim 5.22. *For all adversaries $\mathcal{A}$,*

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] \leq \frac{1}{2} + \frac{n_{se}}{2 \cdot |D|}.$$

Proof.

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] = \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \mathrm{correctpw}\right] \cdot \Pr\left[\mathrm{correctpw}\right] + \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\mathrm{correctpw}\right] \cdot \Pr\left[\neg\mathrm{correctpw}\right]. \quad (5.3)$$

We know from P_6 that $\mathcal{A}$ can test at most one password per instance in an active attack. Then in P_7, $\Pr\left[\mathrm{correctpw}\right] \leq n_{se}/|D|$. We examine the second term of Equation 5.3. The security experiment requires the adversary to make a Test query to some *PFS-fresh* instance Π^i_U of his choice. It is easy to show that the view of $\mathcal{A}$ is independent of the *sk* on which she is challenged: i) P_1 prevents two or more client or server instances accepting with the same *sid*, which would violate the partnering definition allowing $\mathcal{A}$ to trivially win, ii) it follows from P_4 that, before any Corrupt query, only instances that are *paired* instances can reach terminate state – and therefore be target of a Test query – and iii) from P_5 it holds that for such *paired* instances, the view of $\mathcal{A}$ is independent of *sk* for the session target of the Test query. In such a case, $\mathcal{A}$ can only succeed with probability 1/2. We finally obtain $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\mathrm{correctpw}\right] = 1/2$. □

■

5.4 Conclusion and Future Work

We proved that SPAKE2 protocol provably satisfies weak forward secrecy, however, proving perfect forward secrecy seems to be a harder task. Consider the following scenario: $\mathcal{A}$ masquerades as a client and sends an arbitrary message X^* to a server instance Π^j_S, the later computes Y^*, its session key, answers back with Y^* and *terminates*. Now $\mathcal{A}$ makes a $\mathsf{Test}(S, j)$ query, receives the challenge and then corrupts the tested server. This is a legitimate strategy for the adversary, as corruption occurred after the Test query, then the instance Π^j_S remains PFS-fresh or put differently, the session key in the Π^j_S instance should remain secret as it was established before the password compromise – even if it was established with the adversary. The difficulty is that, even though the proof shows that $\mathcal{A}$ cannot test two passwords per instance, in this particular scenario the simulator cannot determine the password to which $\mathcal{A}$ committed in X^* as she has not asked for any random oracle query. This problematic scenario does not take place in the PFS-SPAKE2 protocol, due to the fact that an instance would not accept the session key (then it is not subject to a Test query) unless it confirms that it was established with a legitimate partner.

As future work, we would like to study if the SPAKE2 and PFS-SPAKE2 protocols compose securely with symmetric-key encryption schemes. This question has practical relevance, as in TLS 1.3 the aforementioned primitives would be used not in stand alone operation but as a combined system.

CHAPTER 6

Tightly-Secure PAK(E)

6.1 Introduction

We present a security reduction for the PAK protocol [BMP00, Mac01b, Mac02a] instantiated over Gap Diffie-Hellman groups that is tighter than previously known reductions. We discuss the implications of our results for concrete security. Our proof is the first to show that the PAK protocol can provide meaningful security guarantees for values of the parameters typical in today's world.

6.1.1 Security Models and Reductions for PAKEs

When evaluating different PAKE designs, two main criteria are the protocol's *efficiency* in terms of computation and communication, and the *security guarantees* that the protocol provides. Of these two criteria, the efficiency is easier to understand by just looking at the protocol description. On the other hand, it is difficult to judge whether a protocol is secure. A necessary condition for security is that no attacks on the protocol have been found so far, but most researchers agree that this is not sufficient.

One way to rigorously discuss the security of PAKE protocols is to formally define a security challenge: an interaction between two algorithms called a challenger and an adversary. The interaction is designed to model the capabilities that a real world adversary is believed to have; the success of an adversary in the security challenge corresponds to a successful attack on the protocol. Several such security models have been introduced over the years. A few prominent ones are the indistinguishability-based models of Bellare, Pointcheval and Rogaway [BPR00] and Abdalla, Fouque and Pointcheval [AFP05], the simulation-based model of Boyko, MacKenzie and Patel [BMP00], and the Universally Composable (UC) model of Canetti et al. [CHK$^+$05].

In this approach, the security of a protocol is established in the following way: given an adversary $\mathcal{A}$ that runs in time t and has success probability ϵ in the security challenge, one constructs an algorithm $\mathcal{B}^{\mathcal{A}}$ known as a reduction. $\mathcal{B}^{\mathcal{A}}$ runs $\mathcal{A}$ as a subroutine and solves some known hard computational problem in time $t_{\mathcal{R}}$ and with success probability $\epsilon_{\mathcal{R}}$. If it is widely believed that it is impossible to solve the hard computational problem in time $t_{\mathcal{R}}$ and with success probability $\epsilon_{\mathcal{R}}$, then one can conclude that no adversary running in time t can have a probability of ϵ to successfully attack the protocol.

The aforedmentioned way of analyzing the security of a given protocol is known as *reductionist* approach or *provable security*, which results in theorems of the type: protocol $\mathcal{P}$ satisfies a given *security definition* for all *adversaries* running in polynomial time and under the *assumption* that some problem is hard to compute.

6.1.2 Concrete Security and Tight Reductions

Security Level. Given an instance of a protocol $\mathcal{P}$, its associated *security level* is a measure of the strength provided and it is relevant when investigating the *concrete security* provided by such instance. The statement "an instantiation of $\mathcal{P}$ provides n bits of security" is usually understood as "an adversary would have to do 2^n operations to break the given instance of $\mathcal{P}$". Expressed differently, it captures the notion that a particular *instantiation* of some protocol $\mathcal{P}$, which provides an n-bit *security level*, is as hard to break as some *idealized* cryptographic function where the only attack possible is a brute force attack on a n-bit key space. Thus, it is generally accepted that an attacker would require to perform $T \geq 2^n$ operations to break $\mathcal{P}$, or alternatively, the success probability of any efficient attack is bounded by $\epsilon \leq 2^{-n}$ [MW18].

Security Parameter. Similarly, the *security parameter* κ allows us to tune the desired *security level* of a protocol $\mathcal{P}$ when it is to be instantiated. Intuitively, the larger the security parameter is, the *more secure* $\mathcal{P}$ is and unfortunately the *less efficient* it becomes. Therefore, the correct selection of κ is decisive to achieve an appropriate balance between security and efficiency.

Provable security relies on complexity theory, thus it is natural to expect that security proofs are expressed *asymptotically*: To prove that a protocol $\mathcal{P}$ is secure, it is enough to show the existence of a *polynomial time reduction* $\mathcal{R}$ from a hard problem to $\mathcal{P}$. The intuition behind it is to demonstrate that the *advantage* of any adversary $\mathcal{A}$ on breaking $\mathcal{P}$ is a *negligible* function of the *security parameter* κ. Then it is guaranteed that for *sufficiently large* κ, the advantage of $\mathcal{A}$ on breaking $\mathcal{P}$ is sufficiently small. A legitimate question would be: how large does κ need to be? As nicely explained in [MW18], the asymptotic approach *per se* only provides a qualitative classification of protocols into secure and insecure ones.

Tight Reductions. A security reduction $\mathcal{R}$ not only gives confidence in the security of protocol $\mathcal{P}$, it also guides the choice of the security parameter κ when $\mathcal{P}$ needs to be instantiated. The reason is the following: a reduction $\mathcal{R}$ reduces a problem $\mathcal{C}$ to breaking protocol $\mathcal{P}$, by using an adversary $\mathcal{A}$ against $\mathcal{P}$ as a subroutine with the goal of solving $\mathcal{C}$. Let t and ϵ be $\mathcal{A}'s$ running time and success probability respectively (both are functions of the security parameter). Then

the running time and success probability of $\mathcal{R}$ are given by $t_{\mathcal{R}} \geq t$ and $\epsilon_{\mathcal{R}} \leq \epsilon$ respectively. We can observe that more resources are needed for solving $\mathcal{C}$ than for breaking $\mathcal{P}$, this difference is formalized by the *tightness-gap* introduced in [CKMS16]:

$$\frac{t_{\mathcal{R}}}{\epsilon_{\mathcal{R}}} = \gamma \cdot \frac{t}{\epsilon},$$

where $\gamma \geq 1$. A reduction is called *tight* if γ is a small constant, and *loose* otherwise.

Tight reductions are considered preferable over non-tight ones [Bel99]. The reason is that when the reduction is *tight*, it is guaranteed that breaking the cryptographic system is at least as hard as breaking a well studied hardness assumption, say $\epsilon \approx \epsilon_{\mathcal{R}}$. On the other side, when the reduction is *loose*, it is possible that breaking $\mathcal{P}$ is substantially easier than solving the underlying hardness assumption, i.e. $\epsilon \leq \gamma \cdot \epsilon_{\mathcal{R}}$. That being said, tight reductions are also known as security preserving while loose reductions introduce a *security degradation factor* γ.

The instantiation of a provably-secure protocol is a delicate task. Presumably, the first duty is the analysis of the concrete security that the protocol aims to provide, and then translate this into the selection of the security parameter κ in such a way that: i) it is guaranteed that a reasonable adversary cannot break the protocol and ii) the protocol remains efficient enough to be used in practice. It is in this this scenario where tight reductions are preferrable, since the security parameter can be derived assuming that the *security level* of the protocol is equivalent to that of the hardness assumption. On the other hand, when the reduction is not security preserving, one has to compensate the security degradation factor – also known as tightness-gap – by choosing larger κ, which unavoidably results in less efficient implementations.

6.1.3 Loose reduction in the PAK protocol

One of the PAKE protocols whose security has been studied in the provable security framework is the PAK protocol [BMP00, Mac01b, Mac02a] (in Section 6.2 we provide a detailed description of it). It is a PAKE protocol with several desirable characteristics: low computation and communication cost, and security proofs in two different security models: the simulation-based model of Boyko, MacKenzie and Patel [BMP00] and the so-called Find-then-Guess (FtG) model of Bellare, Pointcheval and Rogaway [BPR00]. A modified version of PAK has been used to detect man-in-the-middle attacks against SSL/TLS without third-parties [DAT12], and a lattice-based version of PAK has been used to provide security against quantum adversaries [DAL$^+$17]. Moreover, variants of the PAK protocol have been included in IEEE standard [IEE02], while the patent held by Lucent Technologies [Mac02b] has recently expired. Therefore, the PAK protocol is a candidate for wide-scale practical deployment.

While there are security proofs for PAK in two different models, in both cases the reductions are loose, meaning either that the running time $t_{\mathcal{R}}$ of the reduction $\mathcal{R}$ is much larger than the running time t of the adversary $\mathcal{A}$ or that the success probability $\epsilon_{\mathcal{R}}$ of $\mathcal{R}$ is much smaller than the success probability ϵ of $\mathcal{A}$, or some combination of the two.

A loose reduction is usually considered less than ideal. From a qualitative point of view, a reduction gives the assurance that "breaking the protocol is at most a little easier than solving the hard computational problem" [Ecr12]. However, if a reduction is loose, it leaves open the possibility that "a little easier" is in fact "substantially easier". From a quantitative point of view, a loose reduction means that larger security parameters must be chosen to guarantee a given level of security, which in turn increases the communication and computation cost of the protocol. We illustrate the last point by looking in detail at the best previous result for PAK [Mac02a, Theorem 6.9], which we reproduce here for convenience.

Theorem 6.1 (Theorem 6.9 in [Mac02a])**.** *Consider the PAK protocol instantiated over a group $\mathbb{G} = \langle g \rangle$ of prime order q and with password dictionary of size $|D|$. Let $\mathcal{A}$ be an adversary that runs in time t and performs at most n_{se}, n_{ex}, n_{re}, n_{co}, n_{ro} queries of type Send, Execute, Reveal, Corrupt, Random Oracle and a single Test query. Let $\mathrm{Adv}(\mathcal{A})$ be the advantage of this adversary in the security challenge as defined in the Find-then-Guess model considering perfect forward secrecy.*[1] *Then for all adversaries $\mathcal{A}$:*

$$\mathrm{Adv}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + \mathcal{O}\left(n_{se} \cdot (n_{ro})^2 \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{B}^{\mathcal{A}}) + \frac{(n_{se}+n_{ex})(n_{ro}+n_{se}+n_{ex})}{q}\right), \tag{6.1}$$

where $\mathcal{B}^{\mathcal{A}}$ is a CDH-solver algorithm, running in time $t' = \mathcal{O}(t + ((n_{ro})^2 + n_{se} + n_{ex})t_{exp})$, and t_{exp} is the time required for an exponentiation in $\mathbb{G}$.

We plug in some concrete values in the above theorem. For the order of the group q, we use the recommended $q \approx 2^{256}$ for long-term security from [Ecr12, Chapter 7]. For the number of random oracle queries, we take $n_{ro} \approx 2^{63}$, the number of SHA1 computations performed in the recent attack [SBK+17]. Next, we use the approximation that solving the discrete logarithm problem in group $\mathbb{G}$ takes about $\sqrt{q} \approx 2^{128}$ operations [Len06, Section 7]. We see that with these values of the parameters, we can estimate for all PPT adversaries

$$(n_{ro})^2 \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{A}) \approx 1$$

and therefore the term

$$n_{se}(n_{ro})^2 \cdot \mathrm{Adv}_{\mathbb{G}}^{\mathtt{CDH}}(\mathcal{A}) >> 1$$

makes the right hand-side of Equation 6.1 meaningless in bounding $\mathrm{Adv}(\mathcal{A})$, which, by definition, is a number less than or equal to 1. This means that we cannot reasonably claim that the security proof gives the guarantee "an adversary can essentially do no better than an online dictionary attack", except in the trivial case when the online dictionary attack itself succeeds with probability close to one.

[1] We refer to Section 4.2 for a description of the Find-then-Guess model.

6.1.4 Our contribution

We provide a tight reduction for PAK instantiated over Gap Diffie-Hellman groups; these are groups in which solving the Decisional Diffie-Hellman problem is easy but solving the Computational Diffie-Hellman problem is equivalent to solving the Discrete Logarithm problem and is believed to be hard [JN03]. We employ proof techniques that have been used previously in [ACP05]. The formal statement of our result can be found in Theorem 6.2.

Theorem 6.2. *(Security in the PFS-FtG Model). Consider the PAK protocol instantiated over a Gap Diffie-Hellman group $\mathbb{G}_1 = \langle g \rangle$ of order q and with password dictionary of size $|D|$. Let $\mathcal{A}$ be an adversary that runs in time t and performs at most n_{se}, n_{ex}, n_{re}, n_{co}, n_{ro} queries of type Send, Execute, Reveal, Corrupt, Random Oracle and a single Test query. Let $\mathrm{Adv}(\mathcal{A})$ be the advantage of this adversary in the security challenge as defined in the FtG model considering perfect forward secrecy. Then for all adversaries $\mathcal{A}$:*

$$\mathrm{Adv}(\mathcal{A}) \leq \frac{n_{se}}{|D|} + 8\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\mathcal{B}^{\mathcal{A}}) + \mathcal{O}\left(\frac{(n_{se}+n_{ex})(n_{ro}+n_{se}+n_{ex})}{q}\right), \tag{6.2}$$

where $\mathcal{B}^{\mathcal{A}}$ is a Gap-DH solver algorithm, running in time $t'' = \mathcal{O}(t + (n_{ro}+n_{se}+n_{ex})t_{exp} + (n_{se}+n_{ro})t_{ddh})$, where t_{exp} and t_{ddh} denote the time required for an exponentiation in $\mathbb{G}_1$ and deciding DDH in $\mathbb{G}_1$, respectively.

We perform a similar analysis of our result as in the previous section, using the same values of q, and n_{ro}. Since $t'' << 2^{128}$ (assuming the most powerful adversaries today have t at most $\approx 2^{80}$ to 2^{85}), we can assume that $\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\mathcal{B}^{\mathcal{A}}) \lessapprox 2^{-35}$, assuming that $\mathcal{B}^{\mathcal{A}}$ can compute at most 2^{85} operations. Furthermore, the term $\mathcal{O}\left((n_{se}+n_{ex})(n_{ro}+n_{se}+n_{ex})/q\right)$ is negligible compared to the other two terms. Thus, by using the tight reduction, we are able to obtain the following guarantee: assuming that $\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\mathcal{B}^{\mathcal{A}}) \lessapprox 2^{-35}$, then for all adversaries $\mathcal{A}$ with running time $t \lessapprox 2^{85}$, the advantage in breaking the PAK protocol instantiated over a Gap Diffie-Hellman group of order $\approx 2^{256}$ is at most $\approx 2^{-30}$ higher than the advantage of breaking the protocol using the best online dictionary attack for the given password distribution and login attempt policy.[2] Furthermore, the reduction is security preserving, in the sense that if the group $\mathbb{G}_1$ provides l-bit security level, then the instantiation of the PAK protocol offers also l-bit security level.

Thus, by relying on the Gap Diffie-Hellman assumption instead of the CDH assumption as in [Mac02a] we are able to remove the degradation factors that cause the previous security proof for PAK to fail to provide meaningful guarantees for typical values of the parameters in today's world.

Notation. In general, we use $\mathbb{G}$ to denote any cyclic group while $\mathbb{G}_1$ refers to a bilinear group. Let H be a full-domain hash mapping $\{0,1\}^*$ to $\mathbb{G}_1$. All remaining hash functions, H_1, H_2 and H_3, map from $\{0,1\}^*$ to $\{0,1\}^{\kappa}$.

[2] We refer to [WW16, Figure 4] for an estimation of the advantage of online dictionary attacks as a function of the number of guesses for two real-world password datasets.

6.2 The PAK Protocol

In this section, we describe the PAK protocol from [Mac02a], whose mathematical description is presented in Figure 6.1. A few other variants of PAK were developed in [Mac01b]. Originally, the first proof of this protocol has been done in the simulation based model [BMP00].

6.2.1 Protocol description

Here, we make use of the same notation as in [Mac02a]. Now, we describe the protocol informally.

Before any protocol execution, public parameters are published and passwords are shared between clients and servers during the initialization phase. More specifically, for efficiency reasons and security in case of compromise of the password file, servers only keep the inverse element of each password's hash value.

The PAK protocol consists of three message rounds. In the first message round, the client sends a group element m – generated by multiplying a random group element α with the mask γ (also a group element) that is derived from the shared password π – along with its ID to the server. In the second message round, upon receiving the message C, m, the server first checks with the *acceptable* function if the received value m is an element of $\mathbb{G}_1$. Then, it selects a random group element μ, removes the mask from the received m, and computes the shared secret σ, confirmation codes k, k', a session key sk and sets sid and pid values (thus accepting). Once all these values are computed, the server sends μ and k to the client. Upon receiving the second message, the client first checks if μ is a valid group element. If so, it computes the shared secret and confirmation code k and checks the validity of the latter. If all checks are correct, the client computes his confirmation code k' and a session key sk, sets sid and pid values, and then it sends k' in the third message round and terminates. The server, once it receives the value k' and checks its validity, also terminates.

6.2.2 Instantiating the protocol over Gap Diffie-Hellman groups

Gap-DH groups were introduced in the pioneering work of Boneh, Lynn and Shacham [BLS01]. For instance, Gap-DH groups can be derived from the supersingular elliptic curve given by the equation $y^2 = x^3 + 2x \pm 1$ over the field $\mathbb{F}_{3^l}$. It can be seen that for some values of l the number of points in this curve divides $3^{6l} - 1$. The value 6 is called the multiplier that has to be neither too small for the CDH problem to be hard, nor too big for the Decision Diffie-Hellman Oracle (DDH-O) to be efficient. An example of DDH-O on this curve is the Weil pairing [Sil09]. Gap-problems were also studied by Okamoto and Pointcheval [OP01].

In order to have efficient implementations of the PAK protocol, $H : \{0,1\}^* \to \mathbb{G}_1$ must be an efficiently computable function. We point the reader to [Mac01b, MJ16] for efficient implementations of H. Note that it is crucial for such an algorithm to run in constant time, otherwise timing attacks on a password are possible. For more details on pairings, we refer readers to [GPS08].

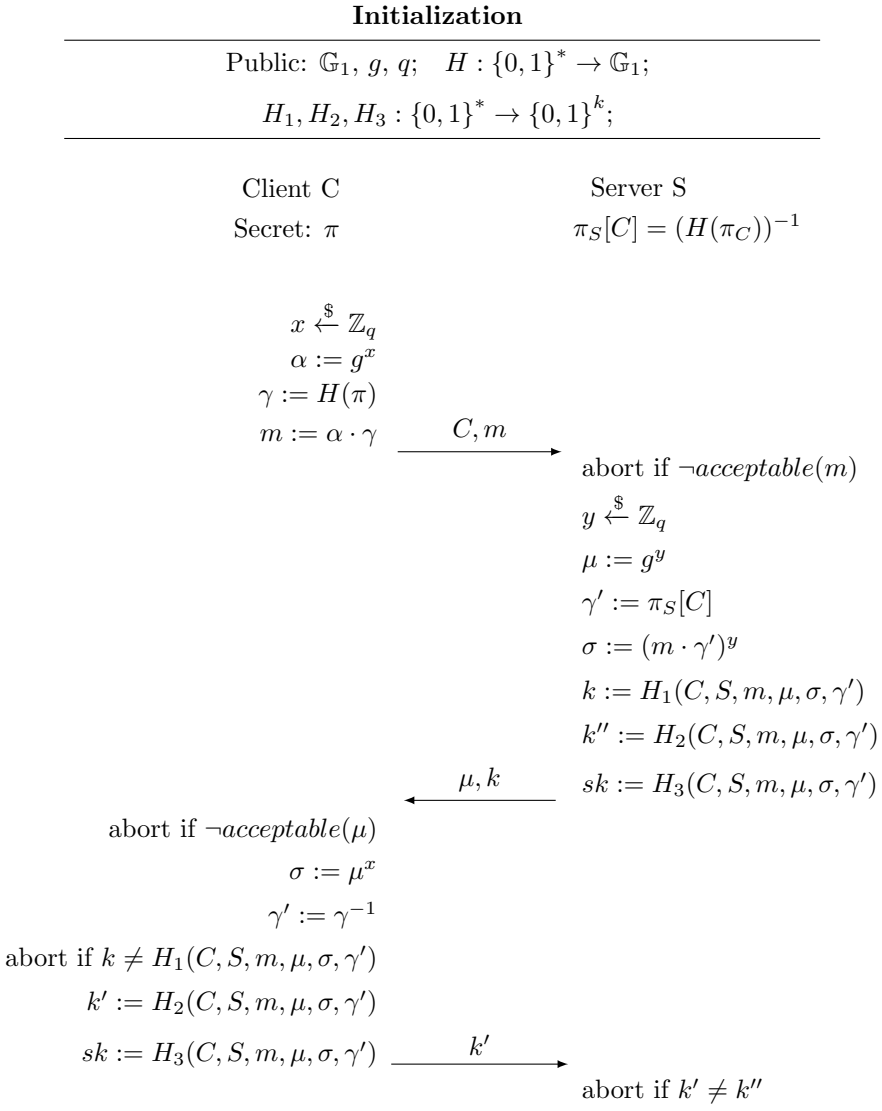

Figure 6.1: The PAK protocol.

G_0 : Original protocol.
G_1 : Force uniqueness of instances.
G_2 : Forbid lucky guesses on hash outputs and backpatch for consistency.
G_3 : **Randomize session keys for Execute queries (Gap-DH).**
G_4 : Check password guesses.
G_5 : **Randomize session keys for paired instances (Gap-DH).**
G_6 : **Forbid two password guesses per online attempt on server (Gap-DH).**
G_7 : Internal password oracle.

Figure 6.2: Description of games for the original PAK.

6.3 Proof of Security

In this section, we prove the security of the PAK protocol in the FtG model, considering perfect forward secrecy (see Definition 4.1 from Chapter 4) and instantiated over Gap Diffie-Hellman groups. Due to similarities with the proof of the original PAK protocol [Mac02a], we present an overview for those security games that remain the same as in the original protocol and focus on those that deviate from the original proof. In Figure 6.2 we provide the description of the game hops and highlight those games which differ from the original security proof.

The main difference between the existing proof in the FtG model and our proof is that our reduction algorithm makes use of a Decisional Diffie-Hellman Oracle (DDH-O). Such oracle is available in gap groups, and it will output 1 on input (g, g^x, g^y, g^z) if $g^z = DH(g^x, g^y)$ and 0 otherwise. This additional information can be leveraged – in games G_3, G_5 and G_6 – to increase the success probability and reduce the running time of the reduction compared to Theorem 6.9 in [Mac02a].

We borrow from [Mac02a] the proof structure and necessary nomenclature that will allow us to prove the security of the PAK protocol instantiated over Gap Diffie-Hellman groups. First, we introduce the terminology that deals with adversary's actions and partnering.

We say "in a CLIENT ACTION κ query to Π_C^i", to refer to "in a Send query to Π_C^i that results in the execution of the CLIENT ACTION κ procedure" and "in a SERVER ACTION κ query to Π_S^j", to refer to "in a Send query to Π_S^j that results in the execution of the SERVER ACTION κ procedure". A client instance Π_C^i is paired with a server instance Π_S^j if there is a CLIENT ACTION 0 query to Π_C^i with input S and output $\langle C, m\rangle$, there is a SERVER ACTION 1 query to Π_S^j with input $\langle C, m\rangle$ and output $\langle \mu, k\rangle$ and there is a CLIENT ACTION 1 query to Π_C^i with input $\langle \mu, k\rangle$. A server instance Π_S^j *is paired* with client instance Π_C^i whenever there is a CLIENT ACTION 0 query to Π_C^i with input S and output $\langle C, m\rangle$, there is a SERVER ACTION 1 query to Π_S^j with input $\langle C, m\rangle$ and output $\langle \mu, k\rangle$, and if there is a SERVER ACTION 2 query to Π_S^j with input k', then there was previously a CLIENT ACTION 1 query to Π_C^i with input $\langle \mu, k\rangle$ and output k'.

Next we describe those events taken from [Mac02a] which are required in our proof of security. Similar to the security proof of SPAKE2 and PFS-SPAKE duly described in Section 5.2 and 5.3 respectively, we make use of the following events to describe the adversarial behavior which would allow us to formally identify i) successful online dictionary attacks and ii) some "bad" events which, if they happened, would cause a break on the security of the protocol. In particular, G_3 corresponds to **testexecpw**, G_5 to **pairedpwguess**, and G_6 to **doublepwserver** event.

- **testpw**(C, i, S, π, l): for some m, μ and γ', $\mathcal{A}$ makes i) an $H_l(C, S, m, \mu, \sigma, \gamma')$ query, ii) a CLIENT ACTION 0 query to a client instance Π_C^i with input S and output $\langle C, m\rangle$, iii) a CLIENT ACTION 1 query to Π_C^i with input $\langle \mu, k\rangle$ and iv) an $H(\pi)$ query returning $(\gamma')^{-1}$, where the last query is either the $H_l(\cdot)$ query or the CLIENT ACTION 1 query, $\sigma = DH(\alpha, \mu)$, $m = \alpha \cdot (\gamma')^{-1}$ and $l \in \{1, 2, 3\}$. The associated value of this event is output of the $H_l(\cdot)$ query, or the k, k' or sk_C^i value, whichever is set first.

- **testpw!**(C, i, S, π): for some k, a CLIENT ACTION 1 query with input $\langle \mu, k\rangle$ causes a **testpw**$(C, i, S, \pi, 1)$ event to occur, with associated value k.

- **testpw**(S, j, C, π, l): for some m, μ, γ' and k, $\mathcal{A}$ makes an $H_l(C, S, m, \mu, \sigma, \gamma')$ query, and previously made i) a SERVER ACTION 1 query to a server instance Π_S^j with input $\langle C, m\rangle$ and output $\langle \mu, k\rangle$, and ii) an $H(\pi)$ query returning $(\gamma')^{-1}$, where $\sigma = DH(\alpha, \mu)$, $m = \alpha \cdot (\gamma')^{-1}$ and ACCEPTABLE(m). The associated value of this event is k, k' or sk_s^j generated by the server instance Π_S^j.

- **testpw!**(S, j, C, π): SERVER ACTION 2 query to Π_S^j is made with input k', and previously a **testpw**$(S, j, C, \pi, 2)$ event occurs with associated value k'.

- **testpw**$^*(S, j, C, \pi)$: **testpw**(S, j, C, π, l) event occurs for some $l \in \{1, 2, 3\}$.

- **testpw**(C, i, S, j, π) : both a **testpw**(C, i, S, π, l) and **testpw**(S, j, C, π, l) event occur, for some $l \in \{1, 2, 3\}$, where Π_C^i is *paired* with Π_S^j, and Π_S^j is *paired* with Π_C^i after its SERVER ACTION 1 query.

- **testexecpw**(C, i, S, j, π): for some m, μ and γ', $\mathcal{A}$ makes an $H_l(C, S, m, \mu, \sigma, \gamma')$ query, for $l \in \{1, 2, 3\}$, and previously made i) an Execute(C, i, S, j) query that generates m, μ, and ii) an $H(\pi)$ query returning $(\gamma')^{-1}$, where $\sigma = DH(\alpha, \mu)$ and $m = \alpha \cdot (\gamma')^{-1}$.

- **correctpw**: before any Corrupt query, either a **testpw!**(C, i, S, π_c) event occurs for some C,i and S, or a **testpw**$^*(S, j, C, \pi_c)$ event occurs for some S, j, and C, where $\pi_c = \pi[C, S]$, i.e. the correct password.

- **doublepwserver**: before any Corrupt query, both **testpw**$^*(S, j, C, \pi)$ and a **testpw*** $(S, j, C, \hat{\pi})$ events occur, for some S, j, C and $\pi \neq \hat{\pi}$.

- **pairedpwguess**: a **testpw**(C, i, S, j, π_c) event occurs, for some C, i, S and j.

Rule H: In a hash query $H(\pi)$, such that π is found in the list Δ_{ro}, look for the $(\pi, \Phi, \psi[\pi])$ record and answer with Φ. Otherwise, define Φ according to the following rule:

- Choose $\psi[\pi] \xleftarrow{\$} \mathbb{Z}_q$ and compute $\Phi := g^{\psi[\pi]}$. Then write the record $(\pi, \Phi, \psi[\pi])$ to Δ_{ro}.

Figure 6.3: Simulation of $H(\cdot)$ random oracle queries.

Rule $\mathbf{H}_l$: In a hash query $H_l(\nu)$, such that ν and l are found in Δ^l_{ro}, look for the (l, ν, r) row and answer with r. Otherwise, define r according to the following rule:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^\kappa$ and write the record (l, ν, r) to Δ^l_{ro}, where $\{0,1\}^\kappa$ denotes the session key space.

Figure 6.4: Simulation of $H_l(\cdot)$ random oracle queries, for $l \in \{1, 2, 3\}$.

Proof of Theorem 6.2: We will denote by P_i the protocol executed in game G_i, for i from 0 to 7. Before we start with the revised games, we will take a moment to describe how the simulator answers H and H_l *random oracle queries* (see Figures 6.3 and 6.4). It is important to highlight that, in an $H(\pi)$ query returning $\Phi = g^{\psi[\pi]}$, the simulator has access to the discrete logarithm of Φ, i.e. the $\psi[\pi]$ value. To simplify the proof, we make the assumption that, whenever the adversary makes an H_l query, it also makes $H_{l'}$ and $H_{l''}$ queries such that $\{l, l', l''\} = \{1, 2, 3\}$.

Game $\mathbf{G_0}$: Execution of the original protocol.
In this game, the challenger runs the original protocol P_0 for the adversary $\mathcal{A}$.

$$\mathrm{Adv}^{\mathrm{FtG}}_{P}(\mathcal{A}) = \mathrm{Adv}^{\mathrm{FtG}}_{P_0}.$$

Game $\mathbf{G_1}$: Force uniqueness of instances.
Let P_1 be exactly the same as P_0, except that if any of the values m and μ chosen by honest instances collide with previously generated ones, the protocol aborts and the adversary fails.

The probability of this event happening is negligible in the security parameter and limited by the birthday bound. More precisely, for all adversaries $\mathcal{A}$:

$$\mathrm{Adv}^{\mathrm{FtG}}_{P_0}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_1}(\mathcal{A}) + \frac{(n_{se} + n_{ex})(n_{se} + n_{ex} + n_{ro})}{q}.$$

Game $\mathbf{G_2}$: Forbid lucky guesses on hash outputs and backpatch for consistency.
Let P_2 be defined as P_1, with the difference that now the simulator answers Send and Execute queries without making any H or H_l random oracle queries. Furthermore, honest instances use

Rule $\mathbf{H}_l^*$: For a hash query $H_l^*(\nu)$, such that l, ν is found in Δ_{ro}^*, look for the (l, ν, r) record and answer with r. Otherwise, define r according to the following rule and add the record (l, ν, r) to Δ_{ro}^*:

- Choose a random string $r \xleftarrow{\$} \{0,1\}^\kappa$.

Figure 6.5: Simulation of internal oracle $\mathrm{H}_l^*(\cdot)$.

an *internal oracle* to establish the confirmation codes and session keys values with a query of the form $H_l^*(C, S, m, \mu)$, duly described in Figure 6.5, i.e. independently of the correct password π_c and the shared secret σ. From start to finish, random oracle queries are *backpatched* – if needed – to ensure consistency in the view of the adversary.

Next, we detail how the simulator responds to queries of the type Send and Execute.

- In an Execute(C, i, S, j) query set $m = g^{\tau[C,i]}$, $\mu = g^{\tau[S,j]}$, $k := H_1^*(\nu)$, $k' := H_2^*(\nu)$ and $sk_C^i \leftarrow sk_S^j := H_3^*(\nu)$, where $\tau[\cdot] \xleftarrow{\$} \mathbb{Z}_q$ and $\nu = C||S||m||\mu$.
- In a CLIENT ACTION 0 query to Π_C^i, set $m = g^{\tau[C,i]}$ where $\tau[C,i] \xleftarrow{\$} \mathbb{Z}_q$.
- In a SERVER ACTION 1 query to Π_S^j, set $\mu = g^{\tau[S,j]}$, $k := H_1^*(\nu)$ and $sk_S^j := H_3^*(\nu)$, for $\tau[S,j] \xleftarrow{\$} \mathbb{Z}_q$ and $\nu = C||S||m||\mu$.
- In a CLIENT ACTION 1 query to Π_C^i proceed as follows:
  - If Π_C^i is *paired* with Π_S^j, then set $k' := H_2^*(\nu)$ and $sk_C^i \leftarrow sk_S^j$, for $\nu = C||S||m||\mu$.
  - Else, if this query triggers a $\mathbf{testpw}(C, iS, \pi_c, l)$ event, for some $l \in \{1,2,3\}$ and π_c the shared password between C and S, then set k' and sk_C^i to the associated value of the events $\mathbf{testpw}(C, i, S, \pi_c, 2)$ and $\mathbf{testpw}(C, i, S, \pi_c, 3)$ respectively.
  - Otherwise Π_C^i aborts.
- In a SERVER ACTION 2 query to Π_S^j proceed as follows:
  - If Π_S^j is paired with Π_C^i after a CLIENT ACTION 1 or if this query triggers a $\mathbf{testpw}(S, j, C, \pi_c, l)$ event, with $l \in \{1,2,3\}$, then *terminate*.
  - Else *abort*.
- In a $H_l(C, S, m, \mu, \sigma, \gamma')$ query made by $\mathcal{A}$, if it triggers a $\mathbf{testpw}(C, i, S, \pi_c, l)$, $\mathbf{testpw}(S, j, C, \pi_c, l)$ or $\mathbf{testexecpw}(C, i, S, j, \pi_c)$ event, then output the associated value of the corresponding event. Otherwise, answer according to **Rule $\mathbf{H}_l$**.

We note that the protocol can be simulated without requiring any password unless i) $\mathcal{A}$ successfully impersonates a principal in an online dictionary attack or ii) $\mathcal{A}$ computes the session key established via Execute queries; we make use of the of $\mathbf{testpw}(C, i, S, \pi_c, l)$, $\mathbf{testpw}(S, j, C, \pi_c, l)$

or **testexecpw**(C, i, S, j, π_c) events to formally describe such scenarios. In such a case, the simulator does *backpatching* to provide consistent views to the adversary, which is possible since it has the necessary information to detect whenever such events occur, i.e. discrete logarithms and access to a DDH oracle.

In addition, P_2 forbids lucky guesses on hash functions. Specifically, in P_1 there are cases where an unpaired client instance Π_C^i may accept a confirmation code k, but the adversary has not asked the required random oracle queries to H_1 and H_2 in order to compute k, i.e. he proactively produced the correct one. The probability of this event happening is $\mathcal{O}(n_{ro} + n_{se})/q$. A similar scenario occurs when considering an unpaired server instance. Then:

$$\mathrm{Adv}_{P_1}^{\mathrm{FtG}}(\mathcal{A}) = \mathrm{Adv}_{P_2}^{\mathrm{FtG}}(\mathcal{A}) + \frac{\mathcal{O}(n_{ro} + n_{se})}{q}.$$

Game $\mathbf{G_3}$: Randomize session keys for Execute queries.
Consider an adversary $\mathcal{A}$ who eventually makes an $\mathsf{Execute}(C, i, S, j)$ query, resulting in the messages m and μ being exchanged and sk the session key established at Π_C^i and Π_S^j instances. Let P_3 be defined exactly as P_2, except that whenever $\mathcal{A}$ makes an $H_l(C, S, m, \mu, \sigma, \gamma')$ query – for $l \in \{1, 2, 3\}$ and $\pi_c = \pi[C, S]$ the shared password between C and S – there is no check for a **testexecpw**(C, i, S, j, π_c) event. As a result of this change, even if **testexecpw**(C, i, S, j, π_c) event is triggered, the simulator will answer $H_l(C, S, m, \mu, \sigma, \gamma')$ queries with **Rule** $\mathbf{H}_l^*$, i.e. with a random string independent of the established sk, the password and any previously exchanged messages. Note that games P_2 and P_3 are indistinguishable if **testexecpw** does not occur.

Next we will demonstrate that the probability of **testexecpw**(C, i, S, j, π_c) event happening is bounded by the Gap-DH assumption.

Claim 6.3. *For all adversaries $\mathcal{A}$ running in time t, there exists a Gap-DH solver algorithm $\mathcal{B}^{\mathcal{A}}$ with running time $t'' = \mathcal{O}(t + (n_{ro} + n_{se} + n_{ex}) \cdot t_{exp} + n_{ro} \cdot t_{ddh})$, such that:*

$$\mathrm{Adv}_{P_2}^{\mathrm{FtG}}(\mathcal{A}) \leq \mathrm{Adv}_{P_3}^{\mathrm{FtG}}(\mathcal{A}) + 2\mathrm{Adv}_{\mathbb{G}_1}^{\mathtt{gap-DH}}(\mathcal{B}^{\mathcal{A}}).$$

Proof: Let ϵ be the probability that **testexecpw** occurs in P_3. We build an algorithm $\mathcal{B}^{\mathcal{A}}$ whose goal is to solve the Gap-DH problem using adversary $\mathcal{A}$ as a subroutine on a simulation of the protocol P_3 and with the same success probability ϵ. On input two group elements $(X, Y) \in \mathbb{G}_1^2$, $\mathcal{B}^{\mathcal{A}}$ simulates P_3 to $\mathcal{A}$ with the following changes:

1. For every $\mathsf{Execute}(C, i, S, j)$ query, set $m = X \cdot g^{\rho_{i,C}}$, $\mu = Y \cdot g^{\rho_{j,S}}$, where $(\rho_{i,C}, \rho_{j,S}) \xleftarrow{\$} \mathbb{Z}_q^2$ are known to the simulator. Then set $k := H_1^*(\nu)$, $k' := H_2^*(\nu)$ and $sk_C^i \leftarrow sk_S^j := H_3^*(\nu)$, where $\nu = C||S||m||\mu$.

2. Each time $\mathcal{A}$ asks for a $H_l(C, S, m, \mu, \sigma, \gamma')$ query – where values m, μ were generated in $\mathsf{Execute}(C, i, S, j)$ query, and a $H(\pi_c)$ query returned $(\gamma')^{-1}$ – $\mathcal{B}^{\mathcal{A}}$ calls DDH-O with input $(m \cdot \gamma', \mu, \sigma)$. Once DDH-O returns 1, the "winning" H_l query is identified, $\mathcal{B}^{\mathcal{A}}$ computes

Z value as follows:

$$Z = \sigma \left(X^{\rho_{j,S}} \cdot Y^{\rho_{i,C}} \cdot g^{\rho_{i,C} \cdot \rho_{j,S}} \cdot \mu^{\psi_1[\pi_c]} \right)^{-1}, \tag{6.3}$$

and submits Z as solution to his (X, Y) challenge and stops.

The advantage of $\mathcal{B}^{\mathcal{A}}$ in solving Gap-DH is equal to ϵ and the running time of the reduction is $t'' = \mathcal{O}(t + (n_{se} + n_{ex} + n_{ro}) \cdot t_{exp} + n_{ro} \cdot t_{ddh})$, where t_{exp} and t_{ddh} represent the time for an exponentiation and deciding DDH in $\mathbb{G}_1$ respectively. It follows immediately that $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_2}(\mathcal{A})\right] \leq \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_3}(\mathcal{A})\right] + \epsilon$, and then by Fact 4.3 $\mathrm{Adv}^{\mathrm{FtG}}_{P_2}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_3}(\mathcal{A}) + 2\epsilon$. □

Game $\mathbf{G_4}$: Check password guesses.
The challenger executes P_3, except that if **correctpw** event occurs, then the protocol execution aborts and the adversary succeeds.

As a consequence, before any Corrupt query, whenever the simulator detects (via oracle queries) that the adversary uses the correct password to compute the confirmation codes k, k' or session key sk, the protocol will be aborted and the adversary will be deemed successful, i.e., no *unpaired* client or server instance will terminate prior to **correctpw** event or Corrupt query.

$$\mathrm{Adv}^{\mathrm{FtG}}_{P_3}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_4}(\mathcal{A}).$$

Game $\mathbf{G_5}$: Randomize session keys for paired instances.
Let P_5 be identical to P_4, except that in case the **pairedpwguess** event occurs, the protocol stops and adversary fails.

In this particular reduction, we will show that an adversary $\mathcal{A}$ who i) can adaptively corrupt users (thus knowing the password π_c) and ii) manages to compute sk for *paired* instances Π^i_C and Π^j_S, could be used as a subroutine to solve the Gap-DH problem.

Claim 6.4. *For all adversaries $\mathcal{A}$ running in time t, there exists a Gap-DH solver algorithm $\hat{\mathcal{B}}^{\mathcal{A}}$ running in time $t'' = \mathcal{O}(t + (n_{se} + n_{ro} + n_{exe}) \cdot t_{exp} + (n_{se} + n_{ro}) \cdot t_{ddh})$ such that:*

$$\mathrm{Adv}^{\mathrm{FtG}}_{P_4}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_5}(\mathcal{A}) + 2\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\hat{\mathcal{B}}^{\mathcal{A}}).$$

Proof: If **pairedpwguess** does not occur, then games P_4 and P_5 are indistinguishable. Let ϵ be the probability that **pairedpwguess** event occurs, when $\mathcal{A}$ is running in G_5.

Next, we will construct an algorithm $\hat{\mathcal{B}}^{\mathcal{A}}$ whose goal is to solve the Gap-DH problem using adversary $\mathcal{A}$ as subroutine on a simulation of the protocol P_5. On input $(X, Y) \in \mathbb{G}^2$, $\hat{\mathcal{B}}^{\mathcal{A}}$ simulates P_5 to $\mathcal{A}$ with the following changes:

1. In CLIENT ACTION 0 query to Π^i_C and input S, set $m = X \cdot g^{\rho_{C,i}}$ where $\rho_{C,i} \xleftarrow{\$} \mathbb{Z}_q$.
2. In SERVER ACTION 1 query to Π^j_S and input $\langle C, m \rangle$, set $\mu = Y \cdot g^{\rho_{S,j}}$, where $\rho_{S,j} \xleftarrow{\$} \mathbb{Z}_q$.
3. In CLIENT ACTION 1 to Π^i_C and input $\langle \mu, k \rangle$, if Π^i_C is unpaired, $\hat{\mathcal{B}}^{\mathcal{A}}$ first verifies k using DDH-O and the list of random oracle queries. If k is correctly constructed (DDH-O outputs 1), Π^i_C outputs k' and terminates, or rejects otherwise.

4. In SERVER ACTION 2 query to Π_S^j with input k', if Π_S^j was paired after its SERVER ACTION 1 but is now unpaired, then $\hat{\mathcal{B}}^{\mathcal{A}}$ verifies k'. If k' is correctly constructed, then Π_S^j terminates. Otherwise, it rejects.

5. After $\mathcal{A}$ terminates, the simulator selects queries of the form $H_l(C, S, m, \mu, \sigma, \gamma')$, for which the following conditions are satisfied: i) m and μ generated by some instances Π_C^i and Π_S^j respectively, ii) Π_C^i is paired with Π_S^j and Π_S^j is paired with Π_C^i after SERVER ACTION 1, iii) $(\gamma')^{-1} = H_1(\pi_c)$. For every such query, $\hat{\mathcal{B}}^{\mathcal{A}}$ calls DDH-O with input $(m \cdot \gamma', \mu, \sigma)$.

   Once DDH-O returns 1, $\hat{\mathcal{B}}^{\mathcal{A}}$ computes Z value in the same way as for G_3 (Equation 6.3), submits it as solution for his challenge and stops.

The advantage of $\hat{\mathcal{B}}^{\mathcal{A}}$ in solving Gap-DH is equal to ϵ and its running time is $t'' = \mathcal{O}(t+(n_{se}+n_{ro}+n_{ex})\cdot t_{exp}+(n_{se}+n_{ro})\cdot t_{ddh})$, where t_{exp} and t_{ddh} represent the time for an exponentiation and deciding DDH in $\mathbb{G}_1$ respectively. It follows from Fact 4.3, $\mathrm{Adv}^{\mathrm{FtG}}_{P_2}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_3}(\mathcal{A})+2\epsilon$. □

Remark 6.5. *To explain why the original reduction from [Mac02a] contains the term n_{se} as degradation factor, and how we can avoid such degradation in ours, consider the following scenario:*

Suppose that the adversary $\mathcal{A}$ against protocol P_4 first makes a CLIENT ACTION 0 query to Π_C^i and receives as an answer $m = X \cdot g^{\rho_{C,i}}$ value in which Diffie-Hellman challenge X is planted. Next, $\mathcal{A}$ obtains $\pi_c = \pi[C, S]$ via Corrupt*(S) query. With this information, $\mathcal{A}$ may decide to impersonate S to C by making a CLIENT ACTION 1 query with an input $\langle \mu, k \rangle$ to Π_C^i. Since $\mathcal{A}$ knows the correct password, she could compute and send the correct confirmation code k; however, $\mathcal{A}$ could also choose to send an incorrect one. Now, the simulator faces a problem: Π_C^i has to verify k and based on the verification outcome either accept or reject. Put differently, the simulator is unable to verify whether* ***testpw****$(C, i, S, \pi_c, l = 2)$ is triggered; this could be done by checking if $\sigma = DH(\alpha, \mu)$, but the simulator does not know the discrete log of X as it is his own challenge.*

To circumvent this obstruction, the reduction in [Mac02a] has to guess an instance that will be the target of the Test *query: this provides guarantee that there won't be any corruption before session keys are accepted, and thus the simulator can safely plant the received Diffie-Hellman challenge (X, Y) in the* Test *session. This technique yields a factor of n_{se} in front of $\mathrm{Adv}^{\mathsf{CDH}}_{\mathbb{G}}$ advantage in Theorem 6.1. In contrast, by using Gap-DH groups, our simulator can query DDH-O with input (α, μ, σ) to verify if $\sigma = DH(\alpha, \mu)$ and check whether the event* ***testpw****$(C, i, S, \pi_c, l = 2)$ is triggered or not. Hence, we can avoid guessing of the* Test *instance, which makes our reduction tight with respect to the success probability. Compared to [Mac02a], the running time of the reduction algorithm has increased by an additive term $(n_{se} + n_{ro})t_{ddh}$, due to the invocation of DDH-O needed for the simulator to identify correct random oracle queries.*

Game $\mathbf{G_6}$: Forbid two password guesses per online attempt on server.
In this game we restrict an adversary, who tries to masquerade as a client, from testing two passwords per session, say π_1 and π_2, in an online dictionary attack. Concretely, let P_6 be identical

to P_5, except that if **doublepwserver** event occurs, the protocol halts and the adversary fails. We assume that the check for **doublepwserver** occurs before the check for **pairedpwguess**.

Next we demonstrate that the probability of **doublepwserver** occurring is bounded by the Gap-DH assuption.

Claim 6.6. *For all adversaries $\mathcal{A}$ running in time t, there exists an Gap-DH solver algorithm $\tilde{\mathcal{B}}^{\mathcal{A}}$ running in time $t'' = \mathcal{O}(t + (n_{se} + n_{ro} + n_{exe}) \cdot t_{exp} + n_{ro} \cdot t_{ddh})$ such that:*

$$\mathrm{Adv}^{\mathrm{FtG}}_{P_5}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_6}(\mathcal{A}) + 4\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\tilde{\mathcal{B}}^{\mathcal{A}})$$

Proof: We construct an algorithm $\tilde{\mathcal{B}}^{\mathcal{A}}$ that attempts to solve the Gap-DH problem by running $\mathcal{A}$ as a subroutine on a simulation of the protocol P_6. On input two group elements $(X, Y) \in \mathbb{G}^2$, $\tilde{\mathcal{B}}^{\mathcal{A}}$ simulates P_6 to $\mathcal{A}$ with the following changes:

1. In $H(\pi)$ query, output $X^{\psi_1[\pi]} g^{\psi'_1[\pi]}$, where $\psi_1[\pi] \xleftarrow{\$} \{0,1\}$ and $\psi'_1[\pi] \xleftarrow{\$} \mathbb{Z}_q$.
2. In a SERVER ACTION 1 query to a server Π^j_S with input $\langle C, m\rangle$ where $acceptable(m)$ is true, set $\mu = Y \cdot g^{\rho'_{S,j}}$.
3. Once $\mathcal{A}$ terminates, the simulator $\tilde{\mathcal{B}}^{\mathcal{A}}$ creates a list L_c of $H_l(C, S, m, \mu, \sigma, \gamma')$ queries, with $l \in \{1,2,3\}$, such that $\sigma = DH(m \cdot \gamma', \mu)$, which can be checked using his DDH oracle. Then $\mathcal{B}^{\mathcal{A}}$ selects from the list L_c two different queries, say $H_l(C, S, m, \mu, \sigma, \gamma')$ and $H_{\hat{l}}(C, S, m, \mu, \hat{\sigma}, \hat{\gamma}')$, for $l, \hat{l} \in \{1,2,3\}$ such that there was i) a SERVER ACTION 1 query to a server instance Π^j_S with input $\langle C, m\rangle$ and output $\langle \mu, k\rangle$, ii) an $H(\pi)$ query that returned $(\gamma')^{-1}$, an $H_1(\hat{\pi})$ query that returned $(\hat{\gamma}')^{-1}$ and iii) $\psi_1[\pi] \neq \psi_1[\hat{\pi}]$. Then $\hat{\mathcal{B}}^{\mathcal{A}}$ outputs:

$$Z = \left(\sigma \cdot \hat{\sigma}^{-1} \cdot (\gamma')^{-\rho'_{S,j}} \cdot (\hat{\gamma}')^{\rho'_{S,j}} \cdot Y^{\psi'_1[\pi] - \psi'_1[\hat{\pi}]}\right)^{\psi_1[\pi] - \psi_1[\hat{\pi}]}, \tag{6.4}$$

   where $Z = DH(X, Y)$.

P_6 is indistinguishable from P_5 until the event **doublepwserver** occurs. Let the ϵ be the probability that **doublepwserver** occurs when $\mathcal{A}$ is running in G_5. When **doublepwserver** occurs for two passwords $\pi \neq \hat{\pi}$, the success probability of $\tilde{\mathcal{B}}^{\mathcal{A}}$ is $\epsilon/2$ and its running time is $t'' = \mathcal{O}(t + (n_{se} + n_{ro} + n_{exe})t_{exp} + n_{ro} \cdot t_{ddh})$. Thus, it follows from Fact 4.3 that $\mathrm{Adv}^{\mathrm{FtG}}_{P_2}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{FtG}}_{P_3}(\mathcal{A}) + 4\mathrm{Adv}^{\mathtt{gap-DH}}_{\mathbb{G}_1}(\tilde{\mathcal{B}}^{\mathcal{A}})$. □

DISCUSSION: This game shows that $\mathcal{A}$'s probability of simultaneously guessing (discarding) more than *one* password during a single online attempt on a server executing P_6 is negligible. In most PAKE proofs (in [Mac02a] too), this reduction typically brings the highest security degradation: e.g. $1/(n_{ro})^3$ appears in the case of Dragonfly [LS15] and SPEKE [Mac01a]. In contrast, our protocol only suffers from a constant loss (4) in the success probability.

The reason for $1/(n_{ro})^2$ degradation when using L-CDH in PAK reduction is the following: $\tilde{\mathcal{B}}^{\mathcal{A}}$ has to compute a list of *possible* DH values and then choose at random an element from the

list as possible solution to his CDH challenge. The list is computed as follows: for particular pairs of queries $H_l(C, S, m, \mu, \sigma, \gamma')$ and $H_{\hat{l}}(C, S, m, \mu, \hat{\sigma}, \hat{\gamma}')$, for $l, \hat{l} \in \{1, 2, 3\}$, $\tilde{\mathcal{B}}^{\mathcal{A}}$ computes Z as in Equation 6.4 and adds it to his list of *possible* DH values. The size of the list is upper bounded by $(n_{ro})^2$, resulting in unfeasible running time for $\tilde{\mathcal{B}}^{\mathcal{A}}$.

In contrast, by using Gap-DH groups, $\tilde{\mathcal{B}}^{\mathcal{A}}$ can identify the right pair of H_l queries (at the cost of at most $n_{ro} t_{ddh}$ in the running time) and then compute a single, correct Z value using Equation 6.4. As a result, we can remove the quadratic factor in the running time of the reduction.

Game $\mathbf{G_7}$: Internal password oracle.
In this final game we estimate the probability of the **correctpw** event occurring, i.e. the adversary guessing the correct password π_c.

Let P_7 be as P_6, except that there is an *internal password oracle* $\mathcal{O}_{pw}$ which generates all passwords during the initialization of the users. The simulator uses it to i) handle $\mathsf{Corrupt}$ queries and ii) test whether **correctpw** occurs. More specifically, when $\mathcal{A}$ asks $\mathsf{Corrupt}(\mathrm{U})$, the query is simply forwarded to $\mathcal{O}_{pw}$ which returns π_U if $U \in Clients$, otherwise returns $\langle \pi_U[C] \rangle_{C \in Clients}$. To determine whether **correctpw** occurs, the simulator queries $\mathcal{O}_{pw}$ with $\mathrm{test}(\pi, C)$, which returns TRUE if $\pi = \pi_C$ and FALSE otherwise. By definition P_6 and P_7 are perfectly indistinguishable. Then, it holds that $\mathrm{Adv}^{\mathrm{FtG}}_{P_6}(\mathcal{A}) = \mathrm{Adv}^{\mathrm{FtG}}_{P_7}(\mathcal{A})$.

Claim 6.7. *For all PPT adversaries $\mathcal{A}$, $\mathrm{Adv}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \leq n_{se}/|D|$.*

Proof. Let ψ denote the **correctpw** event and ψ^c its complement. The probability that $\mathcal{A}$ succeeds in G_7 is given by:

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] = \Pr\left[\text{correctpw}\right] \cdot \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \text{correctpw}\right] + \Pr\left[\neg\text{correctpw}\right] \cdot \Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\text{correctpw}\right]. \quad (6.5)$$

We look at the first term of Equation 6.5. Since there are n_{se} Send queries, the probability of **correctpw** occurring is bounded by $\Pr\left[\text{correctpw}\right] \leq n_{se}/|D|$. Additionally, it follows from G_4 that $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \text{correctpw}\right] = 1$. Now we look at the second term of Equation 6.5. Given that **correctpw** does not occur, $\mathcal{A}$ succeeds by making a Test query to a fresh instance Π^i_U and guessing the bit b used in the Test query. By examining Reveal and H_3 queries throughout the proof, it follows that the view of $\mathcal{A}$ is independent of the session key sk^i_U, thus $\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \mid \neg\text{correctpw}\right] = 1/2$. Putting everything together in Equation 6.5:

$$\Pr\left[\mathrm{Succ}^{\mathrm{FtG}}_{P_7}(\mathcal{A})\right] \leq \frac{1}{2} + \frac{n_{se}}{2 \cdot |D|}.$$

We finally obtain $\mathrm{Adv}^{\mathrm{FtG}}_{P_7}(\mathcal{A}) \leq n_{se}/|D|$. □

■

6.4 Conclusion

We proposed a new instantiation for the PAK protocol and showed that the security proof from [Mac02a] can be adapted to cover our proposal. Our reduction to the Gap Diffie-Hellman problem is significantly tighter than the previous reduction from the CDH problem. From a theoretical point of view, this shows that the security of PAK is closely related to the security of Gap-DH assumption. In terms of concrete security, the advantage of the tighter proof is that it provides the guarantee that with typical values of the group size for today, even the most computationally powerful adversaries today cannot do significantly better than an online dictionary attack.

In Table 6.1 we compare the *quality* of the reduction when the PAK protocol is instantiated in prime order groups $\mathbb{G}$ or Gap-DH groups $\mathbb{G}_1$. We observe that the running time of the reduction, when considering Gap-DH groups, is less than when considering the instantiation over prime order groups $\mathbb{G}$, the reason is that t_2 does not have the quadratic term $(n_{ro})^2$, but instead only an additive term $(n_{se} + n_{ro})t_{ddh}$. Probably more interesting, we obtained a tight-reduction with respect to the success probability. Specifically, let $\epsilon_{\mathcal{R}}$ and $\epsilon_{\mathcal{A}}$ denote the success probability of the reduction and the adversary $\mathcal{A}$ respectively. Let $\psi = \epsilon_{\mathcal{A}}/\epsilon_{\mathcal{R}}$ be the degradation factor induced by the reduction. When the PAK protocol is instantiated on Gap-DH groups, the degradation factor is given by a constant c – exactly $c = 8$, however, when considering prime order groups as in the original proposal, the degradation factor is $\mathcal{O}(n_{se} \cdot (n_{ro})^2)$.

Similar techniques could lead to tighter security proofs in other existing PAKE protocols, for instance PPK [Mac02b], SPAKE2 [AP05b], PFS-SPAKE2 (Section 5.3).

Table 6.1: Comparison of the running time and success probability of the reduction algorithm when using different variants of the CDH assumption in PAK. Variable t corresponds to the running time of $\mathcal{A}$, $t_1 = ((n_{ro})^2 + n_{se} + n_{ex})t_{exp})$ and $t_2 = (n_{ro} + n_{se} + n_{ex})t_{exp}) + (n_{se} + n_{ro})t_{ddh}$, where n_{ro}, n_{se}, n_{ex} represent the number of random oracle, Send and Execute queries respectively, t_{exp} represent the running time to compute an exponentiation – in $\mathbb{G}$ or $\mathbb{G}_1$ – t_{ddh} the time for deciding DDH and c is a constant.

Assumption	Group	Running time Simulation	Degradation Factor
CDH	$\mathbb{G}$	$\mathcal{O}(t + t_1)$	$\mathcal{O}(n_{se} \cdot (n_{ro})^2)$
Gap-DH	$\mathbb{G}_1$	$\mathcal{O}(t + t_2)$	c

CHAPTER 7

On the Relation between SIM and IND-based Security Models for PAKEs

7.1 Introduction

The cryptographic goal when designing PAKE protocols is to ensure that the attacker essentially cannot do better than an online dictionary attack. This goal recognizes that while online dictionary attacks cannot be avoided, offline dictionary attacks can and should be prevented. Numerous PAKE protocols have been designed to meet this goal but have later been found to be flawed [NCPW13, AP05a, Szy06, CH14, BŠŠ17]. Consequently, *security models* for PAKE have been devised to get assurance on the claimed security properties by performing a rigorous analysis.

We consider the provable security approach, where protocols are analyzed in a complexity-theoretic security model: the goal being that no reasonable algorithm can violate security under various hardness assumptions. The complexity-theoretic security models are classified into indistinguishability-based (IND-based) and simulation-based (SIM-based). In the IND-based approach security means that no probabilistic polynomial-time (PTT) adversary can distinguish an established session key sk from a random string, i.e. it guarantees semantic security on sk. The SIM-based approach defines two worlds: an *ideal world* which is secure by definition and the *real world* which is the real protocol execution against some PPT attacker. In the SIM-based setting, security asks for the indistinguishability between the ideal world and real world executions.

When dealing with formal security modeling of PAKE, the difference between the two previously mentioned approaches, IND and SIM, has practical consequences. It is accepted that IND-based models are easier to work with for protocol designers that wish to prove the security

of their protocols. In fact, currently, most of the security proofs for PAKEs are constructed under the IND-based models Find-then-Guess (IND-FtG) from [BPR00] and Real-or-Random (IND-RoR) from [AFP05]. In contrast, constructing security proofs in SIM-based models is considered more challenging. Two SIM-based models for PAKE that have seen wider use are Boyko, MacKenzie and Patel's (BMP) model [BMP00] that is derived from Shoup's SIM-based model for AKE [Sho99] and the Universal Composability (UC) framework of Canetti et al. [CHK+05] that follows the UC paradigm of Canetti [Can01]. While complex for constructing proofs of security, it is fair to recognize that SIM-based security i) offers a more intuitive and natural approach to defining security, ii) it is simpler to describe and interpret the security properties captured by the model, iii) SIM-secure protocols are well suited to accommodate secure composition results, and iv) it is possible to prove security of PAKE protocols even in the case of correlated passwords that may come from arbitrary password distributions.

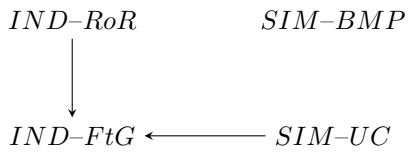

Figure 7.1: Previously known relations between PAKE security definitions.

The known relations between PAKE security definitions are summarized in Figure 7.1. In particular, to the best of our knowledge, no work has been done to formally analyze the relation between the IND-RoR and SIM-BMP security notions for PAKE protocols.[1] As we can see in Figure 7.1, the only existing result that is known to hold between IND and SIM based definitions is the one from [CHK+05]. There, the authors show that their SIM-UC definition implies the IND-FtG definition from [BPR00].

In practical terms, the lack of comparison results between IND-based and SIM-based models for PAKEs means that the security of PAKE protocols, such as SPEKE, that have been studied in the SIM-BMP simulation model of [BMP00] can not be compared with other PAKE protocols that are secure according to the SIM-UC or IND definitions.

Forward Secrecy. Commonly referred as Perfect Forward Secrecy (PFS), it is a security property for AKE and PAKE protocols, which asks to preserve the secrecy of previously established session-keys even if password-related information gets later compromised. It is a highly desirable security property specially for PAKEs as unfortunately, there exist in real life different ways in which the adversary could obtain such password information e.g. via phishing attacks a cheated client could reveal his password to some malicious entity or the data base storing the client's password at the server could get compromised resulting in massive password leakage [Cam17, PG14, Ian12].

[1] The result by Shoup [Sho99] on the equivalence between IND-FTG model and SIM model for authenticated key exchange with a high-entropy long-term secret does not carry over to the PAKE setting. The reason for this is that there is a non-negligible upper bound on the advantage of the adversary in IND-based security definitions for PAKE.

The intuition of forward secrecy was first mentioned by Diffie et al. in [DVOW92]. It was later formalized and incorporated in AKE [Sho99, LLM07, CK01, Kra05] and PAKE [BPR00, KOY02] security models. This formalization enhanced the understanding of forward secrecy by identifying distinct means in which a principal can get compromised and the information revealed to the adversary in such a case. However, it produced a number of definitions and variations on forward secrecy which might make it difficult to tell under which circumstances protocol "P" is *fs-secure*. For example, just in [BPR00] the authors provide three different definitions for forward secrecy.

7.1.1 Our Contribution

Our contributions in this chapter can be summarized as follows:

- We first reconcile the syntactic differences between the IND-RoR and SIM-BMP models for PAKE thus allowing honest comparison between them. More specifically, we slightly modify the initialization procedure of the IND-RoR model [AFP05] such that it follows the SIM-BMP model.
- We incorporate forward secrecy into the SIM-BMP and IND-RoR security models. We consider only the *weak corruption model* as defined in [BPR00], which is the most used type of forward secrecy.
- We prove that SIM-BMP security implies IND-RoR security and that IND-RoR security is equivalent to a slightly modified version of SIM-BMP security adapted to the model of [GL01a]. We also investigate whether IND-RoR security implies (unmodified) SIM-BMP security.

7.1.2 Related Work

Within the AKE research community, it is often stated that there exist more security models than AKE protocols. Thus, it seems convenient to establish how the most relevant security models relate to one another. In this section, we recall the security models for AKEs and PAKEs which we consider are the most relevant.

Authenticated Key Exchange (AKE). In 1993, Bellare and Rogaway formalize the notion of security for AKE protocols [BR93a]. In their model, an AKE protocol is secure if, under the allowed adversary actions, the established session key is computationally indistinguishable from a random string – this corresponds to the IND-based approach. After this initial work, numerous others have appeared studying the cryptographic security for AKE protocols following the IND-based approach [CK01, LLM07, BR95, BM97, BFWW11, JKSS12]. It was believed that the eCK model [LLM07] was stronger than the previously proposed CK model [CK01], however, in an effort to understand the exact relation among these models, Cremers demonstrated that they are incomparable [Cre11]. The first simulation (SIM) definition for AKE was given by Bellare, Canetti and Krawczyk [BCK98]. In 1999, Shoup proposed another security model for AKE protocols in the SIM-based setting [Sho99] and informally compared his model with the one

from [BCK98]. In the same work, the author gave a sketch of a proof arguing that SIM-security against both *static* and *adaptive* adversaries is equivalent to the corresponding IND-security notions of [BR95]. Canetti and Krawczyk in [CK02] took SIM definitions further by expanding the composition guarantees of AKE from [Sho99] to arbitrary protocols within the Universal Composability (UC) framework of Canetti [Can01].

Password Authenticated Key Exchange (PAKE). The first adequate security models for PAKE appeared in [BPR00] and [BMP00] around the same time. Both models were built upon already existing AKE models. Although the SIM-based model from [BMP00] has been used to prove secure several PAKE protocols (PAK [BMP00], RSA-based SNAPI [MPS00b], and SPEKE [Mac01a]), it is the IND-FtG model from [BPR00] that has established itself as the model of choice when analyzing PAKEs. Using the IND-FtG model, Katz et al. [KOY01] managed to achieve a breakthrough: they have shown how one can *efficiently* realize PAKE without random oracles, but instead relying on a common reference string (CRS). In more theoretical work, Goldreich and Lindell [GL01a] proposed a PAKE in the standard model that follows the simulation tradition. A few years later, Abdalla et al. [AFP05] showed that a stronger model than IND-FtG is necessary when trying to achieve three-party PAKE. Hence, they proposed a new model, known as the IND-RoR model, which is proven to be stronger than the IND-FtG model in the case of PAKE. Recently, Škrobot and Lancrenon [SL18] have shown that the IND-FtG model may not be enough when looking at composition between PAKEs and arbitrary symmetric key protocols (SKP). However, on the positive side, they have shown that IND-RoR secure PAKE protocols with weak forward secrecy can be safely composed with arbitrary, higher-level SKPs. For these reasons, the IND-RoR model – enriched to handle forward secrecy – is considered the state-of-the-art model and has been used in the analysis of most recent PAKE protocols [ABM15, LST16]. Another prominent model in PAKE research is the Universal Composability (UC) framework for PAKE of Canetti et al. [CHK+05]. This model has been recently extended to treat strong augmented PAKEs [JKX18]. For more relevant papers on PAKE, we refer the reader to Pointcheval's survey [Poi12].

7.2 The Real or Random Security Model for PAKEs

The Real-or-Random (IND-RoR) security model for 2-party PAKE was introduced by Abdalla, Fouque and Pointcheval in [AFP05]. It is a refinement of IND-FtG model in which the adversary has access to multiple Test queries instead of a single one. In this section, we present a stronger version of the original model that allows us to explicitly incorporate the requirement of forward secrecy. Before we recall the IND-RoR model with forward secrecy, we introduce the notation that will used in this chapter.

Notation. Adversaries (respectively, challengers) will be denoted $\mathcal{A}$ (resp. $\mathcal{CH}$) in the IND-RoR model and $\mathcal{B}$ (resp. $\mathcal{RM}$) in the SIM-BMP model. We write $A \stackrel{c}{\equiv} B$ to denote two computationally indistinguishable distributions.

7.2.1 Description of the IND-RoR Model with Forward Secrecy

The so-called IND-RoR model of Abdalla et al. [AFP05], defines security via a *game* played between a challenger $\mathcal{CH}$ and some adversary $\mathcal{A}$ whose goal is to distinguish *real* session keys from *random* strings. It follows from the Find-then-Guess (IND-FtG) model of [BPR00], however, the IND-RoR model allows $\mathcal{A}$ to ask *multiple* Test queries to different instances while the IND-FtG restricts $\mathcal{A}$ to a *single* Test query. This simple yet important change results in the IND-RoR model being strictly stronger than the IND-FtG model in the PAKE setting. This is in contrast with the AKE scenario in which the two models are considered equivalent.

Recall that in [BPR00], several variants of the IND-FtG model are described: these models can be differentiated depending on the type of forward secrecy they are trying to capture. Nevertheless, the original IND-RoR model from [AFP05] does not include any forward secrecy requirement. In this section, we present an stronger version of the original IND-RoR model to incorporate forward secrecy by following [KJP06, BPR00], which we will simply refer as FS-IND-RoR to differentiate from the original IND-RoR model. In addition to the treatment of forward secrecy, we will introduce a minor change to the IND-RoR and the SIM-BMP model to allow for meaningful comparison between them. Otherwise, the models would be syntactically incomparable. Whenever possible, we prefer to change the SIM-BMP model rather than the IND-RoR since the latter is more widespread.

PROTOCOL PARTICIPANTS. Each participant in a two party PAKE protocol is either a client $C \in \mathcal{C}$ or a server S. Let $\mathcal{U} = \mathcal{C} \bigcup S$ denote the set of all (honest) participants. Additionally, each *initialized* participant U is associated with a unique identifier id_U. During the execution of the protocol, there might be several running instances of each participant. A running instance i of some participant $U \in \mathcal{U}$ is called an *oracle instance* and is denoted by Π_U^i.

LONG-TERM SECRETS. Server S holds a password π for each client C, i.e. it holds a vector $L =< \pi_i >_{i \in \mathcal{C}}$. In the opposite direction, client C shares a single password π with server S. For simplicity let π also denote the function assigning passwords to pair of users. We will refer to $\pi[id_C, id_S]$ as the password shared between client C and server S. Note that $\pi[id_C, id_S] = \pi[id_S, id_C]$, while $\pi[id_C, id_C]$ is not allowed in the model. The passwords are assumed to be independent and uniformly distributed.

PROTOCOL EXECUTION. Protocol P is an algorithm that describes how participants behave in response to inputs from their environment. Each participant can run P in parallel with different partners, which is modeled by allowing an unlimited number of *instances* of each participant to be created. We assume the presence of an adversary $\mathcal{A}$ who has full control over the network. She can enumerate, off-line, the words of the password directory D.

SECURITY EXPERIMENT IN FS-IND-ROR MODEL. Security in the IND-RoR model with forward secrecy is defined via a game played between the challenger $\mathcal{CH}$ and adversary $\mathcal{A}$. At the beginning of the experiment, $\mathcal{CH}$ tosses a coin and sets $b \in \{0, 1\}$ outside of $\mathcal{A}$'s view. Then $\mathcal{A}$ is given access to i) endless supply of user instances Π_U^i and ii) oracle queries to control them.

Oracle queries are answered by the corresponding Π_U^i according to P. $\mathcal{A}$'s goal is to find out the value of the hidden bit b. Next, we summarize the oracle queries $\mathcal{A}$ can access during the security experiment.

- **Initialize user**$(U, id_U, role_U)$. The adversary $\mathcal{A}$ assigns the string id_U as identity and $role_U \in \{client, server\}$ to user $U \in \mathcal{U}$, subject to the restriction that id_U has not been already assigned to another user. There are two cases:
  - If $role_U = server$ we simply write S instead of U. Then, for every initialized client $C \in \mathcal{C}$ with id_C, a password is picked uniformly at random from the dictionary D and assigned to the corresponding pair of client-server, i.e. $\pi[id_C, id_S] \xleftarrow{\$} D$.
  - In case $role_U = client$ we shall simply write C instead of U. Then, provided that S has already been initialized with id_S, do $\pi[id_C, id_S] \xleftarrow{\$} D$.
- **Initialize user instance**$(U, i, role_U^i, pid_U^i)$. An instance $i \in \mathbb{N}$ of an initialized user $U \in \mathcal{U}$ is created and denoted by Π_U^i. It is assigned i) a role $role_U^i \in \{open, connect\}$ and ii) a partner identity pid_U^i corresponding to the *identity* of some user U' that Π_U^i is supposed to communicate with in the future. The following constraint must hold:
  - $role_U$ and $role_{U'}$ are complementary, i.e. $role_U = server$ and $role_{U'} = client$ or the other way around.

  User instances are modeled as state machines with implicit access to the protocol description P and its corresponding password, i.e. some Π_U^i with $pid_U^i = id_{U'}$ is given access to $\pi[U, pid_U^i]$.
- **Send**(U, i, m). $\mathcal{A}$ sends message m to user instance Π_U^i. The latter behaves according to the protocol description, sends back the response m' to $\mathcal{A}$ (if any) and updates its state as follows:
  - continue: Π_U^i is ready to receive another message.
  - reject: Π_U^i aborts the protocol execution and sets the session key $sk_U^i = \perp$. This can be due to receiving an unexpected message m.
  - accept: Π_U^i holds pid_U^i, session identifier sid_U^i and sk_U^i. However, Π_U^i still expects to receive another message to fulfill the protocol specification.
  - terminate: Π_U^i holds pid_U^i, sid_U^i and sk_U^i. It has completed the protocol execution and will not send nor receive any other message.
- **Execute**(U, i, U', j). The transcript of the execution is returned to $\mathcal{A}$. It models the honest execution of the protocol between Π_U^i and $\Pi_{U'}^j$.
- **Corrupt**(U). $\mathcal{A}$ learns the long-term secret information of some initialized user U. If $role_U = client$, then $\mathcal{A}$ gets π_U. Otherwise, if $role_U = server$, then $\mathcal{A}$ receives $L =< \pi_i >_{i \in \mathcal{C}}$.

- **Test** (U, i). $\mathcal{A}$ asks for the session key of user instance Π_U^i. Provided that $status_U^i = terminate$, $\mathcal{CH}$ responds as follows [2]:
  - If there was a $\mathsf{Corrupt}(U^*)$ query – where U^* can be any user – and a Send query directed to Π_U^i before the sk is computed, then $\mathcal{A}$ gets the real sk of Π_U^i. Otherwise:
  - $\mathcal{CH}$ responds using the bit b. If $b = 1$ then $\mathcal{A}$ gets the real sk of Π_U^i, if $b = 0$ she gets a random string $r \xleftarrow{\$} \{0,1\}^\kappa$, where κ denotes the length of session keys. To ensure consistency, whenever $b = 0$ the same random string is returned for Test queries asked to two *partnered* instances.

Matching Instances. Two instances, Π_U^i and $\Pi_{U'}^j$, are matching instances if:

- $pid_U^i = id_{U'}$, $pid_{U'}^j = id_U$
- Users have complementary roles, i.e. one has role *client* and the other has role *server*.
- User instances have complementary roles, i.e. one instance has the role *open* and the other *connect*.

Partnering. Two matching instances Π_U^i and $\Pi_{U'}^j$ are *partners* if both instances *accept* – each holding pid_U^i, sid_U^i, sk_U^i and $pid_{U'}^j$, $sid_{U'}^j$, $sk_{U'}^j$ respectively – and the following holds:

- $sid_U^i = sid_{U'}^j$ and $sk_U^i = sk_{U'}^j$
- No oracle besides Π_U^i and $\Pi_{U'}^j$ accepts with some $sid' = sid_U^i$, except with negligible probability.

Advantage of the adversary. During the experiment, $\mathcal{A}$ is allowed to ask several Test queries directed to different oracle instances Π_U^i in the *terminate* state. All these queries are answered depending on the bit b chosen at the beginning of the experiment with either the real session key if $b = 1$ or a random string otherwise. At the end of the game, $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$, i.e. if she distinguished real session keys from random strings. The advantage of $\mathcal{A}$ in the FS-IND-RoR security game for protocol P and passwords sampled uniformly at random from dictionary D is defined as follows:

$$Adv_{P,D}^{FS-RoR}(\mathcal{A}) := 2 \cdot \Pr(b' = b) - 1. \tag{7.1}$$

Definition 7.1. *Protocol P is FS-IND-RoR secure if*

1. *(Completeness) If protocol messages are faithfully transmitted between two matching instances then both instances accept and compute the same key.*
2. *(Bounded Adversary Advantage) For all PPT adversaries $\mathcal{A}$:*

$$Adv_{P,D}^{FS-RoR}(\mathcal{A}) \leq \frac{n}{|D|} + negl(\kappa) \ , \tag{7.2}$$

[2] This is commonly referred as *freshness* condition.

where n is an upper bound on the number of sessions initialized by $\mathcal{A}$ and κ is the security parameter.

Remark 7.2. *As we mentioned before, different flavors of forward secrecy exist in the literature, e.g. just in [BPR00] the authors provide three particular definitions which could either weaken or strengthen the security guaranteed by the model in case of compromise of long-term secret information. While the intuition of forward secrecy and the security guarantee that it aims to provide are understood, it is unclear which definition of forward secrecy is de facto the right one for PAKE protocols. Therefore, to be explicit, we consider forward secrecy in the weak corruption model described in [BPR00], where corruption of some principal leaks only its password to the adversary, i.e. no internal state is revealed.*[3]

In the Client-Server setting, it is reasonable to assume that compromise of the server leaks the whole password data file to the adversary, even for asymmetric PAKEs. Thus, the model pessimistically renders every instance, whose session key was negotiated after someone got corrupted, as compromised and no security is guaranteed. Such a case is formalized in the Test query, which is answered with the real *session key, i.e. independently of the bit b, whenever the previously mentioned scenario occurs. We note that it is possible to fine-tune the model by distinguishing compromise of a server from a client's one, however, it will place new cumbersome conditions to the Test query making the analysis more complex and without gaining some significant improvement.*

Remark 7.3. *When using passwords as means of authentication, there is a non-negligible probability of an adversary successfully impersonating an honest user by simply guessing its password. This problem is unavoidable and inherent to PAKE protocols. Consequently, the security definition considers a PAKE protocol to be secure if only on-line dictionary attacks are possible, i.e. the protocol should not leak any information that allows the adversary to obtain the password in an off-line manner.*

7.3 Security in the Simulation Model

Security in the *simulation* approach, also known as *real-ideal* paradigm, was first described in the seminal work of Goldreich, Micali and Wigderson on secure multi-party compuation [GMW87]. Their intuition is the following: To evaluate the security of a cryptographic protocol for some *functionality* – say password-based authenticated key-exchange – one first defines an *ideal world* that accomplishes the functionality in a secure way. Thus the ideal world contains the specification and security requirements that the given functionality must fulfill. It is defined via a *trusted party* who receives inputs from the users, *locally* computes the outputs according to the specified functionality, and finally delivers to each user the corresponding value. Then, a protocol is secure

[3] We note that the definition of forward secrecy in the weak corruption model – used in this Chapter – is different to that of weak forward secrecy (see Chapters 4 and 5).

in the real-ideal paradigm, if whatever the adversary can do in the real execution of the protocol, can also be done in the ideal world, which is secure by definition.

When dealing with passwords as long-term secret information for authentication, the security model has to acknowledge the non-negligible probability of an adversary guessing the correct password and successfully impersonating an honest user. There are two ways to incorporate this *defect* due to the low entropy of passwords in the SIM-based security model; the first approach is considered in [BMP00, CHK$^+$05] while the second in [GL01a, NV08]:

1. Incorporate the non-negligible probability of an adversary guessing the password into the ideal world, by explicitly allowing the ideal world adversary to verify the guess of a candidate password. Then one defines a protocol to be secure if the real world execution is computationally indistinguishable from an execution in the ideal world.

2. Do not allow password guessing in the ideal world, but relax the requirement of indistinguishability between the real world and ideal world transcripts. One defines a protocol to be secure as one whose real-world execution is distinguishable from an execution in the ideal world with probability at most $n/|D| + negl(\kappa)$, where n is the number of active user instances and D is the dictionary. Keep in mind that we make use of this approach in Section 7.4 when we prove Theorem 7.12.

For now we consider only the first approach. We adapt the original SIM-BMP model of Boyko et al. [BMP00] to account for scenarios where *forward secrecy* is required. For clarity, we refer to the later simulation model with forward secrecy as FS-SIM-BMP to distinguish from the original one. The inclusion of this security property in the SIM-BMP model allows us to provide a fair comparison to the IND-RoR model with forward secrecy as described in Section 7.2, otherwise, the models would be incomparable simply because they aim for different security guarantees. We consider forward secrecy in the *weak corruption model* as described in [BPR00, Sho99] for this task.

7.3.1 The Ideal World

The ideal world (IW) model describes the service that a PAKE aims to provide, i.e. to allow parties to jointly compute a high entropy secret session key, which can be used later in higher level *applications*. In the IW there are no messages flowing around the network nor cryptography. The session keys are chosen at random by a trusted party and delivered out-of-band to the honest users. More formally, the ideal world involves interaction between a trusted entity called ideal world *Ring Master* and an ideal world adversary, denoted by $\mathcal{RM}^*$ and $\mathcal{B}^*$ respectively. The *ring master* is similar to the *challenger* in the FS-IND-RoR experiment. The details of the ideal world execution follow.

PROTOCOL PARTICIPANTS: As defined in the FS-IND-RoR model.

LONG-TERM SECRETS: The FS-SIM-BMP model does not make any assumption on the password distribution. However, to allow a fair comparison to the FS-IND-RoR model, we assume

the passwords to be independent and uniformly distributed.

PROTOCOL EXECUTION: There is no protocol execution in the ideal world. The session key of an instance is generated by the $\mathcal{RM}^*$ when $\mathcal{B}^*$ asks that instance the *start session* query. Additionally $\mathcal{B}^*$ is given access to the following oracles:

- **Initialize user**$(U, id_U, role_U)$. Identical to that in the FS-IND-RoR model.
  [Transcript:("init. user", $U, role_U$)]
- **Initialize user instance**$(U, i, role_U^i, pid_U^i)$. Identical to that in the FS-IND-RoR model.
  [Transcript: ("init. inst.",U, i, pid_U^i)][4]
- **Abort user instance** (U, i) Adversary $\mathcal{B}^*$ asks $\mathcal{RM}^*$ to abort user instance Π_U^i. We say then that Π_U^i is *aborted*.
  [Transcript: ("abort. user inst.",U, i)]
- **Test instance password** (U, i, π'). For user instance Π_U^i and password guess π', $\mathcal{B}^*$ queries if π' equals $\pi(U, pid_U^i)$. If this is true, the query is called *successful guess on* $\{U, pid_U^i\}$.

  This query can be asked only once per user instance. The user instance must be initialized and not yet engaged in a session, i.e. no start session operation has been performed for that instance. Note that $\mathcal{B}^*$ is allowed to ask a *test instance password* query to an instance that is *aborted*. This query does not leave any records in the transcript.
- **Corrupt**(U). $\mathcal{B}^*$ learns the long-term secret information of some initialized user U. If $role_U = client$, then $\mathcal{B}^*$ gets π_U. Otherwise, if $role_U = server$, then $\mathcal{B}^*$ receives $L =< \pi_i >_{i \in \mathcal{C}}$.
  [Transcript: ("Corrupt",U, π_U)]
- **Start session**(U, i). $\mathcal{B}^*$ specifies that a session key for user instance Π_U^i must be generated, by specifying one of the three *connection assigments* available:
  - **open for connection from** (U', j). This operation is allowed if: c1) $role_U^i = open$ and user instances Π_U^i and $\Pi_{U'}^j$ are *matching instances*, c2) $\Pi_{U'}^j$ has been *initialized* and not *aborted*, c3) no other instance is *open for connection* from $\Pi_{U'}^j$ and c4) no *test instance password* operation has been performed on Π_U^i. Then $\mathcal{RM}^*$ generates session key sk_U^i at random. Then Π_U^i is said to be *open for connection from* $\Pi_{U'}^j$.
  - **connect to** (U', j). This operation is allowed if: c1) $role_U^i = connect$ and user instances Π_U^i and $\Pi_{U'}^j$ are *matching instances*, c2) $\Pi_{U'}^j$ has been *initialized* and not *aborted*, c3) $\Pi_{U'}^j$ was open for connection from Π_U^i after Π_U^i was initialized and $\Pi_{U'}^j$

[4]Note that the original SIM-BMP model [BMP00] also places $role_U^i$ in the transcript, but we have chosen to remove it. This is because in the ideal world, from two partnered instances, the one with the role "open" will always start session first. On the other hand, in the real world, the adversary is free to choose which instance is assigned role "open" and which "connect". Thus, a real world adversary could make an honest execution of a protocol between an instance with role "connect" that terminates first, and an instance with role "open" that terminates second. Such a transcript, which constitutes an honest execution of a protocol, would not be simulatable in the ideal world if the roles "open" and "connect" are placed in the transcript.

is still open for connection and c4) no *test instance password* operation has been performed on Π_U^i. The $\mathcal{RM}^*$ sets $sk_U^i = sk_{U'}^j$ and $\Pi_{U'}^j$ is no longer open for connection.

– **expose** (U, i, sk). $\mathcal{B}^*$ assigns session key sk of his choice to user instance Π_U^i. This connection assignment is allowed if at least one of the following conditions hold: i) there has been a successful test instance password on Π_U^i or ii) there was a Corrupt query, directed to any user, before the *start session* operation.

[Transcript: ("start session", U, i)]

- **Application** (f, U, i). The adversary specifies an efficiently computable function f and a user instance Π_U^i for which a session key sk_U^i has already been established. It gets back $f(\{sk_U^i\}, R)$, where R is a global random bit string which user instances are given access to. R is not correlated to the established session keys and usually is referred to as the environment.
  [Transcript: ("application", $f, f(sk_U^i, R)$)]
- **Implementation**. This is a do nothing operation. $\mathcal{B}^*$ is allowed to place *implementation* operations without taking any effect in the ideal world. It is needed to allow $\mathcal{B}^*$ to construct *transcripts* that are equivalent to those in the real world.
  [Transcript: ("impl", $cmmt$)]

Transcript. Some of the previously mentioned queries are recorded in a *transcript*. Let IWT^* denote the transcript generated by $\mathcal{B}^*$.

Remark 7.4. *The SIM-BMP model handles on-line dictionary attacks, which are unavoidable and inherent to PAKEs, by introducing the notion of passwords and specifically the* Test instance password *query in the ideal world definition. This approach places the fundamental requirement that an active adversary can test at most one password per protocol execution. In fact, provided that the PAKE in question should be deemed SIM-BMP secure, the test instance password allows the simulator to create ideal world transcripts which are computationally indistinguishable from real world ones.*

In a more general sense, the expose connection assignment is allowed whenever the adversary could compute by his own the session key shared with some instance Π_U^i, e.g. a successful online dictionary attack or a Corrupt *query asked before the connection assignment. This is similar to the freshness condition defined for IND-based models, which prevents the adversary from winning the experiment by trivial means.*

The purpose of running PAKE protocol is to later use the established session keys in higher-level application protocols, e.g. the construction of secure communication channels is their most natural application. However, partial information about the established session key could potentially be leaked to the adversary through the usage of such keys, e.g. cryptanalysis, side channel attacks, etc. The application query models the ability of the adversary to get any information she wishes about the environment and the established session keys. The function f is defined by $\mathcal{B}^*$, the only constraint is that it must be efficiently computable.

7.3.2 The Real World

The real-world (RW) describes the scenario where a PAKE protocol runs. There is a real world Ring Master ($\mathcal{RM}$), whose role is similar to the role of the challenger in the FS-IND-RoR experiment, and a real-world adversary $\mathcal{B}$ who tries to attack the PAKE.

PROTOCOL PARTICIPANTS: Identical to IW.

LONG-TERM SECRETS: Identical to IW.

PROTOCOL EXECUTION: The same as in the FS-IND-RoR model. Also, user instances are defined as state machines with implicit access to id_U, pid_u^i and the corresponding password. The communication between the instances is entirely controlled by $\mathcal{B}$ via the following queries:

- **Initialize user**$(U, id_U, role_U)$. Identical to that in the FS-IND-RoR model.
  [Transcript:("init. user", $U, role_U$)]
- **Initialize user instance**$(U, i, role_U^i, pid_U^i)$. This is identical to that in the FS-IND-RoR model.
  [Transcript: ("init. inst.",U, i, pid_U^i)]
- **Send**(U, i, m). The same as in the FS-IND-RoR model except that the following is added to the transcript:
  [Transcript: ("impl", "msg", $U, i, m, m', state_U^i$)]. Additionally, the following record is added to the transcript depending on $state_U^i$.
  If $state_U^i$ = "terminate" add ("start session", U, i).
  If $state_U^i$ = "abort" add ("abort", U, i).
- **Corrupt**(U). The same as in IW.
  [Transcript: ("Corrupt",U, π_U)]
- **Application**(f, U, i). The same as in IW.
  [Transcript: ("application", $f, f(sk_U^i, R)$)]

Transcript. Let RWT be the transcript generated by $\mathcal{B}$. This is a sequence of records describing the actions of $\mathcal{B}$ when interacting with the real world protocol. $\mathcal{RM}$ generates $\mathcal{B}$'s random tape and places it in the first record of the transcript.
[Transcript: ("impl", "random tape", rt)].

Definition 7.5. *A protocol is FS-SIM-BMP secure if*

1. *(Completeness) If protocol messages are faithfully transmitted between two matching instances then both instances accept and compute the same key.*
2. *(Simulatability) for every efficient real-world adversary $\mathcal{B}$, there exists an efficient ideal world adversary $\mathcal{B}^*$, such that $RWT \stackrel{c}{\equiv} IWT^*$. Alternatively:*

$$\forall \mathcal{B}\ \exists \mathcal{B}^*\ \forall \mathcal{D}: \ |\Pr[1 \leftarrow \mathcal{D}(RWT)] - \Pr[1 \leftarrow \mathcal{D}(IWT^*)]| \leq negl(\kappa). \tag{7.3}$$

7.4 Relations between FS-IND-RoR and FS-SIM-BMP

In this section, we establish the relations between FS-IND-RoR and FS-SIM-BMP security models for PAKEs. We start by showing that FS-SIM-BMP security implies FS-IND-RoR security.

Table 7.1: Correspondence of $\mathcal{A}$'s and $\mathcal{B}$'s queries.

FS-IND-RoR	FS-SIM-BMP
Initialize user	Initialize user
Initialize user instance	Initialize user instance
Send	Send
Execute	Send
Corrupt	Corrupt
Test	Application

Theorem 7.6. *(FS-SIM-BMP Security ⇒ FS-IND-RoR Security). For any PAKE protocol P secure in the SIM-BMP model with forward secrecy, P is also secure in the IND-RoR model with forward secrecy.*

Proof. We demonstrate that if protocol P satisfies FS-SIM-BMP security, then the advantage of any adversary $\mathcal{A}$ in the FS-IND-RoR experiment is bounded by $n/|D| + negl(\kappa)$, where n is an upper bound on the number of sessions initialized by $\mathcal{A}$.

For clarity the proof is divided in two parts which we summarize here:

1. First we build a *real-world* adversary $\mathcal{B}^{\mathcal{A}}$ from $\mathcal{A}$. The intention is to generate a real-world *transcript* RWT according to the FS-SIM-BMP model but following $\mathcal{A}$'s commands. Additionally, since P is FS-SIM-BMP secure, the simulatability definition guarantees the existence of an ideal-world transcript IWT^* that is computationally indistinguishable from the RWT. Moreover, we show that one can use the previously generated RWT to instantiate again $\mathcal{A}$ and obtain *identical executions* of the previously simulated experiment to $\mathcal{A}$. The same reasoning applies when initializing $\mathcal{A}$ according to IWT^*.

2. We build a distinguisher $\mathcal{D}^{\mathcal{A}}$ using $\mathcal{A}$ as a subroutine, whose goal is to tell apart RWT from IWT^* transcripts. The distinguisher looks at whether $\mathcal{A}$ wins his security challenge when initialized with the given transcript. From this, we can bound the advantage of $\mathcal{A}$ in the FS-IND-RoR experiment to at most $n/|D| + negl(\kappa)$.

Concrete details of Part 1 and Part 2 follow:

Part 1. We construct $\mathcal{B}^{\mathcal{A}}$ using an $\mathcal{A}$ as a subroutine, where $\mathcal{B}^{\mathcal{A}}$ uses his own $\mathcal{RM}$ to answer $\mathcal{A}$'s queries. $\mathcal{B}^{\mathcal{A}}$ can perfectly simulate the FS-IND-RoR experiment to $\mathcal{A}$ (see Table 7.1). The objective is to generate a transcript RWT from the interaction $\mathcal{RM}$ vs $\mathcal{B}^{\mathcal{A}}$. The resulting transcript will be used in the second part of the proof.

Next we detail the construction of $\mathcal{B}^{\mathcal{A}}$, however, a reader familiar with FS-SIM-BMP and FS-IND-RoR security models could simply go to Table 7.1 and notice that $\mathcal{B}^{\mathcal{A}}$ can *perfectly* simulate the FS-IND-RoR experiment to $\mathcal{A}$.

- The interaction $\mathcal{RM}$ vs $\mathcal{B}^{\mathcal{A}}$ starts when the former initializes $\mathcal{B}^{\mathcal{A}}$ with random tape $rt_{\mathcal{B}}$ - as described in Section 7.3. Next $\mathcal{B}^{\mathcal{A}}$, who simulates the challenger $\mathcal{CH}$ in the FS-IND-RoR game, generates a uniformly distributed bit-string $rt_{\mathcal{A}}$ and initializes $\mathcal{A}$ with random tape $rt_{\mathcal{A}}$.
- $\mathcal{B}^{\mathcal{A}}$ sets $b \xleftarrow{\$} \{0,1\}$ outside $\mathcal{A}$'s view.
- $\mathcal{B}^{\mathcal{A}}$ uses his $\mathcal{RM}$ to answer $\mathcal{A}$'s queries: When $\mathcal{A}$ makes Initialize user, Initialize user instance or Send queries, $\mathcal{B}^{\mathcal{A}}$ simply forwards them to his $\mathcal{RM}$ and its response (if any) is forwarded back to $\mathcal{A}$. Execute queries asked by $\mathcal{A}$ are converted into Send queries appropriately. See Table 7.1.
- $\mathcal{B}^{\mathcal{A}}$ answers $\mathcal{A}$'s Test query using his Application query and the bit b. If there was a Corrupt and a Send query, then $\mathcal{B}^{\mathcal{A}}$ uses his Application query to reveal sk_U^i. Otherwise, if $b = 1$ then $\mathcal{B}^{\mathcal{A}}$ uses his Application query to reveal sk_U^i, however, if $b = 0$, then $\mathcal{B}^{\mathcal{A}}$ generates a random string $r \leftarrow \{0,1\}^{\kappa}$ and gives it to $\mathcal{A}$. In order to avoid strategies where $\mathcal{A}$ could trivially win the game, whenever $b = 0$ the same r is returned for Test queries asked to two *partnered* instances. [5]
- The experiment continues and $\mathcal{A}$ is allowed to make more queries as she wishes. Eventually, $\mathcal{A}$ outputs her guess b' and the FS-IND-RoR game finishes.
- $\mathcal{B}^{\mathcal{A}}$ makes an Application query and writes in the transcript the string "$b, rt_{\mathcal{A}}$". For the sake of the proof, it is not necessary to write the bit b' in the transcript.

The real-world transcript created is RWT. Furthermore, the FS-SIM-BMP definition guarantees the existence of a corresponding ideal-world transcript IWT^*, i.e. $\forall\mathcal{B}\ \exists\mathcal{B}^*$ such that $RWT \stackrel{c}{\equiv} IWT^*$.

Remark 7.7. *Given either RWT or IWT^*, it is possible create instances of $\mathcal{A}$ as needed, simulate to $\mathcal{A}$ the FS-IND-RoR experiment and obtain identical executions as recorded in the corresponding transcript. The reason is that $\mathcal{A}$ can be initialized with random tape $rt_{\mathcal{A}}$ contained in the transcript, and then $\mathcal{A}$'s behavior is deterministic and known in advance – given the corresponding transcript. Rewinding the adversary to a specific state is a standard proof technique [CGJ+99]. However, our requirement is simpler since we only need to initialize and run $\mathcal{A}$ from the beginning.*

Part 2. We use sequence of games and the previously generated transcript to demonstrate that FS-SIM-BMP Security $\Rightarrow$ FS-IND-RoR Security.

[5] In order to achieve sound simulation, we assume that partnering information is publicly computable [BFWW11].

Let G_0 be the experiment where $\mathcal{A}$ is initialized according to the real-world transcript RWT, i.e. a real-world adversary $\mathcal{B}^{\mathcal{A}}$ is simulating the FS-IND-RoR experiment to $\mathcal{A}$. Let S_0 be the event where $\mathcal{A}$ outputs $b' = b$ in G_0. It holds that $\Pr[S_0] = \Pr[\mathcal{A} \text{ wins } \mid t = RWT]$. Similarly, let G_1 be the experiment where $\mathcal{A}$ is initialized according to the ideal-world transcript IWT^*. Let S_1 be the event where $\mathcal{A}$ wins his FS-IND-RoR experiment in G_1; then it holds that $\Pr[S_1] = \Pr[\mathcal{A} \text{ wins } \mid t = IWT^*]$.

We first analyze the term $\Pr[\mathcal{A} \text{ wins } \mid t = IWT^*]$. Provided that FS-SIM-BMP security holds, we will then show that $\mathcal{A}$ can not notice the transition from G_0 to G_1, and this will give us a bound on the advantage of $\mathcal{A}$ in the FS-IND-RoR experiment.

We consider now the experiment G_1 and examine how the keys in IWT^* were generated. Let γ be the event that at least one sk is generated via *expose* connection assignment as a result of a *test instance password* query that occurs during the execution of $\mathcal{B}^*$ interacting with $\mathcal{RM}^*$, i.e. a successful online dictionary attack. Let β be the complement of γ, i.e. the event that no successful password guess occurred during the interaction of $\mathcal{B}^*$ and $\mathcal{RM}^*$.

Claim 7.8. $\Pr[\gamma] \leq n/|D|$.

Proof. For a single user instance, by definition of the ideal world, the probability of a successful password guess by $\mathcal{B}^*$ is $1/|D|$. We apply the union bound, and get that if there are at most n instances, $\Pr[\gamma] \leq n/|D|$. □

Claim 7.9. $\Pr[b = b' \mid \beta] = 1/2$.

Proof. Given than β occurs, the session keys placed in IWT^* were generated either by i) *expose* connection assignment – provided that there was a Corrupt query before the connection assignment – or ii) *open* or *connect* connection assignment. Then, whenever $\mathcal{A}$ makes a Test query to an instance whose session key was generated via case i), the simulator answers with the real sk computed at the tested instance, i.e. the answer is independent of the bit b by definition of the FS-IND-RoR experiment. Similarly, whenever $\mathcal{A}$ makes a Test query to an instance whose session key was generated via case ii), the simulator answers with a random string independent of the bit b. Therefore, the view of $\mathcal{A}$ is independent of the hidden bit b so $\Pr[b = b' \mid \beta] = 1/2$. □

Using Claim 7.8 and Claim 7.9 we get:

$$\Pr[\mathcal{A} \text{ wins} \mid t = IWT^*] = \Pr[(b' = b) \mid \gamma] \cdot \Pr[\gamma] + \Pr[(b' = b) \mid \beta] \cdot \Pr[\beta]$$
$$\Pr[\mathcal{A} \text{ wins} \mid t = IWT^*] \leq \frac{n}{|D|} \cdot \Pr[(b' = b) \mid \gamma] + \frac{1}{2} \cdot \left(1 - \frac{n}{|D|}\right)$$

Finally, by considering that $\Pr[(b' = b) \mid \gamma] \leq 1$ we obtain:

$$\Pr[\mathcal{A} \text{ wins} \mid t = IWT^*] \leq \frac{1}{2} + \frac{n}{2 \cdot |D|}. \tag{7.4}$$

Equation 7.4 expresses the observation that, by construction of the ideal-world, an adversary cannot do better than online dictionary attacks.

Now, we build a PPT algorithm $\mathcal{D}^{\mathcal{A}}$ whose aim is to distinguish real-word from ideal-world transcripts. $\mathcal{D}^{\mathcal{A}}$ gets as input a transcript $t \in \{RWT, IWT^*\}$, and uses it to initialize a PPT adversary $\mathcal{A}$ and simulate a FS-IND-RoR experiment to $\mathcal{A}$. The simulation will be either G_0 or G_1. If SIM-security holds, then $\mathcal{D}^{\mathcal{A}}$ cannot distinguish real world and ideal world transcripts, and so $\mathcal{A}$ cannot win his FS-IND-RoR experiment with advantage greater than $n/|D| + negl(\kappa)$.

In more details, on input some transcript t, $\mathcal{D}^{\mathcal{A}}$ proceeds as follows:

- Look for the last record of the transcript containing the string "$b, rt_{\mathcal{A}}$".
- $\mathcal{D}^{\mathcal{A}}$ "simulates" the challenger in the FS-IND-RoR experiment and initializes $\mathcal{A}$ on random tape $rt_{\mathcal{A}}$. Since $\mathcal{A}$ is given $rt_{\mathcal{A}}$, she behaves (deterministic) the same way as recorded in the transcript t. Every query asked by $\mathcal{A}$ can be answered by $\mathcal{D}^{\mathcal{A}}$ by just reading t.
- Eventually $\mathcal{A}$ outputs her guess b' and $\mathcal{D}^{\mathcal{A}}$ proceeds as follows: If $b = b'$ $\mathcal{D}^{\mathcal{A}}$ outputs "1" and if $b \neq b'$ it outputs "0". Additionally, when a bad event occurs, e.g. $\mathcal{A}$ cannot be initialized, or her queries cannot be answered by reading t, then $\mathcal{D}$ outputs $\perp$.

$\mathcal{A}$ wins her FS-IND-RoR game whenever she outputs $b' = b$. In such a case, $\mathcal{D}^{\mathcal{A}}$ outputs "1" by construction. Then it is true that:

$$\Pr[\, 1 \leftarrow \mathcal{D}(RWT)\,] = \Pr[\, S_0\,].$$

and

$$\Pr[\, 1 \leftarrow \mathcal{D}(IWT^*)\,] = \Pr[\, S_1\,].$$

From Equation 7.3 of FS-SIM-BMP security we know the following holds:

$$|\Pr[\, 1 \leftarrow \mathcal{D}(RWT)\,] - \Pr[\, 1 \leftarrow \mathcal{D}(IWT^*)\,]| \leq negl(\kappa). \tag{7.5}$$

Then it holds that $|\Pr[\, S0\,] - \Pr[\, S_1\,]| \leq negl(\kappa)$. By definition of G_0 and G_1:

$$|\Pr[\, \mathcal{A} \text{ wins } \mid t = RWT\,] - \Pr[\, \mathcal{A} \text{ wins } \mid t = IWT^*\,]| \leq negl(\kappa). \tag{7.6}$$

The term $\Pr[\, \mathcal{A} \text{ wins } \mid t = RWT\,]$ is actually the probability of $\mathcal{A}$ winning on a perfectly simulated FS-IND-RoR experiment. We combine with Equation 7.4 and get:

$$\Pr[\, \mathcal{A} \text{ wins} \mid t = RWT\,] \leq \frac{1}{2} + \frac{n}{2 \cdot |D|} + negl(\kappa)$$

We obtain that, if FS-SIM-BMP-security holds, then $\forall$ PPT $\mathcal{A}$ $Adv_{P,D}^{FS-RoR}(\mathcal{A}) \leq n/|D| + negl(\kappa)$, proving that FS-SIM-BMP $\Rightarrow$ FS-IND-RoR. ■

Now we investigate the reverse, i.e. whether FS-IND-RoR security also implies FS-SIM-BMP security. We obtain the following result:

Theorem 7.10. *If P is not FS-SIM-BMP secure, then $\exists \mathcal{A}$ s.t. $Adv_{P,D}^{FS-RoR}(\mathcal{A}) > n_A/|D| + \omega$, where n_A is the number of explicit password guesses of $\mathcal{A}$ and ω is a non-negligible function of the security parameter.*

Proof. We build a FS-IND-RoR adversary $\mathcal{A}^{\mathcal{B}}$, as the sequential composition of two adversaries: $\mathcal{A}_1$ and $\mathcal{A}_2^{\mathcal{B}}$. First, $\mathcal{A}^{\mathcal{B}}$ invokes $\mathcal{A}_1$. $\mathcal{A}_1$ tries a number of online dictionary attacks. If one of these is successful, then $\mathcal{A}^{\mathcal{B}}$ can win the FS-IND-RoR experiment. If none of the online dictionary attacks is successful, then $\mathcal{A}^{\mathcal{B}}$ invokes $\mathcal{A}_2^{\mathcal{B}}$. Next, we describe the details of $\mathcal{A}_1$ and $\mathcal{A}_2^{\mathcal{B}}$.
i) Construction of $\mathcal{A}_1$. Let $\mathcal{A}_1$ be an adversary who tries to masquerade user U to user V n_A times. Each time, $\mathcal{A}_1$ chooses a new candidate password and runs the protocol with V. If one of the password guesses is successful, then $\mathcal{A}_1$ can win the FS-IND-RoR experiment. By construction,

$$\Pr[\,\mathcal{A}_1 \;\; wins\,] = \frac{n_A}{|D|}. \tag{7.7}$$

ii) Construction of $\mathcal{A}_2^{\mathcal{B}}$. We have assumed that FS-SIM-BMP security does not hold. Then $\exists \mathcal{B}\ \forall \mathcal{B}^*\ \exists \mathcal{D}$ s.t.:

$$|\Pr[\,1 \leftarrow \mathcal{D}(RWT)\,] - \Pr[\,1 \leftarrow \mathcal{D}(IWT^*)\,]| > \omega\ , \tag{7.8}$$

where ω is non-negligible term.

Let $\mathcal{A}_2^{\mathcal{B}}$ be an adversary in the FS-IND-RoR experiment which uses $\mathcal{B}$ and $\mathcal{D}$ as subroutine. The game $\mathcal{A}_2^{\mathcal{B}}$ vs $\mathcal{CH}$ proceeds as follows:

- At the beginning of the experiment, $\mathcal{CH}$ chooses a bit b at random and outside $\mathcal{A}_2^{\mathcal{B}}$'s view.
- $\mathcal{A}_2^{\mathcal{B}}$ uses $\mathcal{B}$ as subroutine and answers $\mathcal{B}$'s queries as follows: When $\mathcal{B}$ asks for Initialize user, Initialize user instance, Send or Corrupt queries, $\mathcal{A}_2^{\mathcal{B}}$ simply forwards them to her $\mathcal{CH}$ and its response (if any) is forwarded back to $\mathcal{B}$.
- $\mathcal{A}_2^{\mathcal{B}}$ uses her Test query to answer $\mathcal{B}$'s Application queries. When $\mathcal{B}$ asks for an application of the efficiently computable function f on sk_U^i and a global random string R, $\mathcal{A}_2^{\mathcal{B}}$ asks $\mathsf{Test}(U, i)$ to her $\mathcal{CH}$, obtains sk_U^i, computes $f(sk_U^i, R)$ and sends the result to $\mathcal{B}$.

Claim 7.11. *The transcript produced by $\mathcal{A}_2^{\mathcal{B}}$ is either RWT or IWT^*.*

Proof. $\mathcal{B}$'s actions produce a transcript t. Consider the following scenario: $\mathcal{B}$ asks an Application query and $\mathcal{A}_2^{\mathcal{B}}$ answers it by asking a Test query to his own challenger. Without loss of generality, let us consider fresh instances, i.e. those where we give credit to the adversary if he answers with $b' = b$: If $b = 1$, $\mathcal{A}_2^{\mathcal{B}}$'s Test queries are answered with real session keys, else if $b = 0$ $\mathcal{A}_2^{\mathcal{B}}$ gets a random string taken from the session key space. Therefore, $\mathcal{A}_2^{\mathcal{B}}$'s answer to $\mathcal{B}$'s Application queries is either a function of the real session key or a function of a random string. Looking at the definition of the real and ideal-world transcripts, we conclude that whenever $b = 1$ the transcript generated is real-world while if $b = 0$ the transcript is ideal world. The reason is that in the real-world, the user instances compute their sk's according to the description of the protocol

and only such *computed sk*'s are placed transcript. However, in the ideal-world, the session keys placed in the transcript are i) random strings provided that freshness condition is satisfied or ii) no restriction about sk provided that freshness is not satisfied, i.e. the simulator is given the freedom to specify the session key as he wishes. □

Let $\mathcal{D}$ be the PPT distinguisher whose existence is guaranteed by the negation of FS-SIM-BMP security.[6] Next, $\mathcal{A}_2^{\mathcal{B}}$ invokes $\mathcal{D}(t)$ and simply forwards $\mathcal{D}$'s output to her own $\mathcal{CH}$. By construction, $\mathcal{A}_2^{\mathcal{B}}$ wins whenever $\mathcal{D}$ is able to distinguish real-world from ideal-world transcripts. Therefore:

$$\Pr\left[\,\mathcal{A}_2^{\mathcal{B}}\ \ wins\,\right] \;=\; \Pr\left[\,b=1\,\right]\cdot\Pr\left[\,1\leftarrow\mathcal{D}(RWT)\,\right] \;+\; \Pr\left[\,b=0\,\right]\cdot\Pr\left[\,0\leftarrow\mathcal{D}(IWT^{*})\,\right]\;,$$

which using Equation 7.8 gives:

$$\Pr\left[\,\mathcal{A}_2^{\mathcal{B}}\ \ wins\,\right] > \frac{1}{2}+\omega. \tag{7.9}$$

We build $\mathcal{A}$ as the sequential composition of $\mathcal{A}_1$ and $\mathcal{A}_2^{\mathcal{B}}$. It follows that:

$$\Pr\left[\,\mathcal{A}^{\mathcal{B}}\ \ wins\,\right] = \Pr\left[\,\mathcal{A}_1\ \ wins\,\right] + \Pr\left[\,\mathcal{A}_2^{\mathcal{B}}\ \ wins\,\right] \;-\Pr\left[\,\mathcal{A}_1\ \ wins\,\right]\cdot\Pr\left[\,\mathcal{A}_2^{\mathcal{B}}\ \ wins\,\right],$$

which from Equations 7.7 and 7.9 yields:

$$\Pr\left[\,\mathcal{A}^{\mathcal{B}}\ \ wins\,\right] > \frac{n_A}{2\cdot|D|}+\frac{1}{2}+\omega$$

$$Adv_{P,D}^{FS-RoR}(\mathcal{A}^{\mathcal{B}}) > \frac{n_A}{|D|}+\omega, \tag{7.10}$$

where ω is a non-negligible function. ■

Unfortunately, Theorem 7.10 is not enough to demonstrate that FS-IND-RoR security implies FS-SIM-BMP security. The reason is that the total number of instances initialized by our construction of $\mathcal{A}$ is n_A+n_B, where n_A is the number of explicit password guesses of subroutine $\mathcal{A}_1$ and n_B is the number of instances initialized while subroutine $\mathcal{A}_2^{\mathcal{B}}$ is simulating the real world ring master to $\mathcal{B}$. Therefore, proving by contradiction that FS-IND-RoR $\Rightarrow$ FS-SIM-BMP would require $Adv_{P,D}^{FS-RoR}(\mathcal{A}) > (n_A+n_B)/|D|+\omega$.

We recall from Section 7.3 that there are two ways to take account of online dictionary attacks in SIM-based security models for PAKEs:

1. Include a *test instance password* query in IW and require computational indistinguishability of RWT and IWT^{*}.
2. Do not include a *test instance password* in IW but allow a non-negligible bound on the distinguishability of RWT and IWT^{*}.

[6] Without loss of generality, we can assume $\mathcal{D}$ is more likely to output 1 on a real world than on an ideal world transcript; otherwise, flip the output bit of $\mathcal{D}$.

The SIM-based model Boyko, MacKenzie and Patel [BMP00] follows the first style. We modify it so it follows the second style. We call the modified model FS-SIM-BMP'. The only changes are the following:

1. Remove the *test instance password* query from IW in FS-SIM-BMP.
2. Relax the requirement of indistinguishability between real and ideal world.

FS-SIM-BMP' security. Protocol P is FS-SIM-BMP' secure if it satisfies completeness and additionally for all *Real World* adversaries $\mathcal{B}$, there exits an *Ideal World* adversary $\mathcal{B}^*$ such that for all distinguishers $\mathcal{D}$:

$$|\Pr[\,1 \leftarrow \mathcal{D}(RWT)\,] - \Pr[\,1 \leftarrow \mathcal{D}(IWT^*)\,]| \leq \frac{n}{|D|} + negl(\kappa), \tag{7.11}$$

where n is an upper bound on the number of sessions initialized by $\mathcal{B}$.

Next, we show that FS-IND-RoR security implies FS-SIM-BMP' security.

Theorem 7.12. *(FS-IND-RoR Security ⇒ FS-SIM-BMP' Security). If protocol P is secure in the IND-RoR model with forward secrecy, then P is also secure in the SIM-BMP' model with forward secrecy.*

Proof. This is a proof by contradiction and the strategy is similar to the one employed in Theorem 7.10.

We assume that FS-SIM-BMP' security does not hold. Then $\exists\mathcal{B}\ \forall\mathcal{B}^*\ \exists\mathcal{D}$ s.t.:

$$|\Pr[\,1 \leftarrow \mathcal{D}(RWT)\,] - \Pr[\,1 \leftarrow \mathcal{D}(IWT^*)\,]| > \frac{n}{|D|} + \omega, \tag{7.12}$$

where n is an upper bound on the number of sessions initialized and ω is a non-negligible function.

Then, we build an adversary $\mathcal{A}^{\mathcal{B}}$ using $\mathcal{B}$ and $\mathcal{D}$ as subroutines such that $\mathcal{A}$ breaks FS-IND-RoR security. We construct $\mathcal{A}^{\mathcal{B}}$ from $\mathcal{B}$ and $\mathcal{D}$ in exactly the same way as we built $\mathcal{A}_2^{\mathcal{B}}$ from $\mathcal{B}$ and $\mathcal{D}$ in the proof of Theorem 7.10.

Using the same analysis as in the proof of Theorem 7.10, we get:

$$\Pr[\,\mathcal{A}^{\mathcal{B}}\ \ wins\,] \;=\; \Pr[\,b=1\,]\ \cdot\ \Pr[\,1 \leftarrow \mathcal{D}(RWT)\,]\ +\ \Pr[\,b=0\,]\ \cdot\ \Pr[\,0 \leftarrow \mathcal{D}(IWT^*)\,]\ ,$$

which using Equation 7.12 gives:

$$\Pr[\,\mathcal{A}^{\mathcal{B}}\ \ wins\,] > \frac{1}{2} + \frac{n}{2\cdot|D|} + \omega\ ,$$

And finally from Equation 7.1:

$$Adv_{P,D}^{FS-RoR}(\mathcal{A}^{\mathcal{B}}) > \frac{n}{|D|} + \omega,$$

but ω is not negligible, a contradiction. ■

Now, we investigate the reverse, i.e. whether FS-SIM-BMP' security implies FS-IND-RoR security. We obtain the following result:

Theorem 7.13. *(FS-SIM-BMP' Security ⇒ FS-IND-RoR Security). If protocol P is SIM-BMP' secure with forward secrecy, then for all PPT $\mathcal{A}$, $Adv_{P,D}^{FS-RoR}(\mathcal{A}) \leq 2 \cdot n/|D| + negl(\kappa)$.*

Proof. We follow the same argument as in the proof of Theorem 7.6 up to Equation 7.5, which we simply update according to the FS-SIM-BMP' security definition given in Equation 7.11. Hence:

$$|\Pr[\mathcal{A} \text{ wins } \mid t = RWT] - \Pr[\mathcal{A} \text{ wins } \mid t = IWT^*]| \leq \frac{n}{|D|} + negl(\kappa). \tag{7.13}$$

It is easy to see that $\Pr[\mathcal{A} \text{ wins } \mid t = IWT^*] = 1/2$ since $\mathcal{A}$ cannot gain any information about the hidden bit b. However, $\Pr[\mathcal{A} \text{ wins } \mid t = RWT] = 1/2 + 1/2 \cdot Adv_{P,D}^{FS-RoR}(\mathcal{A})$ as result of $\mathcal{A}$ running on a perfectly simulated FS-IND-RoR experiment. Following Equation 7.13 we obtain:

$$Adv_{P,D}^{FS-RoR}(\mathcal{A}) \leq \frac{2 \cdot n}{|D|} + negl(\kappa)$$

□

The guarantee that $\forall \mathcal{A}$, $Adv_{P,D}^{FS-RoR}(\mathcal{A}) \leq 2 \cdot n/|D| + negl(\kappa)$ means that protocol P satisfies the definition of FS-IND-RoR security (Definition 7.1) with a degradation factor $c = 2$. A similar constant factor appears in the reduction used in [AFP05] to prove that IND-RoR security implies IND-FtG security.

Using the results of Theorem 7.6 and Theorem 7.12, as well as the previously known relation IND-RoR ⇒ IND-FtG [AFP05], we obtain the following corollaries:

Corollary 7.14. *SIM-BMP Security ⇒ SIM-BMP' Security*

Corollary 7.15. *SIM-BMP Security ⇒ IND-FtG Security*

We make the observation that Corollary 7.15 holds when the Corrupt query is removed from the SIM-BMP and IND-FtG models. The reason is that in [AFP05], the relation IND-RoR ⇒ IND-FtG is proven without considering any notion of forward secrecy, while the IND-RoR model that we consider has forward secrecy incorporated. Nevertheless, it seems reasonable to assume that IND-RoR with forward secrecy implies IND-FtG with forward secrecy.

The question of whether SIM-BMP' Security ⇒ SIM-BMP Security remains open. Note that SIM-BMP' ⇒ SIM-BMP would imply that the three security notions IND-RoR, SIM-BPM and SIM-BMP' are equivalent. A similar reasoning can be applied when considering forward secrecy in the aforementioned security models.

7.5 Conclusion and Future Work

Although PAKE is a widely studied primitive and found in real-world security protocols, a clear relation between its major security notions (IND and SIM) was missing in the literature. In this work, we aimed at filling this gap. We have summarized the relations obtained in this work in Figure 7.2.

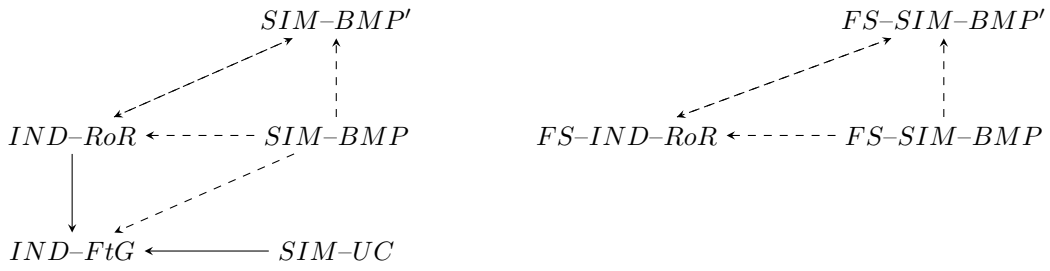

Figure 7.2: Relations between PAKE security definitions. On the left side, we consider security models without forward secrecy , then the results follow from previously known relations [CHK+05, AFP05] and our results [BIOv17]. The right side of the figure, we show the relations between the SIM-BMP and IND-RoR model when forward secrecy is incorporated into the underlying models [BIOS19].

During our work on this topic, we identified some delicate definitional issues veiled under the many subtleties of the security notions for PAKE. We recall what we consider the most relevant:

- In IND-based models [BPR00, AFP05] the possible states in which a user instance could be are continue, reject, accept and terminate. Roughly speaking, an instance is in *accept* state whenever it has computed the sk but is still waiting to receive another message – typically a confirmation code – to fulfill the protocol specification, while an instance in *terminate* state means that it has computed the sk, has finished the protocol execution and is not sending nor receiving any other message. Particularly in the IND-FtG model, a Reveal query can be asked to instances in *accept* state while a Test query can only be directed to instances in *terminate* state. The aforementioned distinction between accept and terminate states does not exist in SIM-based models due to how the ideal world is modeled. The idea is the following:

  - In the SIM-BMP model, the Application query models the leakage of session keys in higher level protocols. The implicit requirement is that the corresponding user instance has *terminated* his protocol execution, which is modeled in the Ideal-World via connection assignments.

  - In the IND-FtG model, the Reveal query models i) the leakage of session keys in higher level protocols and ii) leakage of session keys before the protocol is finished, i.e. *accept* state.

The aforementioned peculiarity is specially relevant for Corollary 7.14. In order for the implication SIM-BMP $\Rightarrow$ IND-FtG to hold, we require the Reveal (U, i) query in the IND-FtG model to be legitimate only if the instance Π_U^i is in *terminate* state. It might be a minor difference between IND-based and SIM-based models, yet we consider it is worth mentioning, specially because it is generally assumed that SIM-based security definitions are stronger than their corresponding IND-based ones. However, as we have just explained, there are technicalities that need to be addressed when formally stating the relation between the security models.

- A more remarkable difference between IND and SIM models for PAKEs is how online dictionary attacks are captured in the security model. In IND-based models, the advantage of an adversary is formulated according to parameter n, which represents the number of active instances created by the adversary in question. Note that such a definition does not specify or take into account the fact that the adversary's strategy is randomized, and thus n may be a randomized function as well. For instance, an adversary could create a large number of instances with negligible probability making the bound on its advantage grow. The difference between models with an explicit formulation of a non-negligible bound on the advantage and models without such an explicit formulation seems to be related to the difficulty in proving IND-RoR $\Rightarrow$ SIM-BMP. Another related issue is about the password distribution and the correlation of passwords between users. We leave the quest for a more precise definition that would take into account the above-mentioned remarks for future work.

- The SIM-BMP model offers a more meaningful security definition by better capturing the capabilities of an attacker against a PAKE protocol, for instance online dictionary attacks. Additionally, the SIM-BMP model does not place any artificial constraints on the passwords distribution, whereas the IND-RoR requires the passwords to be uniformly distributed and independent. The last requirement might be difficult to satisfy in real scenarios. In particular, it is known that certain passwords are more likely to be selected than others and that users tend to choose similar passwords when connecting to different services.

Finally, we demonstrated that the results obtained in [BIOv17] are still satisfied when the corresponding security models incorporate forward secrecy as required security property.

CHAPTER 8

Concluding Remarks and Future Directions

8.1 Summary

The overall goal of this work was to gain more clarity on the security provided by PAKE protocols, particularly EKE-based ones. We expect that this work leads to the standardization and more efficient implementation of the considered protocols. Additionally, by establishing exact relations between SIM-based and IND-based security notions for PAKEs, we aim to provide unity across the underlying security models. The later permits a better understanding of the security guarantees provided by protocols proven secure in such models.

We summarize our contributions as follows:

1. We formalized and incorporated the notion of weak forward secrecy to the Find-then-Guess model (see Chapter 4). Then, we demonstrated that the original SPAKE2 protocol is secure in the FtG model with weak forward secrecy in the random oracle model assuming the CDH problem is intractable. Furthermore, the adoption of key-confirmation codes results in a protocol that provably satisfies the stronger notion of perfect forward secrecy.

2. We provide an instantiation for the PAK protocol which allows us to construct a tight security reduction from the Gap-CDH problem. This technique can be applied to other EKE-based protocols – for instance PPK, SPAKE2 and PFS-SPAKE2 – to obtain tight security reductions.

3. We established the relations between the IND-based and SIM-based security notions for PAKEs. In particular, we first demonstrated that a protocol secure in the SIM-BMP

model is also secure in the IND-RoR model. This result implies that protocols, which were originally proven secure in the SIM-BMP model, allow for secure composition with arbitrary symmetric-key encryption schemes [SL18]. We then demonstrated that IND-RoR security implies security in a slightly modified version of the SIM-BMP model.

8.2 Future Directions

1. Our work from Chapter 5 can be extended in the following way:

   - The SPAKE2 protocol is proven secure in the FtG model, however, it would be meaningful to provide a security proof in a model that guarantees its composability with symmetric-key encryption schemes, for instance the IND-RoR model.
   - It remains an open question whether one-round PAKE protocols, which satisfy only implicit authentication, may satisfy the stronger notion of perfect forward secrecy. Considering *algebraic adversaries* for PAKEs is an interesting research direction that might help to answer the previous question in the affirmative.

2. From Chapter 7, it remains as open question whether security in the RoR model implies security in the original SIM-BMP model. The most important difference between these notions of security is the approach to model online dictionary attacks. We leave as an open problem the formulation of an alternative definition for online dictionary attacks that takes into consideration the fact that the adversarial's strategy is randomized.

3. For future work, it would be reasonable to transform the proposed protocols into *strong augmented* PAKEs in the style of [JKX18] and benefit from stronger security guarantees. This would require to first construct the augmented variant of underlying protocols with security proofs in the UC model.

4. Finally, we noticed that open source, portable and efficient implementations of provably secure PAKEs are mostly missing – probably due to previous patent issues. We believe that providing such implementation in the form of cryptographic libraries could boost the popularity and deployment of PAKEs in real-world applications.

Bibliography

[ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE Password Authenticated Key Exchange Protocol. In *2015 IEEE Symposium on Security and Privacy, SP 2015*, pages 571–587. IEEE Computer Society, 2015.

[ACP05] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange. In Serge Vaudenay, editor, *Public Key Cryptography - PKC 2005*, pages 47–64, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-Based Authenticated Key Exchange in the Three-Party Setting. In Serge Vaudenay, editor, *Public-Key Cryptography – PKC 2005*, volume 3386 of *LNCS*, pages 65–84. Springer, 2005.

[AP05a] Michel Abdalla and David Pointcheval. Interactive diffie-hellman assumptions with applications to password-based authentication. In Andrew S. Patrick and Moti Yung, editors, *Financial Cryptography and Data Security*, pages 341–356, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[AP05b] Michel Abdalla and David Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *LNCS*, pages 191–208. Springer, 2005.

[AS99] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Commun. ACM*, 42(12):40–46, December 1999.

[Bai07] Thomas Baignres. Provable security in cryptography. page 28, 2007.

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In Jeffrey Scott

Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, STOC 98*, pages 419–428. ACM, 1998.

[BCP04] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. New Security Results on Encrypted Key Exchange. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography*, volume 2947 of *LNCS*, pages 145–158. Springer, 2004.

[Bel99] Mihir Bellare. Practice-Oriented Provable Security. *LNCS*, 1561:1–15, 1999.

[BF18] Richard Barnes and Owen Friel. Usage of PAKE with TLS 1.3. Internet-Draft draft-barnes-tls-pake-04, Internet Engineering Task Force, July 2018. `https://datatracker.ietf.org/doc/html/draft-barnes-tls-pake-04`.

[BFWW11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway Key Exchange Protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011*, pages 51–62. ACM, 2011.

[BIO+18] Jose Becerra, Vincenzo Iovino, Dimiter Ostrev, Petra Sala, and Marjan Skrobot. Tightly-secure PAK(E). In *Cryptology and Network Security - 16th International Conference, CANS 2017, Hong Kong, China, November 30 - December 2, 2017, Revised Selected Papers*, pages 27–48. Springer, 2018.

[BIOS19] Jose Becerra, Vincenzo Iovino, Dimiter Ostrev, and Marjan Skrobot. On the relation between SIM and IND-RoR Security Models for PAKEs with Forward Secrecy. In *E-Business and Telecommunications - 14th International Joint Conference, ICETE 2017, Madrid, Spain, July 24-26, 2017, Revised Selected Paper*, pages 173–198. Springer, 2019.

[BIOv17] Jose Becerra, Vincenzo Iovino, Dimiter Ostrev, and Marjan Škrobot. On the Relation Between SIM and IND-RoR Security Models for PAKEs. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017.*, pages 151–162. SCITEPRESS, 2017.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, 2001.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *1992 IEEE Symposium on Research in Security and Privacy, SP 1992*, pages 72–84, 1992.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, pages 244–250, New York, NY, USA, 1993. ACM.

[BM97] Simon Blake-Wilson and Alfred Menezes. Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols, 5th International Workshop*, volume 1361 of *LNCS*, pages 137–158. Springer, 1997.

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 156–171. Springer, 2000.

[BN11] Colin Boyd and Juan M. Gonzalez Nieto. On forward secrecy in one-round key exchange. In *Thirteenth IMA International Conference on Cryptograpphy and Coding*, pages 451–468, Oxford, UK, December 2011. Springer Berlin / Heidelberg.

[Bon12] Joseph Bonneau. *Guessing human-chosen secrets*. PhD thesis, University of Cambridge, UK, 2012.

[BOS18] Jose Becerra, Dimiter Ostrev, and Marjan Skrobot. Forward secrecy of SPAKE2. In Joonsang Baek, Willy Susilo, and Jongkil Kim, editors, *Provable Security - 12th International Conference, ProvSec 2018, Jeju, South Korea, October 25-28, 2018, Proceedings*, volume 11192 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2018.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, 2000.

[BR93a] Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology CRYPTO 1993*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.

[BR93b] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993.

[BR95] Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: the three party case. In Frank Thomson Leighton and Allan Borodin, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, STOC '95*, pages 57–66. ACM, 1995.

[BR00] Mihir Bellare and Phillip Rogaway. The autha protocol for password-based authenticated key exchange. In *IEEE P1363*, pages 136–3, 2000.

[BR05] Mihir Bellare and Phillip Rogaway. Introduction to modern cryptography. In *UCSD CSE 207 Course Notes*, page 207, 2005.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 409–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[BRRS18] Jose Becerra, Peter B. Rønne, Peter Y. A. Ryan, and Petra Sala. Honeypakes. In Vashek Matyás, Petr Svenda, Frank Stajano, Bruce Christianson, and Jonathan Anderson, editors, *Security Protocols XXVI - 26th International Workshop, Cambridge, UK, March 19-21, 2018, Revised Selected Papers*, volume 11286 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2018.

[BŠŠ17] Jose Becerra, Petra Šala, and Marjan Škrobot. An Offline Dictionary Attack against zkPAKE Protocol. Cryptology ePrint Archive, Report 2017/961, 2017. https://eprint.iacr.org/2017/961.

[Cam17] Dell Cameron. Over 560 million passwords discovered in anonymous online database. `https://bit.ly/2vgJqli`, 2017.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 136–145. IEEE Computer Society, 2001.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 98–115, Berlin, Heidelberg, 1999. Springer-Verlag.

[CH14] Dylan Clarke and Feng Hao. Cryptanalysis of the Dragonfly Key Exchange Protocol. *IET Information Security*, 8(6):283–289, 2014.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally Composable Password-Based Key Exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, 2005.

[CHVV03] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 583–599, 2003.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, 2001.

[CK02] Ran Canetti and Hugo Krawczyk. Universally Composable Notions of Key Exchange and Secure Channels. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, 2002.

[CKMS16] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness ii: Practical issues in cryptography. Cryptology ePrint Archive, Report 2016/360, 2016. `https://eprint.iacr.org/2016/360`.

[CKW11] Véronique Cortier, Steve Kremer, and Bogdan Warinschi. A survey of symbolic methods in computational analysis of cryptographic systems. *J. Autom. Reason.*, 46(3-4):225–259, April 2011.

[CR85] Benny Chor and Ronald L. Rivest. A knapsack type public key cryptosystem based on arithmetic in finite fields (preliminary draft). In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 54–65, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[CR88] Benny Chor and Ronald L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE TRANS. INFORM. THEORY*, 34(5):901–909, 1988.

[Cre11] Cas Cremers. Examining Indistinguishability-based Security Models for Key Exchange Protocols: the case of CK, CK-HMQV, and eCK. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011*, pages 80–91. ACM, 2011.

[DAL+17] Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy RV, and Michael Snook. Provably Secure Password Authenticated Key Exchange Based on RLWE for the Post-Quantum World. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017*, volume 10159 of *LNCS*, pages 183–204. Springer, 2017.

[DAT12] Italo Dacosta, Mustaque Ahamad, and Patrick Traynor. Trust No One Else: Detecting MITM Attacks against SSL/TLS without Third-Parties. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security - ESORICS 2012*, volume 7459 of *LNCS*, pages 199–216. Springer, 2012.

[DVOW92] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, Jun 1992.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. Inf. Theor.*, 29(2):198–208, 1983.

[Ecr12] II Ecrypt. ECRYPT II Yearly Report on Algorithms and Keysizes. *European Network of Excellence in Cryptology II, Tech. Rep*, 2012.

[EKSS09] John Engler, Chris Karlof, Elaine Shi, and Dawn Song. Is it too late for PAKE? In *Web 2.0 Security and Privacy Workshop 2009 (W2SP 2009)*, May 2009.

[GKR10] Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Okamoto-tanaka revisited: Fully authenticated diffie-hellman with minimal overhead. In Jianying Zhou and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 309–328, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[GL01a] Oded Goldreich and Yehuda Lindell. Session-Key Generation Using Human Passwords Only. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *LNCS*, pages 408–432. Springer, 2001.

[GL01b] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 408–432, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[GM84a] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GM84b] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2)/ 270-299, 1984.

[GMR06] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, pages 142–159, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[Gol06] Oded Goldreich. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA, 2006.

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, September 2008.

[Har08] Dan Harkins. Simultaneous Authentication of Equals: A Secure, Password-Based Key Exchange for Mesh Networks. In *Proceedings of the 2008 Second International Conference on Sensor Technologies and Applications*, SENSORCOMM '08, pages 839–844. IEEE Computer Society, 2008.

[HR10] Feng Hao and Peter Ryan. J-PAKE: Authenticated Key Exchange without PKI. *Transactions on Computational Science*, 11:192–206, 2010.

[Ian12] Paul Ian. Linkedin confirms account passwords hacked. `https://bit.ly/2v2qjMh`, 2012.

[IEE02] Standard Specifications for Password-Based Public Key Cryptographic Techniques. Standard, IEEE Standards Association, Piscataway, NJ, USA, 2002.

[ISO09] ISO/IEC 11770-4:2006/cor 1:2009, Information Technology – Security techniques – Key Management – Part 4: Mechanisms Based on Weak Secrets. Standard, International Organization for Standardization, Genève, Switzerland, 2009.

[Jab96] David P Jablon. Strong Password-Only Authenticated Key Exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, 1996.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, 2012.

[JKSS18] Stanislaw Jarecki, Hugo Krawczyk, Maliheh Shirvanian, and Nitesh Saxena. Two-factor authentication with end-to-end password security. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 431–461, Cham, 2018. Springer International Publishing.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-Computation Attacks. In Orr Dunkelman, editor, *Advances in Cryptology – EUROCRYPT 2018*, LNCS. Springer, 2018.

[JN03] Antoine Joux and Kim Nguyen. Separating decision Diffie–Hellman from computational Diffie–Hellman in cryptographic groups. *Journal of cryptology*, 16(4):239–247, 2003.

[JR13] Ari Juels and Ronald L. Rivest. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 145–160, New York, NY, USA, 2013. ACM.

[Kah96] D. Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996.

[KJP06] Sébastien Kunz-Jacques and David Pointcheval. About the security of mti/c0 and mqv. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks*, pages 156–172, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[KKS13] Czeslaw Koscielny, Miroslaw Kurkowski, and Marian Srebrny. *Modern Cryptography Primer - Theoretical Foundations and Practical Applications*. Springer, 2013.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, 2001.

[KOY02] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Forward Secrecy in Password-Only Key Exchange Protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks – SCN 2002*, volume 2576 of *LNCS*, pages 29–44. Springer, 2002.

[Kra03] Hugo Krawczyk. Sigma: The 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike protocols. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 400–425, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Kra05] Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 546–566, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *Proceedings of the 8th Conference on Theory of Cryptography*, TCC'11, pages 293–310, Berlin, Heidelberg, 2011. Springer-Verlag.

[Len06] Arjen K Lenstra. Key lengths. Technical report, Wiley, 2006.

[Lin17] Yehuda Lindell. *How to Simulate It – A Tutorial on the Simulation Proof Technique*, pages 277–346. Springer International Publishing, Cham, 2017.

[LK18] Watson Ladd and Benjamin Kaduk. Spake2, a pake. Internet-Draft draft-irtf-cfrg-spake2-05, IETF Secretariat, February 2018. `http://www.ietf.org/internet-drafts/draft-irtf-cfrg-spake2-05.txt`.

[LLM07] Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *LNCS*. Springer, 2007.

[LMQ+03] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2), March 2003.

[Low95] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, November 1995.

[LS15] Jean Lancrenon and Marjan Skrobot. On the Provable Security of the Dragonfly Protocol. In Javier Lopez and Chris J. Mitchell, editors, *Information Security – ISC 2015*, volume 9290 of *LNCS*, pages 244–261. Springer, 2015.

[LST16] Jean Lancrenon, Marjan Skrobot, and Qiang Tang. Two More Efficient Variants of the J-PAKE Protocol. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security – ACNS 2016*, volume 9696 of *LNCS*, pages 58–76. Springer, 2016.

[Mac01a] Philip MacKenzie. On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. Cryptology ePrint Archive, Report 2001/057, 2001. http://eprint.iacr.org/2001/057.

[Mac01b] Philip D. MacKenzie. More Efficient Password-Authenticated Key Exchange. In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 361–377. Springer, 2001.

[Mac02a] Philip MacKenzie. The PAK Suite: Protocols for Password-Authenticated Key Exchange. DIMACS Technical Report 2002-46, 2002.

[Mac02b] Phillip MacKenzie. Methods and Apparatus for Providing Efficient Password-Authenticated Key Exchange, 2002. Publication number US20020194478 A1.

[MJ16] Nadia El Mrabet and Marc Joye. *Guide to Pairing-Based Cryptography*. Chapman & Hall/CRC, 2016.

[moz] Mozilla telemetry. `https://telemetry.mozilla.org/`.

[MPS00a] Philip MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-authenticated key exchange based on rsa. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, pages 599–613, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[MPS00b] Philip D. MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-Authenticated Key Exchange Based on RSA. In *Advances in Cryptology - ASIACRYPT 2000*, LNCS, pages 599–613. Springer, 2000.

[MSHH18] Nathaniel McCallum, Simo Sorce, Robbie Harwood, and Greg Hudson. Spake pre-authentication. Internet-Draft draft-ietf-kitten-krb-spake-preauth-05, IETF Secretariat, February 2018. `http://www.ietf.org/internet-drafts/draft-ietf-kitten-krb-spake-preauth-05.txt`.

[MW18] Daniele Micciancio and Michael Walter. On the bit security of cryptographic primitives. In *EUROCRYPT (1)*, volume 10820 of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2018.

[NCPW13] Junghyun Nam, Kim-Kwang Raymond Choo, Juryon Paik, and Dongho Won. An Offline Dictionary Attack against a Three-Party Key Exchange Protocol. *IACR Cryptology ePrint Archive*, 2013:666, 2013. `http://eprint.iacr.org/2013/666`.

[NV08] Minh-Huyen Nguyen and Salil P. Vadhan. Simpler Session-Key Generation from Short Random Passwords. *J. Cryptology*, 21(1):52–96, 2008.

[OP01] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In Kwangjo Kim, editor, *Public-Key Cryptography - PKC 2001*, volume 1992 of *LNCS*, pages 104–118. Springer, 2001.

[OWT09] Yutaka Oiwa, Hajime Watanabe, and Hiromitsu Takagi. Pake-based mutual HTTP authentication for preventing phishing attacks. *CoRR*, abs/0911.5230, 2009.

[Pat97] Sarvar Patel. Number theoretic attacks on secure password schemes. In *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*, pages 236–247, 1997.

[Per] Colin Percival. Stronger key derivation via sequential memory-hard functions.

[PG14] Nicole Perlroth and David Gelles. Russian hackers amass over a billion internet passwords. `https://nyti.ms/2Apak05`, 2014.

[Poi05] David Pointcheval. *Provable Security for Public Key Schemes*, pages 133–190. Birkhäuser Basel, Basel, 2005.

[Poi12] David Pointcheval. Password-Based Authenticated Key Exchange. In Marc Fischlin, Johannes A. Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012*, volume 7293 of *LNCS*, pages 390–397. Springer, 2012.

[PW17] David Pointcheval and Guilin Wang. Vtbpeke: Verifier-based two-basis password exponential key exchange. In *Proceedings of the 2017 ACM on Asia Conference on*

Computer and Communications Security, ASIA CCS '17, pages 301–312, New York, NY, USA, 2017. ACM.

[Res18] Eric Rescorla. The transport layer security (tls) protocol version 1.3. Internet-Draft draft-ietf-tls-tls13-28, IETF Secretariat, March 2018. `http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-28.txt`.

[Riv] Ronald L. Rivest. The impact of technology on cryptography. In *Proceedings 1978 International Conference on Communications*. IEEE.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. *IACR Cryptology ePrint Archive*, 2017:190, 2017.

[Sha49] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal, Vol 28, pp. 656715*, Oktober 1949.

[Sho99] Victor Shoup. On Formal Models for Secure Key Exchange. Cryptology ePrint Archive, Report 1999/012, 1999. `http://eprint.iacr.org/1999/012`.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[Sil09] Joseph H Silverman. *The Arithmetic of Elliptic Curves*, volume 106. Springer Science & Business Media, 2009.

[SJKS17] Maliheh Shirvanian, Stanislaw Jarecki, Hugo Krawczyk, and Nitesh Saxena. SPHINX: A password store that perfectly hides passwords from itself. In *ICDCS*, pages 1094–1104. IEEE Computer Society, 2017.

[SL18] Marjan Skrobot and Jean Lancrenon. On composability of game-based password authenticated key exchange. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 443–457, 2018.

[Szy06] Michael Szydlo. A note on chosen-basis decisional diffie-hellman assumptions. In Giovanni Di Crescenzo and Avi Rubin, editors, *Financial Cryptography and Data Security*, pages 166–170, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[TWMP07] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin. Using the secure remote password (srp) protocol for tls authentication. RFC 5054, RFC Editor, November 2007.

[Vac13] John R. Vacca. *Computer and Information Security Handbook, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2013.

[Vau98] Serge Vaudenay. Cryptanalysis of the chor-rivest cryptosystem. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 243–256, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[WRBW16] Rick Wash, Emilee Rader, Ruthie Berman, and Zac Wellmer. Understanding password choices: How frequently entered passwords are re-used across websites. In *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, pages 175–188, Denver, CO, 2016. USENIX Association.

[Wu98] Thomas D. Wu. The Secure Remote Password Protocol. In *Proceedings of the Network and Distributed System Security Symposium, 1998*. The Internet Society, 1998.

[WW16] Ding Wang and Ping Wang. On the Implications of Zipf's Law in Passwords. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *Computer Security - ESORICS 2016*, volume 9878 of *LNCS*, pages 111–131. Springer, 2016.

[YBAG04] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, September 2004.