Design and Analysis of Cryptographic Protocols

YONG LI
(PLACE OF BIRTH: HENAN, CHINA)

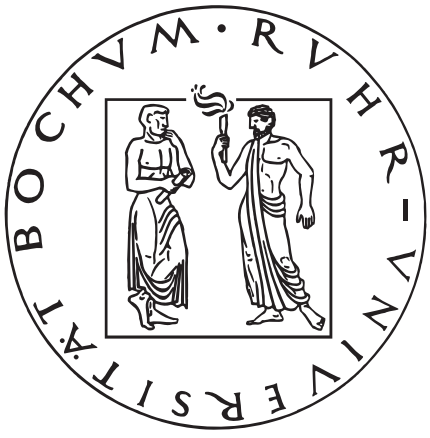
Dissertation

for the Degree of `Doktor-Ingenieur` (`Dr. Ing.`)

Department for

Electrical Engineering and Information Technology

Ruhr University Bochum (Germany)



Chair for Network and Data Security

Bochum, June 2015

Author Contact Information:
yong.li@rub.de

Thesis Advisor: Prof. Jörg Schwenk
Prof. Dennis Hofheinz
PhD Defense: 2015/06/24

Abstract

Cryptographic protocols are a critical element of the infrastructure for secure communication. They can be used to provide security guarantees for the exchanged data when multiple parties are communicating in an insecure or untrusted environment. The core of cryptographic protocols such as SSL, TLS, SSH, or IPSec is the key exchange protocol, which allows two parties to generate a shared secret key used for establishing a secure channel. Authenticated key exchange (AKE) and authenticated and confidential channel establishment (ACCE) protocols are the most important building blocks in secure communication. Intuitively, they allow a party "Alice" to authenticate a communication partner "Bob" and securely establish a common session key (used for establishing a secure communication channel) with Bob, and vice versa. The communication between Alice and Bob is then protected with this secret session key.

This thesis is concerned with the approach to design and analysis of cryptographic protocols based around the most widely used security models. In particular, we show how to systematically construct and analyze security protocols in realistic models. This work splits into two parts.

In the first part of this thesis we give an overview of the most widely used security models for AKE/ACCE protocols. Then, we extend existing security models to capture relevant attacks that were originally outside the scope of these models. Finally, we discuss the differences between the proposed security models and show the relationship between the existing models and our models.

In the second part of this thesis we first analyze the security of all three TLS-PSK ciphersuites in our extended ACCE model. It is the first detailed security analysis for the symmetric-key based TLS-PSK protocols. Then, we construct the first tightly-secure AKE protocol in an enhanced Bellare-Rogaway security model under standard assumptions. In contrast to other AKE protocols, its security does not degrade with an increasing number of participating parties and protocol sessions. Moreover, we present new efficient compilers that generically turn passively secure key exchange protocols into efficient AKE protocols where the security properties hold in the realistic setting of communication channels controlled by an active adversary. Finally, we introduce a new theoretical attack for AKE protocols, named *no-match attack*, and show that proving security under the matching conversations (MC) as session IDs (MC-based sID) is a delicate issue. In particular, we provide several examples of AKE protocols that claim to be secure under a security definition based on MC-based session identifier but where the security proof is actually flawed. Additionally, we give several generic ways to thwart our *no-match* attacks.

Kurzfassung

Informationssicherheit spielt eine wichtige und entscheidende Rolle in der heutigen digitalen Gesellschaft. Integrität und Authentizität von Daten muss sichergestellt werden können und Informationen müssen vertraulich gesendet werden können. Kryptographische Protokolle zur authentischen Schlüsselvereinbarung (AKE-Protokolle) bilden einen wichtigen Baustein, um eine sichere Kommunikation zu gewährleisten. Diese Protokolle werden von mehreren Parteien in einer unsicheren oder nicht vertrauenswürdigen Kommunikationsumgebung, wie dem Internet, ausgeführt um anschließend Daten authentisch, integritätsgeschützt und vertraulich übertragen zu können. Etwas genauer erlauben sie einer Partei, etwa Alice (A), einen Kommunikationspartner, beispielsweise Bob (B), zu authentifizieren und einen gemeinsamen, "sicheren" Sitzungsschlüssel zu generieren. Die Kommunikation wird anschließend mit diesem Sitzungsschlüssel geschützt.

Das Design und die Analyse der sicheren Kommunikationsprotokolle haben sich als nicht-triviale Aufgabe erwiesen. Der Beweis der Sicherheit einfacher kryptographischer Protokolle ist üblicherweise sehr komplex. Viele Protokolle wurden ohne theoretische Rechtfertigung, d.h. ohne formalen Sicherheitsbeweis, entwickelt. Dies umfasst etwa die Protokolle TLS/SSL, IPSec oder Kerberos. Aus diesem Grund gibt es in regelmäßigen Abständen Angriffe auf diese Protokolle. Daher ist die Analyse bstehender Protokolle, sowie der Entwurf theoretisch fundierter kryptographischer Protokolle von großer Wichtigkeit, wenn man solche Angriffe vermeiden möchte.

Diese Arbeit beschäftigt sich mit dem Design und Analyse kryptographischer Protokolle in den aktuell gängisten Sicherheitsmodellen. Wir zeigen wie man die Kommunikationsprotokolle systematisch konstruieren und analysieren kann, insbesondere für authentifizierte Schlüsselaustauschprotokollen in realitätsnahen Modellen. Protokoll entwickeln und analysieren kann. Diese Arbeit besteht aus zwei Teilen.

Im ersten Teil dieser Dissertation erweitern wir existierende Sicherheitsmodelle, um zusätzlich praktisch relevante Angriffe zu simulieren, die durch existierende Modelle noch nicht berücksichtigt wurden. Außerdem diskutieren wir die vorgeschlagenen Sicherheitsmodelle und zeigen die Beziehungen zu vorherigen Modellen auf.

Im zweiten Teil dieser Dissertation beschreiben wir die Ergebnisse mit den folgenden Schwerpunkten:

- Wir analysieren die Sicherheit des TLS-PSK Protokolls. TLS-PSK ist ein sehr wichtiges Sicherheitsprotokoll, welches häufig für den Remotezugriff auf eine Smartcard eingesetzt wird. Um die Sicherheit des Protokolls zu beweisen haben wir das bekannte ACCE Modell erweitert.
- Wir konstruieren generisch das erste AKE-Protokoll mit einer "scharfen Reduktion" (tight reduction). Der Sicherheitsbeweis verliert nur einen konstanten Faktor.

- We erforschen zwei neue effiziente Transformationen, die sichere Schlüsselaustauschprotokolle gegen passive Angreifer in effiziente sichere AKE-Protokolle gegen aktive Angreifer überführen.
- Die Dissertation schließt mit einen neuen generischen Angriff gegen AKE-Protkolle ("No-Match Attack"). Wir zeigen, dass unser Angriff auf viele existierende AKE-Protokolle durchgeführt werden kann. Wir betonen, dass unsere Angriffe nicht das Modell verletzen, in dem die Sicherheit dieser Protokolle bewiesen wurde. Schließlich geben wir einige Lösungen an mit deren Hilfe unser Angriff verhindert werden kann.

Acknowledgements

I am grateful to many people who have helped me throughout my PhD. First, I thank my supervisor Prof. Jörg Schwenk, who gave me the opportunity of pursuing research at his chair with a family atmosphere. Prof. Schwenk always supported me, gave me the freedom for my research interests and excellent advices which improved my work. My thanks also go to my thesis reader Prof. Dennis Hofheinz for giving me invaluable advice and support.

I thank to all members of the chair of Network- and Data Security with whom I had a lot of productive, funny, and interesting discussions. I feel very privileged for having them as my colleagues. In particular, I would like to thank Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, Sven Schäge and Zheng Yang for very interesting discussions, their very helpful support. Working with you was a pleasure. At the same time, I would like to thank Christoph Bader and Tibor Jager once again for giving me detailed advice on how to improve my work. Finally, my biggest thanks goes to my family for their support during my studies in Germany.

Contents

List of Figures

List of Tables

Listings

- **Integer**, **Strings**, **Transcript** and **Vectors**. Let $\mathcal{N} = [n] = \{1, \ldots, n\} \subset \mathbb{N}$ be the set of integers between 1 and n. Unless otherwise specified, a natual number is presented in its binary expansion. We denote by 1^n the unary expansion of n, i.e., the concatenation of n 1's. Given a string x, we denote $x|_i$ be the i'-th bit of x. We denote $\overrightarrow{x}$ be a finite sequence of elements $\{x_1, \cdots x_n\}$, and we let $|\overrightarrow{x}|$ denote the number of elements in the sequence. Let '||' denote the operation concatenating two binary strings. Let T be the transcript of messages sent and received by communication parties.
- **Parameter Spaces.** Let $\mathcal{SK}$ be the long-term secret key space and $\mathcal{PK}$ be the long-term public key space. Let $\mathcal{EPK}$ be the ephemeral public key space, $\mathcal{ESK}$ be the ephemeral secret key space and $\mathcal{SSK}$ be the session key space. Let $\mathcal{M}$ be the message space and $\mathcal{C}$ be the ciphertext space. Let $\mathcal{R}$ be the random value space. Let $\mathcal{IDS}$ be the identity space of the communication parties and $\mathit{\Pi}$ be the set of system parameters.
- **Algorithms.** Let S be a set, we then denote by $s \xleftarrow{\$} S$ the operation of picking an element s of S uniformly at random. We write $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and by $z \xleftarrow{\$} \mathcal{A}(x, y, \ldots)$ we denote the operation of running $\mathcal{A}$ with inputs (x, y, ...) and letting z be the output of the algorithm $\mathcal{A}$. We write $\mathcal{A}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot),\cdots}(x, y, \ldots)$ to indicate that $\mathcal{A}$ is an algorithm with inputs $x, y, \ldots$ and black-box access to oracles $\mathcal{O}_1(\cdot), \mathcal{O}_2(\cdot), \ldots$. If $\mathcal{A}$ is a randomized algorithm, the notion $\mathcal{A}(x; r)$ means running $\mathcal{A}$ with input x and randomness r.
- **Protocols.** Let KE be key exchange protocols, and AKE be authenticated key exchange protocols. Let tAKE be AKE protocols with a tight reduction. Let ACCE be authenticated and confidential channel establishment protocols.
- **Security Models.** Let $\mathsf{BR93}$ be the Bellare-Rogaway model introduced by Bellare and Rogaway in 1993 for the security analysis of authenticated key exchange protocols. Let $\mathsf{CK01}$ be the Canetti and Krawczyk security model introduced by Canetti and Krawczyk in 2001 for (implicitly) authenticated key exchange protocols. Let $\mathsf{CK}_{\mathsf{HMQV}}$ be the extension of the Canetti and Krawczyk security model introduced by

Krawczyk in 2005 in order to capture KCI and weak PFS attacks. Let eCK be the extended Canetti-Krawczyk model introduced by LaMacchia er al. to capture exposure of ephemeral keys attacks. Let eBR be an extension of the BR93 model. Let $\mathsf{eBR}^{\mathsf{C}}$ be an extended Bellare-Rogaway security model for AKE compilers. Let $\mathsf{eBR}^{\mathsf{T}}$ be an extended Bellare-Rogaway security model for AKE protocols with a tight reduction. Let ACCE12 be a security model for authenticated and confidential channel establishment protocols introduced by Jager et al. in 2012. Let eACCE be an extension of ACCE12.

Chapter 1

Introduction

Recent years have seen the emergence of various electronic services in an attempt to facilitate everyday life, such as on-line banking, electronic trading, online shopping, social networking, e-business or e-voting, and so forth. Naturally, security is a major concern for these applications. Sensitive information transferred over the public networks need to be protected. Cryptographic protocols as the best solutions are always considered for achieving these given security requirements, for example authenticity, privacy and integrity.

Cryptographic protocols are used to provide security guarantees for the exchanged data when the communication parties communicate over insecure networks. Authenticated key exchange (AKE) and authenticated and confidential channel establishment (ACCE) protocols are central building blocks of secure communication, such as HMQV [Kra05a], TLS/SSL [DA99, Res00, BWNH+03, BU04, Uri10, PUM11], SSH [BKN06, FPS06, Har06, HSGW06, WG06, GT06] and IPSec [Pip98, MSST98, HC98, Kau05] which are always used to establish a secure connection between the communication parties. Roughly speaking, they establish a shared secret session key between the parties and subsequent communication is protected with this session key, i.e., this "secret" is used for encrypting all messages in one communication session.

The design and analysis of cryptographic protocols has proved to be a non-trivial task for every protocol designer. Bad security is worse than no security. Many designers fail to provide a formal proof of security. Flaws in the protocol can lead to disastrous effects, especially if the protocol is widely used. A number of examples exists in the academic literature of flaws that were found in protocols that had not received intensive security analysis [CBHM04, CBH05a, CBH05b, DP07, BCF+13, PS14]. For instance, the new attacks on SSL/TLS protocol are found and published in renown international conferences frequently [PRS11a, PA12, ABP+13, AP13, MSW+14]. Such errors could have been found by protocol designers if a correct proof of security was to be constructed. Therefore, the analysis of secure protocols is important for protocol development.

This thesis is focused on design and analysis of cryptographic protocols that are used to authenticate parties and to establish cryptographic session keys, which are then com-

monly used to enable a confidential transport of messages over insecure networks. We show how to systematically construct and analyze cryptographic protocols in realistic models. We will first elaborate on several important topics in the field of cryptographic protocols (especially key exchange protocols), and then formulate corresponding research questions.

- TLS is undeniably the most prominent key exchange protocol in use today. While the security of most web applications relies on the classical Diffie-Hellman and RSA-based ciphersuites of TLS, there also exist several important applications that make use of one of the less common ciphersuites [BU04, Uri10, PUM11]. One such application is (remote) authentication of resource-restricted clients like smart-cards. In these scenarios, computational efficiency and low power consumption often are one of the most important system features. Instead of using the public-key based ciphersuites of TLS, applications can apply a variant of TLS that assumes pre-shared symmetric keys between client and server. The corresponding ciphersuite family is termed TLS with pre-shared keys (TLS-PSK) and available in many TLS releases and libraries [Ope13, MJ, Bot13, Pau13, Pee13, Pet13, Tod13, Bou13].

  QUESTION 1: The research question arising from the above objectives is as follows: How secure are the commonly used symmetric TLS-PSK ciphersuites in a realistic environment?

- The wide application of AKE protocols, such as TLS protocol, make it necessary and interesting to study their security in a large-scale setting with many millions of parties. Known provably secure AKE protocols come with a reduction which has a reduction loss factor that depends on the number ℓ of parties and the number d of session per party. For example, if the reduction has to guess only one party participating in a particular session, it will lose a factor of $(d\ell)$, i.e. the number of sessions. This may become significant in large-scale applications. Informally, assume that a 1248 bit RSA modulars N refers to 80 bits of security. Thus, a solver algorithm running in time t has success probability at most $\frac{t}{2^{80}}$. Now let us assume that $d\ell = 2^{30}$. According to [BR96, Cor00], the bound on the work factor of the adversary $\mathcal{A}$ is $\frac{2^{80}}{2^{30}} = 2^{50}$. To achieve 80 bits of security, we have to choose an RSA-modulus N of size roughly 2460 bits [ECR12]. Moreover, if $d\ell > 2^{30}$, then we can not rule out the attacks, even if we choose the RSA-modulus N to be of size 2460 bits. Therefore, developing efficient AKE protocols with **tight** reduction is the most important for practical applications of cryptosystems.

  The existence of tight security reductions has been studied for many cryptographic primitives, like identity-based encryption [CW13, BKP14], digital signatures [Ber08, Sch11, KK12], and public-key encryption [BBM00, HJ12, LJYP14]. However, there is no example of an authenticated key exchange protocol that comes with tight security proof under a standard assumption, not even in the Random Oracle Model [BR93b].

QUESTION 2: As indicated above, one can raise the question: Is it possible to construct a secure authenticated key exchange protocol with a tight security reduction in an enhanced Bellare-Rogaway security model under standard assumptions?

- A generic compiler of authenticated key exchange systems has several admirable benefits and advantages [BCK98, KY07, JKSS10]. One is flexibility as one of these can resort to a rich collection of existing authentication schemes and key exchange protocols that can be combined to yield new authenticated key exchange systems. Another benefit is applicability, as a generic compiler (ideally) does not require any modifications in existing implementations of the input protocols. Instead, security can be established by simply "adding" the implementation of the compiler to the system. Also, a generic compiler can considerably simplify the security analysis, as only the input protocols have to be analyzed to meet their respective security requirements.

  QUESTION 3: How can protocol designers modularly construct secure, efficient and reliable authenticated key exchange systems?

- An essential part of the definition of security for any key exchange protocol is the notion of *partnering* [BR95, BWJM97, BCK98, Sho99a, BPR00, CK01, CK02b, KP05, Kra05a, LLM07a]. This essentially defines when two processes running the key exchange protocol can be considered to have communicated with each other and thus share important confidential information. The de-facto standard definition is that of matching conversations (MC), which essentially states that two processes are partnered if every message sent by the first is actually received by the second and vice versa.

  QUESTION 4: Is this an appropriate choice if we use the notion of matching conversation as session identifier (MC-based sID) to prove the security for (all) key exchange protocols?

CONTRIBUTIONS OF THE THESIS.

This thesis answers the previous questions such as:

- ***1: First formal security analysis of all the TLS-PSK ciphersuites.***

  To answer "How secure are the commonly used symmetric TLS-PSK ciphersuites in a realistic environment?", we give the security analysis of Transport Layer Security Pre-Shared Key ciphersuites (TLS-PSK), including `TLS_PSK`, `TLS_RSA_PSK` and `TLS_DHE_PSK`. In oder to prove the protocols, we present a new variant of pseudo-random functions, called *double pseudo-random function* (DPRF) and a strengthened variant of forward secrecy, named *asymmetric perfect forward secrecy* (APFS). We first give a brief description of all three protocols, and then introduce an extension of the ACCE security model for authentication protocols with pre-shared keys. Finally, we prove that all ciphersuite families of TLS-PSK meet our strong

notion of our security model. This result was joint work with Sven Schäge, Zheng Yang, Jörg Schwenk and Florian Kohlar. and published in the proceedings of the International Conference on Practice and Theory of Public-Key Cryptography (PKC) 2014 [LSY+14b] and in the IACR archive ePrint [LSY+14c].

- ***2: First tightly-secure authenticated key exchange protocol.***

  In order to answer the question: "Is it possible to construct a secure authenticated key exchange protocol with a tight security reduction in an enhanced Bellare-Rogaway security model under standards assumptions?", we construct the first tightly-secure authenticated key exchange (AKE) protocol, called tAKE, where security does not degrade with an increasing number of participating parties and protocol sessions. We describe a generic three-pass AKE protocol and prove its security in an enhanced Bellare-Rogaway security model under the standard assumption. Our construction is modular and enjoys a tight security reduction. This result was joint work with Christoph Bader, Dennis Hofheinz, Tibor Jager and Eike Kiltz and published in the proceedings of the International Conference on Theory of Cryptography Conference (TCC) 2015 [BHJ+15] and in the IACR archive ePrint [BHJ+14].

- ***3: New modular compilers for authenticated key exchange protocols.***

  "How can protocol designers modularly construct secure, efficient and reliable AKE systems?" For this question, we give two new efficient AKE compilers that generically construct secure and efficient AKE systems from Passive Key Exchange Module (PKEM) and Authentication Module (AM). Our first compiler is very efficient which relies on secure signature schemes and only requires two additional moves in which signatures are exchanged. The second compiler relies on public key encryption systems and accounts for scenarios where the parties do not have certified signature keys but only encryption keys. This scenarios can often occur in practice. This result was joint work with Sven Schäge, Zheng Yang, Jörg Schwenk and Christoph Bader and published in the proceedings of the International Conference on Applied Cryptography and Network Security (ACNS) 2014 [LSY+14a].

- ***4: A new theoretical attack: "no-match attack".***

  "Is this an appropriate choice if we use the notion of matching conversation as session identifier (MC-based sID) to prove the security for (all) authenticated key exchange protocols?" To answer this important question, we present a new theoretical attack, named here *no-match attack*, and show that proving security under the matching conversations as session identifier is a delicate issue. In particular, we provide several examples of protocols that claim to be secure under a security definition based on MC-based sID but where the security proof is actually flawed. We show that no-match attacks are often hard to avoid without costly modifications of the original

protocol. Simultaneously, we also give several ways to thwart no-match attacks. This result was joint work with Sven Schäge and Zheng Yang.

STRUCTURE OF THE THESIS.

The rest of this thesis is organized as follows:

- Chapter 2 providers a set of cryptographic basis and describes some preliminaries and definitions that will be used in this thesis.
- In Chapter 3, we introduce security models for AKE and ACCE protocols. The chapter shows how to incorporate the Bellare-Rogaway approach to modeling a security environment and produce a model-system description which is suitable for the security analysis of protocols. Firstly, we give security properties that theses communication protocols should satisfy. Then, we describe an overview of the most widely used security models for authenticated key exchange protocols. Next. we give a formal definition of passively-secure key exchange protocol and a corresponding security model. Moreover, we present an extension of Bellare-Rogaway (BR93) model and an extension of ACCE model used in the security analysis of our protocols. Finally, we give the relationship between the existing models and our models.
- In Chapter 4, we describe all three TLS-PSK ciphersuites and prove the security in an extended ACCE model. To prove our results on `TLS_RSA_PSK` and `TLS_DHE_PSK`, we introduce a new variant of pseudo-random functions (PRFs), called *double pseudo-random function* (DPRF) and prove the ACCE security of `TLS_RSA_PSK` with asymmetric perfect forward secrecy and `TLS_DHE_PSK` with perfect forward secrecy in the *standard model*.
- In Chapter 5, we construct the first tightly-secure AKE protocol in an enhanced under standard assumptions where security does not degrade with an increasing number of participating parties and protocol sessions, i.e., tight security reduction.
- In Chapter 6, we present new efficient AKE compilers that generically turn passively-secure key exchange (KE) protocols into authenticated key exchange (AKE) protocols where the security properties hold in the realistic setting of communication channels controlled by an active adversary.
- Finally, we present a new theoretical attack for key exchange protocols in Chapter 7, named *no-match attack*, and show that proving security under the matching conversations as session identifier (MC-based sID) is a delicate issue. In particular, we provide several examples of protocols that claim to be secure under a security definition based on MC-based sID but where the security proof is actually flawed. At the end of this chapter, we discuss several ways to thwart our *no-match* attacks.

Chapter 2

Preliminaries and Definitions

Contents

In this chapter we recall a set of cryptographic basic, and describe some preliminaries and definitions that will be used in this thesis. References for these topics include the books by Kranakis [Kra86], Menezes et al. [MvOV96], Schneier [Sch96] and Goldreich [Gol98, Gol01, Gol04, Gol10].

2.1 Number Theoretic Problems

In 1976, Whitfield Diffie and Martin Hellman first introduced the notion of public key cryptography. After that, there is a lot of research in public key cryptography and moreover, numerous schemes are proposed. All of these public key cryptographic systems based their security on the difficulty of solving some mathematical problem. In this section, we first simply describe the concept of computational security, then list a set of hard problems used in this thesis.

2.1.1 Computational Security for Cryptography

Many modern cryptosystems are proved computationally secure under some computational assumptions, e.g. RSA assumption, DDH assumption, CDH assumption, DL

assumption, etc. In computational security many assumptions are defined under the concept of Turing machine $\mathcal{TM}$ which was introduced in 1936 by Alan Turing. No such a probabilistic polynomial Turing machine $\mathcal{TM}$ which can solve some computational Problem with non-negligible probability. In this thesis, we use $\epsilon(\kappa)$ as a negligible function. For more information on $\mathcal{TM}$, see [Gol10]. To compare running times of algorithms, one uses some standard asymptotic notations [Gol01]. We represent these standard asymptotic notations as follows:

- $f(x) = \mathcal{O}(g(x))$;
- $f(x) = o(g(x)), (x \to \infty)$;

Suppose that two functions $f(x)$ and $g(x)$ are functions of an integer variable x. The expression $f(x) = \mathcal{O}(g(x))$ means that there exists constants x_0 and c such that $|f(x)| \leq c|f(x)|$ for all $x \geq x_0$. The notation $f(x) = o(g(x)), (x \to \infty)$ means that $g(x) \neq 0$ for sufficiently large x and $\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$. Note that if C is a positive constant, then $f(x) = \mathcal{O}(Cg(x))$ is equivalent to $f(x) = \mathcal{O}(g(x))$. In particular, $f(x) = \mathcal{O}(C)$ is equivalent to $f(x) = \mathcal{O}(1)$.

In cryptanalysis, some technical factors contribute to the time complexity of a cryptanalytic attack, e.g. key size, input length, processor speed, a amount of space, etc. With an algorithmic input size of l we denote the notions as follows: $\mathcal{O}(l)$ is linearly time bounded; $\mathcal{O}(l^x)$ is polynomial time bounded, where x is a constant; $\mathcal{O}(x^{p(l)})$ is exponentially time bounded, where x is a constant $x > 1$ and $p(l)$ is a polynomial.

Definition 2.1. *We say that a function $\epsilon(\kappa)$ is negligible if for every $c > 0$ there exists $\kappa_c > 0$ such that $\epsilon(\kappa) < \kappa^{-c}$ for all $\kappa > \kappa_c$.*

In the following subsections we describe some assumptions which are used in this thesis.

2.1.2 The Diffie-Hellman Assumption

The Diffie-Hellman problem can be classified into two types: the *Computational* Diffie-Hellman problem and the *Decisional* Diffie-Hellman problem. There are some variations of both types. We only give the descriptions used in this thesis in the next paragraphs. In describing these assumptions, we use the notations of the preceding subsection.

2.1.2.1 The Decisional Diffie-Hellman Assumption

The decision Diffie-Hellman problem DDH is defined as follows:

- Input: A cyclic group $\mathbb{G}$ of prime order p, g is a generator of $\mathbb{G}$, and three random elements g^a, g^b, $g^c \xleftarrow{\$} \mathbb{G}$;

- Output: Decide whether or not $g^c \stackrel{?}{=} g^{ab}$.

Definition 2.2 (DDH-Definition). *A* DDH *adversary $\mathcal{A}$ takes as input a tuple ($\mathbb{G}$, g, g^a, g^b, g^b) in $\mathbb{G}^4$ and outputs 0 or 1. We say that $\mathcal{A}$ $(t, \epsilon_{\mathsf{DDH}})$-breaks the decisional Diffie-Hellman (*DDH*) Assumption in $\mathbb{G}$ if it runs in time at most t,*

$$|\Pr[\mathcal{A}(\mathbb{G}, g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(\mathbb{G}, g, g^a, g^b, g^c) = 1]| \leq \epsilon_{\mathsf{DDH}},$$

where the probability is taken over $a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_p$.

2.2 Cryptographic Foundations

In this section we describe the foundations of cryptography that will be used in the subsequent chapters. The foundations of cryptography are the paradigms, approaches and techniques used to conceptualize, define and provide solutions to the cryptographic problems.

2.2.1 Hash Functions

Informally, a hash functions is applied to reduce messages of arbitrary length to binary strings of a fixed length ℓ. Naturally, we require a hash function to be *efficiently* computable. For cryptographic purposes a hash function must have at least one of the following properties: *one-wayness* and/or *collision resistance*. In this subsection we describe the security definitions of *collision-resistant hash function* and *target collision-resistant hash function*.

2.2.1.1 Collision-Resistant Hash Function

A *collision-resistant hash function* is a deterministic algorithm CRHF: $\mathcal{K}_{\mathsf{CRHF}} \times \mathcal{M} \rightarrow \{0,1\}^\ell$ which given a key $k \in \mathcal{K}_{\mathsf{CRHF}}(1^\kappa)$[1] and a bit string $\mathsf{M} \in \mathcal{M}$ outputs a hash value $w = \mathsf{CRHF}(k, \mathsf{M})$ in the hash space $\{0,1\}^\ell$ with ℓ polynomial in κ.

The security of a collision-resistant hash function CRHF is captured via the following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- The challenger $\mathcal{C}$ selects a key $k \in \mathcal{K}_{\mathsf{CRHF}}$ and gives k to $\mathcal{A}$.
- The attacker $\mathcal{A}$ outputs two strings M_0 and $\mathsf{M}_1 \in \mathcal{M}$.

[1] We assume that there is a PPT algorithm $\mathcal{K}_{\mathsf{CRHF}}$ that on input 1^κ produces a random key k, where κ is the security parameter.

- The attacker $\mathcal{A}$ wins the security game if and only if $\mathsf{M}_0 \neq \mathsf{M}_1$ and $\mathsf{CRHF}(k, \mathsf{M}_0) = \mathsf{CRHF}(k, \mathsf{M}_1)$.

Definition 2.3. *We say that* CRHF *is a* $(t, \epsilon_{\mathsf{CRHF}})$-secure *collision-resistant hash function, if all* t*-time adversaries* $\mathcal{A}$ *have an advantage of at most* ϵ_{CRHF}*, that is*

$$\Pr\left[\begin{array}{c}\mathsf{CRHF}(k, \mathsf{M}_0) = \mathsf{CRHF}(k, \mathsf{M}_1) \bigwedge \mathsf{M}_1 \neq \mathsf{M}_0 \bigwedge \mathsf{M}_1, \mathsf{M}_0 \in \mathcal{M}_{\mathsf{CRHF}} : \\ k \in \mathcal{K}_{\mathsf{CRHF}}; (\mathsf{M}_0, \mathsf{M}_1) \leftarrow \mathcal{A}(1^\kappa, k)\end{array}\right] \leq \epsilon_{\mathsf{CRHF}}.$$

2.2.2 Pseudo-Random Functions.

Let $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{M}_{\mathsf{PRF}} \rightarrow \mathcal{R}_{\mathsf{PRF}}$ denote a family of deterministic functions, where $\mathcal{K}_{\mathsf{PRF}}$ is the key space, $\mathcal{M}_{\mathsf{PRF}}$ is the domain and $\mathcal{R}_{\mathsf{PRF}}$ is the range of PRF.

To define security, we consider the following security game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Let $\pi_{\mathsf{PRF}}(\cdot)$ denote an oracle implemented by $\mathcal{C}$, which takes as input a message $m \in \mathcal{M}_{\mathsf{PRF}}$ and outputs a value $z \in \mathcal{R}_{\mathsf{PRF}}$.

1. The challenger samples $b \xleftarrow{\$} \{0, 1\}$. If b = 0, the challenger samples $k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$ and assigns oracle $\pi_{\mathsf{PRF}}(\cdot)$ to $\mathsf{PRF}(k, \cdot)$. If b = 1, the challenger assigns oracle $\pi_{\mathsf{PRF}}(\cdot)$ to $\mathsf{RF}(\cdot)$ which is a truly random function associated with the same range and domain as $\mathsf{PRF}(k, \cdot)$.
2. The adversary may adaptively make queries m_i for $1 \leq i \leq q$ to oracle $\pi_{\mathsf{PRF}}(\cdot)$ and receives the result depending on the random bit b.
3. Finally, the adversary outputs its guess $b' \in \{0, 1\}$ of b. If $b = b'$ the adversary wins.

We assume that the overall number of queries made to the challenger is $q = q(\kappa)$. As before, we call $Pr\left[b = b'\right]$ the success probability of the adversary in winning the above game. We call $|\Pr\left[b = b'\right] - 1/2|$ the advantage of the adversary.

Definition 2.4. *We say that* PRF *is a* $(q,\ t, \epsilon_{\mathsf{PRF}})$-secure *pseudo-random function, if any adversary running in time* t *has an advantage of at most* ϵ *to distinguish the* PRF *from a truly random function, i.e.,*

$$\Pr\left[b = b'\right] \leq 1/2 + \epsilon_{\mathsf{PRF}},$$

while the number of allowed queries q *is upper bounded by* t*.*

2.2.3 Message Authentication Code

We first recall the notions of general message authentication codes scheme. A message authentication code MAC consists of three probabilistic algorithms (MAC.KGen, MAC.Tag, MAC.Vfy) with associated key space $\mathcal{K}_{\mathsf{MAC}}$, message space $\mathcal{M}_{\mathsf{MAC}}$ and tag space $\mathcal{T}_{\mathsf{MAC}}$.

- $\mathsf{K}_{\mathsf{MAC}} \xleftarrow{\$} \mathsf{MAC.KGen}(1^\kappa)$: The probabilistic key-generation algorithm takes as input a security parameter κ and outputs a secret key $\mathsf{K}_{\mathsf{MAC}} \in \mathcal{K}_{\mathsf{MAC}}$.
- $\rho \leftarrow \mathsf{MAC.Tag}(\mathsf{K}_{\mathsf{MAC}}, m)$: The algorithm MAC.Tag is a deterministic algorithm which takes as input a secret key $\mathsf{K}_{\mathsf{MAC}} \in \mathcal{K}_{\mathsf{MAC}}$ and a message $m \in \mathcal{M}_{\mathsf{MAC}}$ and outputs an authentication tag $\rho \in \mathcal{T}_{\mathsf{MAC}}$.
- $b \in \{0,1\} \leftarrow \mathsf{MAC.Vfy}(\mathsf{K}_{\mathsf{MAC}}, m, \rho)$: The deterministic verification algorithm MAC.Vfy takes as input a secret key $\mathsf{K}_{\mathsf{MAC}} \in \mathcal{K}_{\mathsf{MAC}}$, a message $m \in \mathcal{M}_{\mathsf{MAC}}$ and a tag $\rho \in \mathcal{T}_{\mathsf{MAC}}$ and outputs $b = 1$ if it accepts. Otherwise, it returns $b = 0$. In practice, we do not need to specify a separate tag-verification algorithm. Here we assume that the tag-verification works the same way: the receiver, having received (m, ρ), computes $\rho' \leftarrow \mathsf{MAC.Tag}(\mathsf{K}_{\mathsf{MAC}}, m)$. If $\rho' = \rho$ then $b = 1$; otherwise, $b = 0$.

The (UF-CMA) security of a MAC scheme MAC = (MAC.KGen, MAC.Tag, MAC.Vfy) via the following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger $\mathcal{C}$ computes $\mathsf{K}_{\mathsf{MAC}} \xleftarrow{\$} \mathsf{MAC.KGen}(1^\kappa)$.
2. The adversary $\mathcal{A}$ may adaptively query $q \in \mathbb{N}$ messages m_i of his choice, where i is an index, $1 \leq i \leq q$.
3. The challenger $\mathcal{C}$ replies to each query with $\rho_i \leftarrow \mathsf{MAC.Tag}(\mathsf{K}_{\mathsf{MAC}}, m_i)$.
4. Eventually, $\mathcal{A}$ outputs a fresh pair (m^*, ρ^*). If $\mathsf{MAC.Vfy}(\mathsf{K}_{\mathsf{MAC}}, m^*, \rho^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$, the adversary $\mathcal{A}$ wins.

We assume that the overall number of queries made to the challenger $\mathcal{C}$ is $q = q(\kappa)$.

Definition 2.5. *We say that a message authentication code scheme* MAC *is* $(q, t, \epsilon_{\mathsf{MAC}})$*-secure against forgeries under adaptive chosen-message attacks (UF-CMA), if for all adversaries* $\mathcal{A}$ *that run in time* t *and queries at most* q *messages holds that*

$$\Pr\left[(m^*, \rho^*) \xleftarrow{\$} \mathcal{A}(1^\kappa) : \mathsf{MAC.Vfy}(\mathsf{K}_{\mathsf{MAC}}, m^*, \rho^*) = 1 \wedge m^* \notin \{m_1, \ldots, m_q\}\right] \leq \epsilon_{\mathsf{MAC}},$$

while the number of allowed query at most q *is upper bounded by* t*.*

2.2.3.1 One-Time Message Authentication Code

Definition 2.6. *We say that a one-time message authentication code scheme* OTM *is* $(t, \epsilon_{\mathsf{OTM}})$*-secure, if* OTM *is a* $(1, t, \epsilon_{\mathsf{OTM}})$*-secure message authentication code scheme in the sense of Definition 2.5.*

2.2.4 Public-Key Encryption

A public key encryption (PKE) scheme consists of three polynomial time algorithms PKE =(PKE.KGen, PKE.Enc, PKE.Dec) with the following semantics:

- $(pk, sk) \xleftarrow{\$} \mathsf{PKE.KGen}(1^\kappa)$: is a probabilistic polynomial-time key generation algorithm which generates a encryption/decryption key pair $(pk,\ sk) \in \mathcal{SK} \times \mathcal{PK}$ on input of the security parameter κ,
- $C \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m)$: is a probabilistic polynomial-time encryption algorithm which takes as inputs a public key $pk \in \mathcal{PK}$ and a message $m \in \mathcal{M}$ and outputs ciphertext $C \in \mathcal{C}$.
- $m \leftarrow \mathsf{PKE.Dec}(sk, C)$: is a deterministic polynomial-time decryption algorithm which takes as input a key sk and a ciphertext C, and outputs either $m \in \mathcal{M}$ or an error symbol $\perp$.

CORRECTNESS. For consistency, we require the correctness properties. For all $(pk,\ sk)$ $\xleftarrow{\$} \mathsf{PKE.KGen}(1^\kappa)$, and all $C \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m)$ we have

$$\Pr\left[m \leftarrow \mathsf{PKE.Dec}(sk, C)\right] = 1,$$

where the probability is the coins of all the algorithms described above.

SECURITY GAME. The (IND-CCA) security of a PKE scheme PKE = (PKE.KGen, PKE.Enc, PKE.Dec) is captured via the following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger $\mathcal{C}$ computes $(pk, sk) \xleftarrow{\$} \mathsf{PKE.KGen}(1^\kappa)$ and gives pk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ may adaptively decrypt polynomially (in κ) many ciphertexts C of his choice. At any point $\mathcal{A}$ may query PKE.Enc oracle, which takes two messages m_0 and m_1 from the message space.
3. $\mathcal{C}$ samples $b \xleftarrow{\$} \{0, 1\}$ and computes $C^* \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m_b)$ and sends C^* to $\mathcal{A}$.
4. $\mathcal{A}$ may adaptively decrypt polynomially (in κ) many ciphertexts C of his choice with the restriction that C^* is not among the values queried by $\mathcal{A}$.
5. $\mathcal{A}$ outputs his guess $b' \in \{0, 1\}$ of b. If $b = b'$ the adversary wins.

We assume that the overall number of queries made to $\mathcal{C}$ is $q = q(\kappa)$. We call $Pr\left[b = b'\right]$ the success probability of the adversary in winning the above game. We call $|\Pr\left[b = b'\right] - 1/2|$ the advantage of the adversary.

Definition 2.7 (IND-CCA-PKE Security). *We say that a public encryption scheme is a $(q, t, \epsilon_{\mathsf{PKE}})$-secure (IND-CCA)* PKE *scheme, if an adversary running in time t in the above security game has an advantage of at most ϵ_{PKE}, i.e.,*

$$\Pr\left[b = b'\right] \leq 1/2 + \epsilon_{\mathsf{PKE}},$$

while the number of allowed queries q is upper bounded by t.

IND-CPA security is a weaker security notion than IND-CCA security. In other words, IND-CPA security for PKE scheme is based on the same security game as IND-CCA security except for, in *step 2* and *step 4*, where $\mathcal{A}$ cannot access the decryption oracle.

Definition 2.8 (IND-CPA-PKE Security). *We say that a public encryption scheme is a* $(t, \epsilon_{\mathsf{PKE}})$*-secure (IND-CPA)* PKE *scheme, if an adversary within running time* t *in the above security game without accessing the decryption oracles has an advantage of at most* ϵ*, i.e.,*

$$\Pr\left[b = b'\right] \leq 1/2 + \epsilon_{\mathsf{PKE}}.$$

2.2.5 Key Encapsulation Mechanism

In this subsection, we introduce key encapsulation mechanism in multi-user setting with corruptions ($\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$), and describe the definitions of this primitive, security games and security definition [BHJ+15].

2.2.5.1 Key Encapsulation Mechanism in the Multi-User Setting with Corruptions

A key encapsulation mechanism with corruptions $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ consists of three probabilistic polynomial algorithms:

- $(sk, pk) \xleftarrow{\$} \mathsf{KEM.Gen}^{\mathsf{C}}_{\mathsf{MU}}(1^{\kappa})$: This algorithm $\mathsf{KEM.Gen}^{\mathsf{C}}_{\mathsf{MU}}$ takes as input the security parameter 1^{κ}, and outputs a key pair $(sk, pk) \in \mathcal{SK} \times \mathcal{PK}$.
- $(\mathsf{K}, C) \xleftarrow{\$} \mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}(pk)$: This algorithm $\mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}$ takes as input a long-term public key pk, and outputs a ciphertext $C \in \mathcal{C}$ along with a session key $\mathsf{K} \in \mathcal{K}$.
- $\mathsf{K} \leftarrow \mathsf{KEM.Decap}^{\mathsf{C}}_{\mathsf{MU}}(sk, C)$: This algorithm $\mathsf{KEM.Decap}^{\mathsf{C}}_{\mathsf{MU}}$ takes as input a long-term secret key sk and a ciphertext C, and outputs a key $\mathsf{K} \in \mathcal{K}$ or an error symbol $\perp$.

CORRECTNESS. For consistency, we require the correctness properties. For all $(\mathsf{K}, C) \xleftarrow{\$} \mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}(pk)$ we have

$$\Pr\left[\mathsf{K} \leftarrow \mathsf{KEM.Decap}^{\mathsf{C}}_{\mathsf{MU}}(sk, C)\right] = 1,$$

where the probability is taken over the choice of $(sk, pk) \xleftarrow{\$} \mathsf{KEM.Gen}^{\mathsf{C}}_{\mathsf{MU}}(1^{\kappa})$ and the coins of all the algorithms described above.

SECURITY GAME. The (MU-C-IND-CPA) security of a key encapsulation mechanism scheme in the multi-user setting with corruptions of secret keys is captured via the

following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ and that is parametrized by two numbers, $\ell \in \mathbb{N}$ denoted here as the number of honest parties and $d \in \mathbb{N}$ as the maximum number of encapsulation per party. An adversary $\mathcal{A}$ is able to interact with the execution environment by issuing two queries: Encaps and $\mathsf{Corrupt}$.

1. At the beginning of the game, $\mathcal{C}$ generates public parameters and $(\ell \cdot d)$ key pairs $(sk_i^s, pk_i^s) \xleftarrow{\$} \mathsf{KEM.Gen}_{\mathsf{MU}}^{\mathsf{C}}(1^\kappa)$ for each $i \in [\ell]$ and $s \in [d]$. Then, it gives public parameters and all public keys $\{pk_i^s\}$, $i \in [\ell]$ and $s \in [d]$, to $\mathcal{A}$. Finally, $\mathcal{C}$ selects $b_i^s \xleftarrow{\$} \{0,1\}$ and initializes a set $\mathcal{S}^{\mathsf{Corrupt}} := \emptyset$ to keep track of the corrupted keys.
2. The adversary $\mathcal{A}$ may adaptively issue q Encaps and $\mathsf{Corrupt}$ queries. $\mathsf{Corrupt}(pk_i^s)$ takes as input a public key pk_i^s supplied by $\mathcal{A}$, and responds with sk_i^s to $\mathcal{A}$. The index (i, s) has been saved to $\mathcal{S}^{\mathsf{Corrupt}}$, i.e., $\mathcal{S}^{\mathsf{Corrupt}} := \{(\mathrm{i, s})\}$, $i \in [\ell]$ and $s \in [d]$. For Encaps queries made by $\mathcal{A}$ with pk_i^s, the challenger $\mathcal{C}$ generates a (ciphertext and key) pair $(\mathsf{K}_i^s, C_i^s) \xleftarrow{\$} \mathsf{KEM.Encap}_{\mathsf{MU}}^{\mathsf{C}}(pk_i^s)$ and sets $\mathsf{K}_{i,0}^s = \mathsf{K}_i^s$. Then, it selects a random session key $\mathsf{K}_{i,1}^s \in \mathcal{K}$. Finally, $\mathcal{C}$ responds with $(\mathsf{K}_{i,b_i^s}^s, C_i^s)$ to $\mathcal{A}$.
3. Finally, $\mathcal{A}$ outputs his guess $b' \in \{0,1\}$ of b_i^s. If $b_i^s = b'$ the adversary $\mathcal{A}$ wins this security game.

We assume that the overall number of queries made to the challenger is $q = q(\kappa)$. We call $\Pr[b_i^s = b']$ the success probability of the adversary in winning the above game. We call $|\Pr[b_i^s = b'] - 1/2|$ the advantage of the adversary.

Definition 2.9 (MU-C-IND-CPA-KEM Security). *We say that a key encapsulation mechanism scheme in the multi-user setting with corruptions* $\mathsf{KEM}_{\mathsf{MU}}^{\mathsf{C}}$ *is a* $(d, \ell, t, \epsilon_{\mathsf{KEM}_{\mathsf{MU}}^{\mathsf{C}}})$*-secure (MU-C-IND-CPA)* $\mathsf{KEM}_{\mathsf{MU}}^{\mathsf{C}}$ *scheme, if an adversary running in time* t *has an advantage of at most* $\epsilon_{\mathsf{KEM}_{\mathsf{MU}}^{\mathsf{C}}}$*, i.e.,*

$$\Pr\left[b_i^s = b' \wedge (s,i) \notin \mathcal{S}^{\mathsf{Corrupt}}\right] \leq 1/2 + \epsilon_{\mathsf{KEM}_{\mathsf{MU}}^{\mathsf{C}}},$$

while the number of allowed queries is upper bounded by t*.*

2.2.6 Digital Signatures

In this subsection, we introduce standard security notion for digital signature scheme as proposed by Goldwasser, Micali, and Rivest [GMR88], named SU-EUF-CMA, and an extension of this notion, i.e., digital signature in multi-user setting with corruptions ($\mathsf{SIG}^{\mathsf{C}}$), named MU-EUF-CMA.

2.2.6.1 Standard Digital Signature Scheme

First, we give the standard digital signature definition. A digital signature scheme SIG consists of the following algorithms:

- $(sk, vk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$: The key generation algorithm $\mathsf{SIG.Gen}$ that on input 1^κ outputs $vk \in \mathcal{PK}$ and $sk \in \mathcal{SK}$, where $\mathcal{PK}$ is the verification key space and $\mathcal{SK}$ is the private key space;
- $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(sk, m)$: The signature algorithm $\mathsf{SIG.Sign}$ that on input sk and a massage $m \in \mathcal{M}_{\mathsf{SIG}}$, outputs a signature $\sigma \in \mathcal{S}_{\mathsf{SIG}}$, where $\mathcal{M}_{\mathsf{SIG}}$ is the message space and $\mathcal{S}_{\mathsf{SIG}}$ is the signature space;
- $b \in \{0, 1\} \leftarrow \mathsf{SIG.Vfy}(vk, m, \sigma)$: The verification algorithm $\mathsf{SIG.Vfy}$ that on input a verification key vk, a message m and a signature σ, outputs $b = 1$ if σ is a valid signature for m under the verification key vk, and 0 otherwise.

2.2.6.2 Standard SU-EUF-CMA Security Definition

The standard security definition for signature schemes in the single user setting is *existential unforgeability under chosen-message attacks* proposed by Goldwasser, Micali and Rivest [GMR88].

SECURITY GAME. We consider the following security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger $\mathcal{C}$ generates a key pair $(vk, sk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$ and gives vk to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ may adaptively query polynomially (in κ) many messages m_i of his choice, where i is an index, $1 \leq i \leq q$ for some $q \in \mathbb{N}$.
3. The challenger $\mathcal{C}$ responds to each of the sign-queries with a corresponding signature $\sigma_i \xleftarrow{\$} \mathsf{SIG.Sign}(sk, m_i)$.
4. The adversary $\mathcal{A}$ outputs a fresh message/signature pair (m^*, σ^*). If $\mathsf{SIG.Vfy}(vk, m^*, \sigma^*) = 1$ and $m^* \notin \{m_1, \ldots, m_q\}$, the adversary $\mathcal{A}$ wins.

Definition 2.10. *We say that a signature scheme* SIG *is* $(q, t, \epsilon_{\mathsf{SIG}})$*-secure against existential unforgeability under adaptive chosen-message attack (EUF-CMA), if in the above security game for all adversaries* $\mathcal{A}$ *running in time* t *holds that*

$$\Pr\left[(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}(1^\kappa) : \mathsf{SIG.Vfy}(vk, m^*, \sigma^*) = 1 \wedge m^* \notin \{m_1, \ldots, m_q\}\right] \leq \epsilon_{\mathsf{SIG}},$$

while the number of allowed queries q *is upper bounded by* t*.*

2.2.6.3 Digital Signature in the Multi-User Setting with Corruptions

In this subsection, we introduce digital signature schemes and their security in the multi-user setting with corruptions, i.e., *Existential Unforgeability under adaptive Chosen-Message attacks in the Multi-User setting with adaptive Corruptions*, named MU-C-EUF-CMA [BHJ+15]. The (MU-C-EUF-CMA) security of a signature scheme is captured via the following game that is played between a challenger $\mathcal{C}$ and an adversary

$\mathcal{A}$. Note that in this security game $\mathcal{A}$ can access a sign oracle $\mathcal{O}^{\mathsf{SIG}}$ and corrupt oracle $\mathcal{O}^{\mathsf{Corrupt}}$. Fix a set of honest parties $\{ \mathsf{P}_1, \ldots, \mathsf{P}_\ell \}$, where each honest party $\mathsf{P}_i \in \{ \mathsf{P}_1, \ldots, \mathsf{P}_\ell \}$ is a potential protocol participant and has a pair of long-term verification/private key $(vk_i, sk_i) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$ that corresponds to its identity i.

1. The challenger $\mathcal{C}$ generates key pairs $(vk_i, sk_i) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\kappa)$, $i \in [\ell]$ and gives vk_i to the adversary $\mathcal{A}$, where ℓ is the number of verification keys. Moreover, $\mathcal{C}$ initializes a set $\mathcal{S}^{\mathsf{Corrupt}}$ to keep track of corrupted keys, and ℓ sets $\{ \mathcal{S}_1, \cdots, \mathcal{S}_\ell \}$ to keep track of the sign-message queries for each party.
2. Then, $\mathcal{A}$ may adaptively make two different types of queries, i.e., SIGN-query and Corrupt-query.
   - Corrupt-query: $\mathcal{A}$ supplies an identity $i \in [\ell]$. Then the challenger $\mathcal{C}$ returns the corresponding secret key sk_i. Moreover, $\mathcal{C}$ updates $\mathcal{S}^{\mathsf{Corrupt}} := \mathcal{S}^{\mathsf{Corrupt}} \cup \{i\}$.
   - SIGN-query: $\mathcal{A}$ supplies a pair (Pid_i, m_i^j) of his choice, where $i \in [\ell]$ is an identity and j is a message-index, $1 \leq j \leq q_{\mathsf{SIG}}$ for some $q_{\mathsf{SIG}} \in \mathsf{N}$. The challenger $\mathcal{C}$ responds to each of the sign-queries with a corresponding signature $\sigma_i^j \xleftarrow{\$} \mathsf{SIG.Sign}(sk_i, m_i^j)$ and updates $\mathcal{S}_i := \mathcal{S}_i \cup \{ (m_i^j, \sigma_i^j) \}$.
3. The adversary $\mathcal{A}$ outputs a message/signature pair (i^*, m', σ'_{i^*}).
4. If $\mathsf{SIG.Vfy}(vk_{i^*}, m', \sigma'_{i^*}) = 1$ and $(m'_{i^*}, \sigma'_{i^*}) \notin \mathcal{S}_{i^*}$ and $i^* \notin \mathcal{S}^{\mathsf{Corrupt}}$, the adversary $\mathcal{A}$ wins.

Definition 2.11. *We say that a signature scheme* SIG *is (q_{SIG}, ℓ, t, ϵ_{SIG})-secure against existential unforgeability under adaptive chosen-message attacks in multi-user setting with corruptions (MU-C-EUF-CMA), if in the above security game for any adversaries $\mathcal{A}$ running in time t holds that*

$$\Pr\left[\begin{array}{c}(i^*, m'_{i^*}, \sigma'_{i^*}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\cdot)}(1^\kappa) : \\ \mathsf{SIG.Vfy}(vk_{i^*}, m'_{i^*}, \sigma'_{i^*}) = 1 \wedge (m'_{i^*}, \sigma'_{i^*}) \notin \mathcal{S}_{i^*} \wedge i^* \notin \mathcal{S}^{\mathsf{Corrupt}}\end{array}\right] \leq \epsilon_{\mathsf{SIG}},$$

while the number of allowed queries q_{SIG} is upper bounded by t.

Definition 2.12 (One-Time Signature). *We say that a one-time signature scheme* SIG *is (1, ℓ, t, $\epsilon_{\mathsf{OTSIG}}$)-secure against strong existential unforgeability under adaptive chosen-message attacks in multi-user setting without corruptions (MU-OTS-EUF-CMA), if in the above security game for any adversaries $\mathcal{A}$ running in time t holds that*

$$\Pr\left[\begin{array}{c}(i^*, m'_{i^*}, \sigma'_{i^*}) \xleftarrow{\$} \mathcal{A}(1^\kappa) : \\ \mathsf{SIG.Vfy}(vk_{i^*}, m', \sigma'_{i^*}) = 1 \wedge (m'_{i^*}, \sigma'_{i^*}) \notin \mathcal{S}_{i^*} \wedge \\ \mathcal{S}^{\mathsf{Corrupt}} = \emptyset \wedge |S_i| \leq 1 \wedge i \in [\ell]\end{array}\right] \leq \epsilon_{\mathsf{OTSIG}}.$$

2.2.7 Stateful Length-Hiding Authenticated Encryption

We use the stateful variant of LHAE security (sLHAE) as originally defined by Paterson et al. [PRS11b] and used by JKSS [JKSS12] for their analysis of TLS-DHE. A *stateful symmetric encryption scheme* basically consists of two algorithms $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$.

- $(C, \mathsf{st}'_e) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, \mathsf{Hd}, m, \mathsf{st}_e)$: takes as input a secret key $k \in \mathcal{K}_{sLHAE}$, an output ciphertext length $\mathsf{len} \in \mathbb{N}$, some header data $\mathsf{Hd} \in \{0,1\}^*$, a plaintext $m \in \mathcal{M}_{sLHAE}$, and the current state $\mathsf{st}_e \in \{0,1\}^*$, and outputs either a ciphertext $C \in \{0,1\}^{\mathsf{len}}$ and an updated state st'_e or an error symbol $\perp$ if for instance the output length len is not valid for the message m, e.g. $\mathsf{len} < |\mathcal{M}_{sLHAE}|$, or the encryption algorithm does not support ciphertexts of length len.
- $(m', \mathsf{st}'_d) = \mathsf{StE.Dec}(k, \mathsf{Hd}, C, \mathsf{st}_d)$: takes as input a key k, header data Hd, a ciphertext C, and the current state $\mathsf{st}_d \in \{0,1\}^*$, and returns an updated state st'_d and a value m' which is either the message encrypted in C, or a distinguished error symbol $\perp$ indicating that C is not a valid ciphertext.

Both, encryption state st_e and decryption state st_d are initialized to the empty string $\emptyset$. Algorithm $\mathsf{StE.Enc}$ may be probabilistic, while $\mathsf{StE.Dec}$ is always deterministic. See [PRS11b] for more details.

The security is formalized in the following security game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. First, we describe how the oracles $\mathcal{O}^{\mathsf{ENC}}$ and $\mathcal{O}^{\mathsf{DEC}}$ respond to $\mathcal{A}$'s queries as follows:

- $\mathsf{ENC}(m_0, m_1, \mathsf{len}, \mathsf{Hd})$:
  1: $u := u + 1$;
  2: $(C^{(0)}, \mathsf{st}_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, \mathsf{Hd}, m_0, \mathsf{st}_e)$;
  3: $(C^{(1)}, \mathsf{st}_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \mathsf{len}, \mathsf{Hd}, m_1, \mathsf{st}_e)$;
  4: **IF** $(C^{(0)} = \perp$ OR $C^{(1)} = \perp)$ **THEN RETURN** $\perp$;
  5: $(C_u, \mathsf{Hd}_u, \mathsf{st}_e) := (C^{(b)}, \mathsf{Hd}, \mathsf{st}_e^{(b)})$;
  6: **RETURN** C_u.
- $\mathsf{DEC}(C, \mathsf{Hd})$:
  1: $v := v + 1$;
  2: **IF** $b = 0$ **THEN RETURN** $\perp$;
  3: $(m, \mathsf{st}_d) = \mathsf{StE.Dec}(k, \mathsf{Hd}, C, \mathsf{st}_d)$;
  4: **IF** $v > u$ **OR** $C \neq C_v$ **OR** $\mathsf{Hd} \neq \mathsf{Hd}_v$; **THEN** $\mathsf{phase} := 1$;
  5: **IF** $\mathsf{phase} = 1$ **THEN RETURN** m;
  6: **RETURN** $\perp$.

The values u, v and phase are all initialized to 0 at the beginning of the security game. Note that the security game described above gives an attacker access to select messages (m_0, m_1) of arbitrary lengths and a chosen ciphertext-length len.

1. The challenger $\mathcal{C}$ selects $b \xleftarrow{\$} \{0,1\}$ and $k \xleftarrow{\$} \{0,1\}^\kappa$ and sets $\mathsf{st}_e := \emptyset$ and $\mathsf{st}_d := \emptyset$,
2. $\mathcal{A}$ may adaptively query the encryption oracle ($\mathcal{O}^{(\mathsf{ENC})}$) q_{ENC} times and the decryption oracle ($\mathcal{O}^{(\mathsf{DEC})}$) q_{DEC} times.
3. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$.

We assume that the overall number of queries made to $\mathcal{C}$ is $q = q(\kappa)$. We call $Pr\left[b = b'\right]$ the success probability of the adversary in winning the above game. We call $|\Pr\left[b = b'\right] - 1/2|$ the advantage of the adversary.

Definition 2.13. *We say that the stateful symmetric encryption scheme* StE=*(*$\mathsf{StE.Enc}$*,* $\mathsf{StE.Dec}$*) is* (t, ϵ)-secure*, if any adversary running in time* t *has an advantage of at most* ϵ *to output* b' *such that* $b' = b$*, i.e.,*

$$\Pr\left[b' = b\right] \leq 1/2 + \epsilon,$$

while the number of allowed queries q *is upper bounded by* t*.*

2.2.8 CCLA-2-Secure Public-Key Cryptosystem

Dziembowski et al. [DF11] constructed an adaptively chosen ciphertext after-the-fact leakage-resilient public-key cryptosystem which is secure against continuous leakage. We use the same definition described in [ABS14].

A public-key encryption scheme PKE =($\mathsf{PKE.KGen}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$) consists of three PPT algorithms. This algorithms is similar to that described in 2.2.4.

The (CCLA2) security of a PKE scheme PKE =($\mathsf{PKE.KGen}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$) is defined via the following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger $\mathcal{C}$ computes $(pk, sk) \xleftarrow{\$} \mathsf{PKE.KGen}(1^\kappa)$ and gives pk to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ is given access to the decryption oracle and the leakage oracle. Note that $\mathcal{A}$ is able to get the leakage of the secret key sk using the leakage oracle. We denote the λ-leakage oracle that any adversary $\mathcal{A}$ can select efficiently computable leakage function f, $f(sk) \leftarrow \mathsf{Leakage}(f)$ when $|f(sk)| \leq \lambda$.
3. At any point $\mathcal{A}$ may query $\mathsf{PKE.Enc}$ oracle, which takes two messages m_0 and m_1 from the message space. $\mathcal{C}$ samples $b \xleftarrow{\$} \{0,1\}$ and computes $C^* \xleftarrow{\$} \mathsf{PKE.Enc}(pk, m_b)$ and sends C^* to $\mathcal{A}$.

4. Then, $\mathcal{A}$ may adaptively query the decryption oracle and the leakage oracle. However, the ciphertexts of his choice with the restriction that C^* is not among the values queried by $\mathcal{A}$.
5. $\mathcal{A}$ outputs his guess $b' \in \{0,1\}$ of b. If $b = b'$ the adversary wins.

We assume that the overall number of queries made to $\mathcal{C}$ is $q = q(\kappa)$. We call $Pr[b = b']$ the success probability of the adversary in winning the above game. We call $|\Pr[b = b'] - 1/2|$ the advantage of the adversary.

Definition 2.14 (CCLA-2-PKE Security). *We say that a public encryption scheme is a* $(q, t, \epsilon_{\mathsf{PKE}})$*-secure (CCLA-2)* PKE *scheme, if an adversary within running time* t *in the above security game has an advantage of at most* ϵ_{PKE}*, i.e.,*

$$\Pr[b = b'] \leq 1/2 + \epsilon_{\mathsf{PKE}},$$

while the number of allowed queries q *is upper bounded by* t*.*

Chapter 3

Formal Security Models for Cryptographic Protocols

Contents

A two-party authenticated key exchange protocol (AKE) allows two parties to exchange a secret session key, and an authenticated and confidential channel establishment protocol (ACCE) achieves a secure channel between the two users. Intuitively speaking, the main security goals for AKE protocols are that session keys cannot be distinguished from random values by adversary (key indistinguishability), and that a party really generates a session key with its intended partner, and not with some other user. In contrast with AKE protocols, ACCE protocols consider the security of the record layer protocol, i.e., the ciphertext security is integrated in the model. Roughly speaking, in the ACCE model no challenge key is ever given to the adversary, the adversary wins the ACCE security game if it can distinguish between encryptions of two distinct messages (ciphertext indistinguishability).

In order to formally analyze the security of these cryptographic protocols, one needs a suitable (formal) model. In this chapter, we will introduce the security models used in this thesis. Firstly, we give security properties that these communication protocols

should satisfy in Section 3.1. Then, we give an informal overview of the widely-used security models for authenticated key exchange protocols, i.e., the Bellare-Rogaway (BR93) and the (extended) Canetti-Krawczyk models in Section 3.2. Since passively-secure key exchange protocol is the most important building block of our compilers, we present a formal definition of passively-secure key exchange protocol and a corresponding security model in Section 3.3. In Section 3.4, we describe two security models for our AKE compilers and tightly-secure authenticated key exchange protocol. Moreover, we present an extension of the formal security model for two-party authenticated and confidential channel establishment (ACCE) protocols introduced by Jager et al. in [JKSS12] to also cover scenarios with pre-shared, symmetric keys in Section 3.5. Finally, we discuss the relationship between the previously proposed models in Section 3.6. Our main focus lies on the security of two-party AKE and ACCE protocols in the presence of active adversaries.

Remark 3.1. Typically, there are two main approaches to capture security of cryptographic protocols. One approach is based on the simulation paradigm, namely simulation-based security model, such as the framework of universal composability [Can00, CK02b, CLOS02, NMO05, KL05, Küs06, FHH14]. The other approach uses game to model security. Namely, one proves security using the sequence-of-games approach [Sho04, BR06]. For key exchange, the game is won if the adversary can distinguish the correct session key from a random key. In this thesis, we only consider game-based security models for our design and analysis of cryptographic protocols.

3.1 Security Properties of Key Exchange Protocols

In this section, we introduce a number of distinct security requirements for key exchange protocols. Before we begin to describe the requirements, we first classify two types of attacks as follows:

- **Passive Attacks**: We formalize a notion of passive attacks, where the adversary is limited to manage the delivery of the messages exchanged by the communication parties. But it can not inject, delete or manipulate messages;
- **Active Attacks**: In contrast to passive attacks, active attacks, where an adversary is allowed to interact actively with the communicating parties, and can additionally disrupt the communications in any possible way in order to achieve its goal, e.g. by inserting, deleting, or modifying messages on any protocol communication.

Clearly, a secure protocol should be able to withstand both passive attacks and active attacks. Protocols for AKE or ACCE allow parties within an insecure network to establish a common session key which can then be used to secure their future communication. Therefore, it is for precisely this reason that a comprehensive security model is very

useful for the analysis of the cryptographic protocols. A number of security properties are generally believed to be necessary for key agreement protocols [BR93a, BWJM97, BWM99b]. In the following, we informally describe the security attributes for a secure AKE/ACCE protocol as follows.

- **Session Keys Expose Resilience (SK-E).** Even if some session keys are compromised by the adversary, the secrecy of the session keys established in other sessions should not be affected.
- **Prefect Forward Secrecy (PFS).** Informally, this property holds if the long-term secret keys of all the parties are compromised, the secrecy of previously generated session keys is not affected [BR93a, BWJM97, BWM99b].
- **Weak Perfect Forward Secrecy (wPFS).** The adversary compromises the long-term keys of parties if the target session is executed, Weak-PFS guarantees the secrecy of previously generated session keys, but only for the passive adversary, i.e., the adversary does not modify messages of the session except for eavesdropping the communication [Kra05a].
- **Asymmetric Prefect Forward Secrecy (APFS).** Asymmetric perfect forward secrecy is similar to that of perfect forward secrecy except that only one of the parties to the communication is allowed to be corrupted. This notion is used in the security analysis of `TLS_RSA_PSK` protocol.
- **Unknown Key-Share Attacks (UKS).** Unknown key-share attacks are applicable in a security model taht allows malicious insiders. An unknown key-share attack is an attack whereby party P_i finally believes that he shares the session key with the adversary $\mathcal{A}$, but in fact it is shared with party P_j. As usual, the adversary $\mathcal{A}$ does not get the session key [BWJM97, BWM99b].
- **Key Compromise Impersonation Resilience (KCI).** Suppose that the adversary $\mathcal{A}$ corrupts the long-term secret key of party P_i. In general, $\mathcal{A}$ can impersonate party P_i to other parties. However, for key compromise impersonation resilience $\mathcal{A}$ should not be able to impersonate other parties to P_i [BWJM97, BWM99b].
- **Session-State Expose Resilience (S-S-E).** Suppose that a session state is computed from the static long-term secret keys. This property holds that even if the adversary is allowed access to all the internal state information of a party (P_i) associated to a particular session with its partner party (P_j), it also can not distinguish the session key from a random value [CK02a, Kra05a, LLM07a].
- **Ephemeral-Key Expose Resilience (E-K-E).** Suppose that a party who wants to share information with its partner sends the ephemeral public keys which is generated by the corresponding ephemeral secret keys, e.g. randomness. The ephemeral-key expose resilience holds even if given the ephemeral secret keys used in the target session, an adversary can not break the session key indistinguishability property [CK02a, Kra05a, LLM07a].

- **PKI-Related Attacks (PKI-R-A).** PKI-related attacks in which an adversary can register arbitrary valid public keys (even public keys of honest parties) at the trusted third party, e.g. a certificate authority. CA does not require that the adversary performs *"proof of knowledge"* of the secret key corresponding to a public key when a certificate is established [BWM99b].

3.2 Overview of the Bellare-Rogaway and Canetti-Krawczyk Models

In this section, we give an informal overview of the Bellare-Rogaway (BR93) and (extended) Canetti-Krawczyk Models for authenticated key exchange protocols. Intuitively speaking, the main security goals for AKE protocols are that session keys cannot be distinguished from random values by an outside malicious adversary, and that a party really generates a session key with its intended partner, and not with some other user, i.e., entity authentication and key exchange. In order to formally analyze these key exchange protocols we need a security model. Therefore, some models are well-studied in the cryptographic literature. We mention here just the Bellare-Rogaway and (extended) Canetti-Krawczyk models that are most relevant to this thesis and give informal overview of these models.

3.2.1 Overview of the Bellare-Rogaway Model

In 1993, Bellare and Rogaway [BR93a] first introduced an indistinguishability-based security model for authenticated key exchange protocols, named as BR93 model, which captures the security of key exchange for active adversary such as mutual-entity authentication and confidentiality of agreed session keys. In this paper, they only consider the entity authentication for the symmetric case, i.e., parties share a secret key. Since the pioneering work of Bellare and Rogaway, many extensions have been made to the definition of secure authenticated key exchange protocol [BR95, BWJM97, BCK98, Sho99a, BPR00, CK01, CK02b, KP05, Kra05a, LLM07a]. In this subsection, we describe the BR93 model and introduce some notation which will be useful later in this thesis.

Suppose that a set of honest parties (as potential protocol participants) in the model, where each honest party has long-term secret. In order to formalize the several sequential and parallel protocol executions, we use the same notion introduced by Bellare and Rogaway in [BR93a] and denote that each party is characterized by a polynomial number of oracles, i.e π_i^s. Oracle π_i^s shows the i-th protocol instantiation of a principal party P_i in a protocol execution. In order to simulate adversarial capabilities, they model the adversary $\mathcal{A}$ as a probabilistic polynomial-time (PPT) Turing machine that controls the communication of the participants using the following queries: Send, Reveal, Corrupt and Test queries. We formally describe these involved queries in Section 3.4.2. More details can be found in Section 3.4.2.

The BR93 models partnership of two oracles via the concept of *matching conversations*. Informally, a conversation is the sequence of messages sent and received by an oracle. If the transcripts of two oracles are pairwise equivalent, they have matching conversations. This notion was firstly introduced by Bellare and Rogaway in Paper [BR93a] and later refined in Paper [JKSS12]. In this thesis, we use the refined definition of [JKSS12]. A formal definition of *matching conversations* is given in Section 3.4.3.

In order to capture the security goals for authenticated key exchange protocols, firstly a "fresh" session must be identified. In original paper, a completed session is "fresh" if this session as well as its matching session (if it exists) is not corrupted (i.e., no session key was revealed by the adversary) and if none of the participating parties is corrupted. It is apparent that the BR93 model does not capture *key compromise impersonation* (KCI) and *forward secrecy* (FS).

A two-party AKE protocol is *correct* if for any two accepting oracles that have matching conversation it holds that both oracles have the same session key. The security definition of the BR93-model includes two security properties: *explicit* entity authentication and session key indistinguishability. The security of explicit entity authentication is based on the notion of *matching conversation*. Informally, there exists no fresh oracle π_i^s such that there is no unique oracle π_j^t such that π_i^s and π_j^t have matching conversations. For key indistinguishability, an adversary should not be able to distinguish a true session key from a random key. In other words, the adversary selects a fresh completed session [1] which is called a test session and makes a Test-query for this test session to the challenger. In response, the challenger selects a random bit $b \in \{0, 1\}$: if $b = 0$, it returns a random string from the session key space to the adversary; Otherwise the challenger returns the session key of the test session. Security of AKE protocol is defined based on a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In this game $\mathcal{A}$ interacts with $\mathcal{C}$. $\mathcal{A}$ is allowed to adaptively perform Send, Reveal and Corrupt queries. Then $\mathcal{A}$ makes a Test-query to a fresh session of its choice. At the end of run the adversary outputs a guess b'. The advantage of the adversary is defined as $\epsilon = |Pr[b = b'] - 1/2|$. We say that an AKE protocol is secure in BR93 model if no adversary has more than a negligible advantage in the AKE experiment.

3.2.2 Overview of the Canetti-Krawczyk Model

Cryptographic models for key agreement protocol were initiated by Bellare and Rogaway in the 1990s. These initial models, the BR93 and BR95 models [BR93a, BR95], did not consider various secret properties, e.g. forward secrecy (FS), key compromise impersonation (KCI), and so forth. Later, many papers extended their models to consider various security attributes. In 2001, Canetti and Krawczyk introduced a new key exchange model in [CK01], today known as the CK-model, where implicit authentication and session state are considered. It involves a wide range of practical attacks and has

[1] We denote that a session is completed by party when it receives the last protocol message from its partner session and computes the session key.

become more popular. In 2005, Krawczyk extended their mode to capture key impersonation (KCI) security and weak forward secrecy, named as $\mathsf{CK}_{\mathsf{HMQV}}$ model [Kra05a]. In this paper, he analyzed an efficient one-round key exchange protocol (HMQV) in random oracle model. In this subsection, we simply give an overview of the CK model.

Given a similar execution environment as described above, supposing a set of honest parties (as potential protocol participants) in the model, where each honest party has long-term public/secret keys. A session π_i^s is defined as a s-th instance of a protocol run at a protocol participant P_i. In the CK model, each session has an associated session state, including the intermediate random values used in computing the session key. The adversary can obtain the secure intermediate values by a RevealState-query. Each session is identified by a (pre-specified) session identifier sID. The unique sID is externally given to the oracles of a session at protocol start-up phase. In practice, one can define the sID as the concatenation of the messages sent and received by the party, e.g. in the $\mathsf{CK}_{\mathsf{HMQV}}$ model. The partnership of two oracles via the concept of *matching sessions*. Two sessions (P_i, P_j, sID_i) and (P_j, P_i, sID_j) are said to be *matching sessions* if $\mathsf{sID}_i = \mathsf{sID}_j$.

An active adversary $\mathcal{A}$ is able to issue Send, Reveal, RevealState, Corrupt and Test queries. In contrast to the BR93 model, the CK-model considers *implicit* AKE security. Informally, the security definition only satisfies session key indistinguishability property. The adversary selects a completed session as its test session and makes a Test-query for this session to the challenger. This query can only be issued to a session that has not been *exposed*. A session is not exposed if it as well as its matching session (if it is exists) is not revealed (i.e. no session key or intermediate secure values were revealed) and if none of the participating parties is corrupted by the adversary. In response, the challenger returns a random string from the session key space or the session key of the test session.

Security is defined based on a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. $\mathcal{A}$ is allowed to adaptively perform Send, Reveal, RevealState and Corrupt queries. Then $\mathcal{A}$ makes a Test-query to a fresh session of its choice. At the end of run the adversary outputs a guess for Test-query. $\mathcal{A}$ wins the game if it can give the correct answer.

Remark 3.2. In the $\mathsf{CK}_{\mathsf{HMQV}}$ model, Krawczyk extended the CK01 mode to capture key impersonation (KCI) security and weak perfect forward secrecy (wPFS)(i.e., passive $\mathcal{A}$ in the test session), i.e., $\mathcal{A}$ has learned the private key of party.

3.2.3 Overview of the extended Canetti-Krawczyk Model

In 2007, LaMacchia et al. also introduced a new model known as eCK for analysis of two-party key exchange protocols [LLM07a]. The eCK model captures exposure of ephemeral keys attack by using ephemeral key reveal query, allowing exposure of ephemeral secret keys of the test session and its matching session (if it is exists). Note that the eCK is not stronger than the $\mathsf{CK}_{\mathsf{HMQV}}$ model. In other words, security in the eCK model does not imply security in the $\mathsf{CK}_{\mathsf{HMQV}}$ model. More details can be found

in [Ust09, Cre09]. Here we simply give an overview of the difference between the CK and the eCK model.

In contrast to the CK01 model, the eCK07 model do not assume the existence of explicit session identifiers. They defined a session identifier as follows: A session identifier sID_i^s of a protocol instance π_i^s is identified by a 4-tuple $(\mathsf{P}_i, \mathsf{P}_j, \mathsf{Role}, \mathsf{T}_i^s)$, where P_i is the executing party, P_j is the other party, $\mathsf{Role} \in \{\mathsf{Init.}, \mathsf{Resp.}\}$ is P_i's role in the protocol execution and T_i^s consists of all messages sent and received by P_i. Two sessions π_i^s, π_j^t are said to be matching if π_i^s with $(\mathsf{P}_i, \mathsf{P}_j, \mathsf{Init.}, \mathsf{T}_i^s)$ matches π_j^t with $(\mathsf{P}_j, \mathsf{P}_i, \mathsf{Resp.}, \mathsf{T}_i^s)$ and vice versa.

An active adversary $\mathcal{A}$ is able to adaptively issue Send, EphemeralKeyReveal, Reveal, Corrupt and Test queries. However, in eCK model EphemeralKeyReveal query reveals the randomness (as ephemeral-key) used in computation of the session key. For example, in ephemeral Diffie-Hellman key exchange (g^x, g^y) the attacker can use EphemeralKeyReveal-query to get the ephemeral secret key x or y. However, note that in contrast to the CK model, the ephemeral-key does not include session state that has been calculated using the long-term secret of parties. In addition, the eCK model defines a new definition of freshness, i.e., the adversary is allowed to perform EphemeralKeyReveal and Corrupt queries on the test session.

3.3 Security Model for Passively-secure Key Exchange Protocol

Section 6 describes our authenticated key exchange (AKE) compilers which transform any passively-secure key exchange (KE) protocol to an AKE protocol against an active adversary who fully controls the communication network. Passively-secure key exchange protocol is the most important building block of our compilers. Thus, we give a formal definition of passively-secure key exchange protocol and a corresponding security model in this section.

3.3.1 Passively-secure Two-Move Key Exchange Protocols

A passively-secure two-party key exchange (KE) protocol is a protocol that enables those two parties to compute a shared secret key in presence of a passive adversary model. Roughly speaking, the attacker is not allowed to insert or change information transmitted over the communication links between participating parties. Passively-secure key exchange protocols have many benefits, such as more effectively design and execute and more efficient.

In the following, we formally provide a technical definition of KE which is more detailed than in most other works. This is solely for the purpose of deriving a technical result on general KE protocols without long-term keys. Namely, we require that every secret keys used to generate the session keys must be chosen freshly in each session.

For simplicity of exposition, we only focus on the practically most important class of two-move key exchange protocols.

A passively-secure key exchange scheme consists of three algorithms which may be called by a party in each session, $\mathsf{KE} = (\mathsf{KE.Setup}, \mathsf{KE.EphemeralKeyGen}, \mathsf{KE.SessionKeyGen})$. Let $\mathcal{M}_{\mathsf{KE}}$ be the message space, $\mathcal{SSK}$ be the session key space, $\mathcal{R}$ be the random value space, and $\mathcal{EPK} \times \mathcal{ESK}$ be the space for ephemeral public/secure key. Let T be the transcript of all messages exchanged in a KE protocol instance (see Figure 3.1).

- $\Pi^{\mathsf{KE}} \leftarrow \mathsf{KE.Setup}(1^{\kappa})$: This probabilistic polynomial time algorithm (PPT) takes as input the security parameter κ and outputs a set of system parameters Π^{KE}, defining the message space $\mathcal{M}_{\mathsf{KE}}$, identity space $\mathcal{IDS}$, session key space $\mathcal{SSK}$, and ephemeral key space $\mathcal{EPK} \times \mathcal{ESK}$.
- $(esk_{\mathsf{PID}}, epk_{\mathsf{PID}}, \mathsf{M}_{\mathsf{out}}^{\mathsf{PID}}) \xleftarrow{\$} \mathsf{KE.EphemeralKeyGen}(\Pi^{\mathsf{KE}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID}}; \omega)$: The probabilistic polynomial time algorithm takes as input the system parameters Π^{KE}, a random value $\omega \in \mathcal{R}$ and message $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID}} \in \mathcal{M}_{\mathsf{KE}}$ and outputs an ephemeral key pair $(esk_{\mathsf{PID}}, epk_{\mathsf{PID}})$, where $esk_{\mathsf{PID}} \in \mathcal{ESK}$ and $epk_{\mathsf{PID}} \in \mathcal{EPK}$ and a message $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID}} \in \mathcal{M}_{\mathsf{KE}}$ that requires to be sent in a protocol move, e.g. including ephemeral public keys. The execution of this algorithm might be determined by the input message ($\mathsf{M}_{\mathsf{in}}^{\mathsf{PID}}$) which could be any information including for example identities of session participants, ephemeral public key or just empty string $\emptyset$. If $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID}} = \emptyset$, for simplicity we may write $(esk_{\mathsf{PID}}, epk_{\mathsf{PID}}) \xleftarrow{\$} \mathsf{KE.EphemeralKeyGen}(\Pi^{\mathsf{KE}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID}}; \omega)$.
- $k \leftarrow \mathsf{KE.SessionKeyGen}(esk_{\mathsf{PID}}, \mathsf{T})$: The session key generator is a deterministic polynomial time algorithm which takes as input esk_{PID} of a session participant PID and transcript T of all messages exchanged in this session, and outputs a session key k.

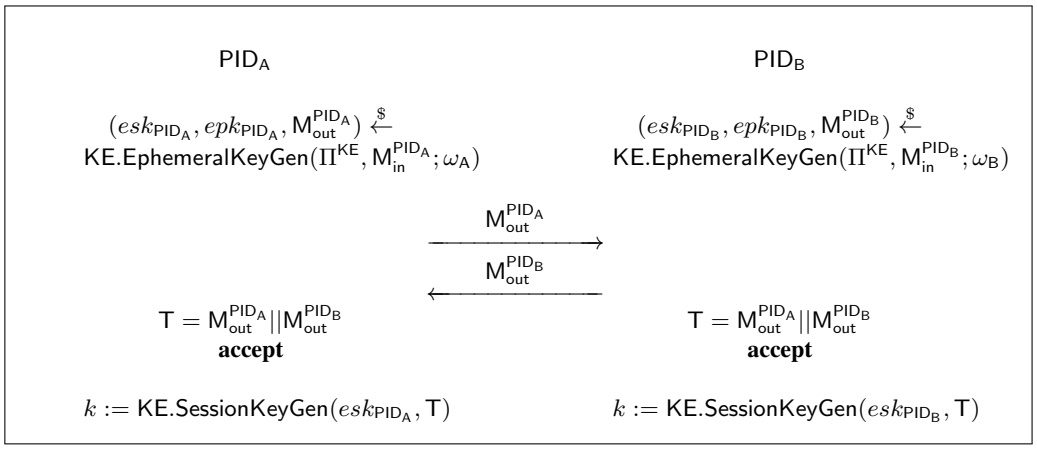


Fig. 3.1. General Two-move KE Protocol

CORRECTNESS. For consistency, we require the correctness properties. For all ($esk_{\mathsf{PID_A}}$, $epk_{\mathsf{PID_A}}$, $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_A}}$) $\xleftarrow{\$}$ $\mathsf{KE.EphemeralKeyGen}(\Pi^{\mathsf{KE}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A}}; \omega_{\mathsf{A}})$, and all ($esk_{\mathsf{PID_B}}$, $epk_{\mathsf{PID_B}}$, $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_B}}$) $\xleftarrow{\$}$ $\mathsf{KE.EphemeralKeyGen}(\Pi^{\mathsf{KE}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID_B}}; \omega_{\mathsf{B}})$, where ($\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_A}}$, $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_B}}$) = T and (ω_{A}, ω_{B}) $\in \mathcal{R}$ we have

$$Pr\left[\mathsf{KE.SessionKeyGen}(esk_{\mathsf{PID_A}}, \mathsf{T}) = \mathsf{KE.SessionKeyGen}(esk_{\mathsf{PID_B}}, \mathsf{T})\right] = 1.$$

We observe that in a passively-secure key exchange protocol where we do not rely on long-term keys it is necessary that the ephemeral values $epk_{\mathsf{PID_A}}$ and $epk_{\mathsf{PID_B}}$ are non-empty. To our knowledge most prominent two-party passively-secure key exchange protocols satisfy our generic definition. In many protocols, such as ephemeral Diffie-Hellman key exchange (EDHKE) protocol [DH76], the algorithm KE.EphemeralKeyGen is executed without any additional message, i.e., $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A}}$ = $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_B}}$ = $\emptyset$, and the generated messages such that $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_A}}$ = $epk_{\mathsf{PID_A}}$ and $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_B}} = epk_{\mathsf{PID_B}}$. However, if only one party $\mathsf{PID}_d \in \{\mathsf{PID_A}, \mathsf{PID_B}\}$ decides on the session key k, ($k = esk_{\mathsf{PID}_d}$, i.e., key transport mechanism, such as RSA-based one-time key exchange protocol), then the shared session key k has to securely be transferred to the other party ($\mathsf{PID}_{\widetilde{d}}$) via some form of encryption of k. In order to guarantee that only the single party $\mathsf{PID}_{\widetilde{d}}$ can decrypt the session key k, the encryptor has to encrypt the session key k using an ephemeral public key of $\mathsf{PID}_{\widetilde{d}}$. As we do not rely on long-term keys, $\mathsf{PID}_{\widetilde{d}}$ has to generate this ephemeral key freshly and send $epk_{\mathsf{PID}_{\widetilde{d}}}$ to PID_d in the first move of the key exchange protocol, resulting in $\mathsf{PID}_{\widetilde{d}} = \mathsf{PID_A}$ and $\mathsf{PID}_d = \mathsf{PID_B}$. In encrypted key transport with freshly chosen key material, in which case we could instantiate those messages in Figure 3.1 as: $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A}} = \emptyset$, $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_B}}$ = $\mathsf{M}_{\mathsf{out}}^{\mathsf{PID_A}}$ = $epk_{\mathsf{PID_A}}$[2].

3.3.2 Security Model

Key indistinguishability (KI) with respect to passive execution environment is the basic security requirement for key exchange security. In this subsection, we give a formal security model for passively-secure key exchange protocols.

EXECUTION ENVIRONMENT. Assume that each party P_i is characterized by a polynomial number of oracles $\{\pi_i^s\}$, where $s \in [d]$, $d \in \mathsf{Poly}(\kappa)$ and $i \in [\ell]$, $\ell \in \mathsf{Poly}(\kappa)$ is the number of parties. An oracle π_i^s represents a process in which the party P_i executes the s-th protocol instance. We define that π_i^s maintains a list of independent internal state variables as described in Table 3.1.

The internal state of each oracle π_i^s is initialized as (Φ_i^s, K_i^s, ESK_i^s, T_i^s) = ($\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$), where $\emptyset$ denotes the empty string. We assume that the session key is assigned to the variable K_i^s such that $\mathsf{K}_i^s \neq \emptyset$ if and only if each oracle completes the execution with an internal state $\Phi_i^s = \texttt{accept}$.

[2] Moreover, we also stress that the key pairs $(esk_{\mathsf{PID_A}}, epk_{\mathsf{PID_A}})$ and $(esk_{\mathsf{PID_B}}, epk_{\mathsf{PID_B}})$ may have distinct forms depending on specific KE protocol, which are also determined by the forms of messages ($\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A}}$, $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_B}}$) while running KE.EphemeralKeyGen.

Variable	Decryption
Φ_i^s	denotes $\Phi_i^s \in \{\texttt{accept}, \texttt{reject}\}$
K_i^s	records the session key $\mathsf{K}_i^s \in \mathcal{K}$
ESK_i^s	records some ephemeral secret values used to compute the session key K_i^s
T_i^s	records all messages sent and received in the order of appearance by oracle π_i^s

Table 3.1. Internal States of Oracles for passive KE Model

ADVERSARY MODEL. In order to model passive attacks in which the adversary is passive during the execution of the target session, we define two queries, Execute, EphemeralKeyReveal queries which model the adversary capability in passive execution environment.

- $(\mathsf{T}, \mathsf{K}) \leftarrow \mathsf{Execute}(\pi_i^s, \pi_j^t)$: An instance of KE protocol is honestly executed between π_i^s and π_j^t. For successful execution, $\Phi_i^s = \Phi_j^t = \texttt{accept}$, $\mathsf{T}_i^s = \mathsf{T}_j^t = \mathsf{T}$ and $\mathsf{K}_i^s = \mathsf{K}_j^t = \mathsf{K}$. We allow the adversary obtain the transcript T exchanged during the honest execution of the protocol and the session key K by this query. The adversary can use Execute-query to perform passive attacks in which the attacker initiates and eavesdrops on honest executions between parties i and j, where $i, j \in [\ell]$ and $s, t \in [d]$.
- $esk_i^s \leftarrow \mathsf{EphemeralKeyReveal}(\pi_i^s)$: Oracle π_i^s responds to this query with the contents of variable ESK_i^s to $\mathcal{A}$. This query models the attacks that loss of ephemeral secret values of a session should not be damaging to other sessions.

SECURITY GAME. The security of a passively-secure key exchange protocol KE = (KE.Setup, KE.EphemeralKeyGen, KE.SessionKeyGen) via the following game that is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. First, the challenger $\mathcal{C}$ generates the public parameters using $\Pi^{\mathsf{KE}} \leftarrow \mathsf{KE.Setup}(1^\kappa)$, and a set of identities $\{\mathsf{PID}_1, \ldots, \mathsf{PID}_\ell\}$ for potential protocol participants where $\ell \in \mathbb{N}$.
2. $\mathcal{A}$ is given the public parameters Π^{KE} and all identities as input and is allowed to interact with $\mathcal{C}$ via making Execute and EphemeralKeyReveal queries. As response, $\mathcal{C}$ returns (T, K) and esk to $\mathcal{A}$.
3. At some point, $\mathcal{A}$ supplies a *fresh* protocol instance (π_i^s, π_j^t) as its test instance (i.e., $\mathcal{A}$ does not issue Execute and EphemeralKeyReveal queries to π_i^s and π_j^t) and sends them to $\mathcal{C}$. Given the test instance (π_i^s, π_j^t), the challenger $\mathcal{C}$ runs a new protocol instance by calling Execute-query, i.e., $(\mathsf{T}, \mathsf{K}_0) \leftarrow \mathsf{Execute}(\pi_i^s, \pi_j^t)$. Then, $\mathcal{C}$ samples $\mathsf{K}_1 \in \mathcal{SSK}$ uniformly at random from the session key space $\mathcal{SSK}$ of the protocol, and tosses a fair coin $b \in \{0, 1\}$. Then $\mathcal{C}$ returns $(\mathsf{T}, \mathsf{K}_b)$ to $\mathcal{A}$.
4. After that $\mathcal{A}$ may continually perform Execute and EphemeralKeyReveal queries. However, $\mathcal{A}$ can not be allowed to make these queries for the test instance. Finally, $\mathcal{A}$ may terminate with returning a bit b' as output.

5. At the end of the security experiment, $\mathcal{A}$ wins if $b' = b$.

We call $Pr\left[b = b'\right]$ the success probability of the adversary $\mathcal{A}$ in winning the above security game. We call $|\Pr\left[b = b'\right] - 1/2|$ the advantage of the adversary $\mathcal{A}$.

Definition 3.3 (Passively-secure Key Exchange). *We say that* KE *is a* $(t_{\mathsf{KE}}, \epsilon_{\mathsf{KE}})$ *passively secure key exchange scheme, if for all probabilistic polynomial-time adversary* $\mathcal{A}$ *running in time* t_{KE} *in the above security game has an advantage of at most* ϵ_{KE}*, i.e.,*

$$\Pr\left[b = b'\right] \leq 1/2 + \epsilon_{\mathsf{KE}},$$

while the number of allowed queries is upper bounded by t_{KE}*.*

3.4 Extended Bellare-Rogaway Model

In this section we describe the (active) security models used for our AKE compilers and tightly-secure AKE protocol. These models can be seen as an extension of the BR93-model, named here as eBR model. We extended the BR93 model since it is unsuitable for security analysis of our protocols. First, the original BR93 model does not consider many practical attacks. Second, the BR93 model allows the adversary to make only a single Test query, but for our tightly-secure AKE protocol we allow the adversary to make multiple adaptive Test queries. Hence, it makes sense to extend the BR93 model for these additional security requirements.

Our model follows the important line of research that was initiated by Bellare and Rogaway [BR93a], and later modified and extended in [BWJM97, Sho99a, BPR00, CK01, Kra05a, LLM07a]. We model security according to two games, one for key indistinguishability, and one for entity authentication based on matching conversation. In order to additionally cover session-state leakage security, we use a similar query "RevealState" as described in Paper [LLM07a] in our model. Moreover, we also model some practical attacks, i.e. PKI-related attacks described in [BWM99c, Oka07, MU09], by using RegCorruptParty query. The adversary can register arbitrary valid public at a certificate authority (CA), and dose not require to perform "proof of knowledge" of the secret key.

In our model, we model an adversary $\mathcal{A}$ by providing an "execution environment", explain how parties behave in a real execution of an AKE scheme with environment and adversary $\mathcal{A}$. In the real world, the environment and the adversary $\mathcal{A}$ can consists of several computers that work in sequential and parallel. In the following, we first describe the execution environment. Then we show how an adversary $\mathcal{A}$ interacts with the challenger $\mathcal{C}$ in the environment, i.e., the power of the adversary. Finally, we give the security definition of authenticated key exchange protocols.

3.4.1 Execution Environment

Let $\mathcal{K}$ be the key space of session keys, and $\{\mathcal{PK}, \mathcal{SK}\}$ be key spaces of long-term public/private keys respectively. Fix a set of honest parties $\{ \mathsf{P}_1, \ldots, \mathsf{P}_\ell \}$ for $\ell \in \mathsf{Poly}(\kappa)$, where each honest party $\mathsf{P}_i \in \{ \mathsf{P}_1, \ldots, \mathsf{P}_\ell \}$ is a potential protocol participant and has a pair of long-term public/private key $(pk_i, sk_i) \in (\mathcal{PK}, \mathcal{SK})$ that corresponds to its identity i.

In order to formalize several sequential and parallel executions of the protocol, each party P_i is characterized by a polynomial number of oracles $\{\pi_i^s\}$, where $s \in [d]$ and $d \in \mathsf{Poly}(\kappa)$. An oracle π_i^s represents a process in which the party P_i executes the s-th protocol instance with access to the long-term key pair (pk_i, sk_i) of party P_i and to all public keys of the other parties. Moreover, we assume that each oracle π_i^s maintains a list of independent internal state variables as described in Table 3.2.

Variable	Decryption
PID_i^s	records the identity $j \in \{1, \ldots, \ell\}$ of intended communication partner P_j
Φ_i^s	denotes $\Phi_i^s \in \{\texttt{accept}, \texttt{reject}\}$
K_i^s	records the session key $\mathsf{K}_i^s \in \mathcal{K}$
$\boxed{\mathsf{State}_i^s}$	records some secret states used to compute the session key K_i^s
T_i^s	records all messages sent and received in the order of appearance by oracle π_i^s

Table 3.2. Internal States of Oracles for extended BR model

Note that for better comparison with BR93 model we will subsequently use **boxes** to highlight state variables that are essentially new in our model.

The internal state of each oracle π_i^s is initialized as $(\mathsf{PID}_i^s, \Phi_i^s, \mathsf{K}_i^s, \mathsf{State}_i^s, \mathsf{T}_i^s) = (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$, where $\emptyset$ denotes the empty string [3]. We assume that the session key is assigned to the variable K_i^s such that $\mathsf{K}_i^s \neq \emptyset$ if and only if each oracle completes the execution with an internal state $\Phi_i^s = \texttt{accept}$.

3.4.2 Adversarial Model

An active adversary $\mathcal{A}$ interacts with the execution environment by issuing the following queries.

Note that for better comparison with BR93 model we will subsequently use **boxes** to highlight state variables that are essentially new in our model.

- $\mathsf{Send}(\pi_i^s, m)$: $\mathcal{A}$ can use this query to send any message m of his own choice to oracle π_i^s. The oracle will respond according to the protocol specification and depending on

[3] Note that in case of one-side authentication AKE protocol we also allow that the space PID contains a special bit-string called $\bowtie$. I.e., if a party does not need to authenticate its intended partner, it can set the variable $\mathsf{PID} = \bowtie$.

its internal state. If m consists of a special symbol $\top$ ($m = \top$), then π_i^s will respond with the first protocol message.

- $\mathsf{Corrupt}(\mathsf{P}_i)$: Oracle π_i^1 responds with the long-term private key sk_i of party P_i. If $\mathsf{Corrupt}(\mathsf{P}_i)$ is the τ_i-th query issued by $\mathcal{A}$, then we say that P_i is τ_i-corrupted. For parties that are not corrupted we define $\tau_i := \infty$.
- $\boxed{\mathsf{RegCorruptParty}(pk_c, \mathsf{P}_c)}$: This query allows $\mathcal{A}$ to register a new party P_c, with a valid public key pk_c on behalf of P_c. If the same party P_c is already registered (either via RegCorruptParty-query or $c \in [\ell]$), a failure symbol $\perp$ is returned to $\mathcal{A}$. Otherwise, P_c is registered, the pair (P_c, pk_c) is distributed to all other parties, and a symbol of success $\triangle$ is returned. This query formalizes a malicious insider setting which can be used to model unknown key share (UKS) attacks and other chosen public key attacks [BWM99c, Oka07, MU09]. We here formalize the arbitrary key registration policy via this query. Parties established by this query are called corrupted or adversarially-controlled. Note that the adversary can register arbitrary valid public keys (even public keys of honest parties) on half of adversary-controlled parties. If parties P_c were adversary-controlled, then the adversary is not allowed to issue Corrupt-queries to P_c.
- $\mathsf{Reveal}(\pi_i^s)$: Oracle π_i^s responds to this query with the contents of variable K_i^s to $\mathcal{A}$. This query models the attacks that loss of a session key should not be damaging to other sessions. Note that we have $\mathsf{K}_i^s \neq \emptyset$ if and only if $\Phi_i^s = \texttt{accept}$.

  If $\mathsf{Reveal}(\pi_i^s)$ is the τ-th query issued by $\mathcal{A}$, π_i^s is called τ-revealed. If $\mathsf{Reveal}(\pi_i^s)$ has never been issued by $\mathcal{A}$, then we say that oracle π_i^s is ∞-revealed.
- $\boxed{\mathsf{RevealState}(\pi_i^s)}$: Oracle π_i^s responds with the contents of the secret state stored in variable State_i^s. Note that for our tightly-secure AKE protocol described in 5.2 (named as tAKE) the variable is always set as $\emptyset$, $\mathsf{State}_i^s := \emptyset$. I.e., we do not allow the adversary to make the query for tightly-secure AKE protocol.

  If $\mathsf{RevealState}(\pi_i^s)$ is the τ-th query issued by $\mathcal{A}$, π_i^s is called τ-state-revealed. If $\mathsf{RevealState}(\pi_i^s)$ has never been issued by $\mathcal{A}$, then we say that oracle π_i^s is ∞-state-revealed.
- $\mathsf{Test}(\pi_i^s)$: Oracle π_i^s handles this query as follows: if the oracle has state $\Phi_i^s \neq \texttt{accept}$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element $k_0 \xleftarrow{\$} \mathcal{K}$, sets $k_1 = \mathsf{K}_i^s$ to the "real" session key, and returns k_b. If $\mathsf{Test}(\pi_i^s)$ is the τ-th query issued by $\mathcal{A}$, we called π_i^s τ-tested. If $\mathsf{Test}(\pi_i^s)$ has never been issued by $\mathcal{A}$, then we say that oracle π_i^s is ∞-tested.

3.4.3 Security Definitions

We model the partnership of two oracles via the concept of *matching conversations* which was first introduced by Bellare and Rogaway [BR93a] and later refined in [JKSS12]. In this thesis, we use the refined definition of [JKSS12].

Let T_i^s denote the transcript of messages sent and received by oracle π_i^s. We assume that messages in a transcript T_i^s are represented as binary strings. Let $|\mathsf{T}_i^s|$ denote the number of the messages in the transcript T_i^s. Assume there are two transcripts T_i^s and T_j^t, where $w := |\mathsf{T}_i^s|$ and $n := |\mathsf{T}_j^t|$. We say that T_i^s is a prefix of T_j^t if $0 < w \leq n$ and the first w messages in transcripts T_i^s and T_j^t are pairwise equivalent as binary strings.

Definition 3.4 (Matching Conversations). *We say that π_i^s has a matching conversation to oracle π_j^t, if*

- *π_i^s has sent the last message(s) and T_j^t is a prefix of T_i^s, or*
- *π_j^t has sent the last message(s) and T_i^s is a prefix of T_j^t.*

We say that two oracles π_i^s and π_j^t have matching conversations if π_i^s has a matching conversation to process π_j^t or vice versa.

Definition 3.5 (Correctness). *We say that a two-party* AKE *protocol, Σ, is* correct *if for any two oracles, π_i^s and π_j^t, that have matching conversations it holds that $\Phi_i^s = \Phi_j^t =$* accept*, $\mathsf{PID}_i^s = j$ and $\mathsf{PID}_j^t = i$ and $\mathsf{K}_i^s = \mathsf{K}_j^t$.*

SECURITY GAME. We formally consider the following security experiment that is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ and that is parametrized by two numbers, ℓ denoted here as the number of honest identities and d as the maximum number of protocol executions per identity.

1. At the beginning of the game, $\mathcal{C}$ generates public parameters Π that are specified by the protocol and $\ell \in \mathsf{Poly}(\kappa)$ long-term key pairs $(sk^{(i)}, pk^{(i)}) \xleftarrow{\$} \mathsf{KeyGen}(1^\kappa)$, $i \in [\ell]$. Then $\mathcal{C}$ implements a collection of oracles oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. It passes to $\mathcal{A}$ all public keys, $pk^{(1)}, \ldots, pk^{(\ell)}$, and the public parameters Π.
2. Then $\mathcal{A}$ may adaptively issue Send, RevealState, Corrupt, Reveal and RegCorruptParty queries [4].
3. At any point during its run $\mathcal{A}$ may choose a (single) accepted oracle π_i^s, i.e., one that already produced the session key, as its test session, and issue $\mathsf{Test}(\pi_i^s)$-query to $\mathcal{C}$. Note that we allow the adversary to make multiple adaptive Test queries for our tightly-secure AKE protocol as described in Section 5.2.1.
4. $\mathcal{C}$ samples a random bit $b \in \{0, 1\}$, returns a test value k_b to $\mathcal{A}$.
5. $\mathcal{A}$ can then continue running Send, RevealState, Corrupt, Reveal and RegCorruptParty queries.
6. At the end of run $\mathcal{A}$ outputs a guess b'. $\mathcal{A}$ wins if $b' = b$.

In this thesis, we prove our generic AKE compilers and tightly-secure AKE protocol in two different security models, $\mathsf{eBR}^{\mathsf{C}}$ and $\mathsf{eBR}^{\mathsf{T}}$. However, there are many similarities

[4] For our tightly-secure AKE protocol, we do not allow $\mathcal{A}$ to supply a RevealState-query.

between both models. Thus, in order to clearly describe the difference between the two, we give separate security definitions as follows.

3.4.3.1 Security Definition for AKE Compilers

In order to capture the security goals for our authenticated key exchange compilers ($\mathsf{eBR}^{\mathsf{C}}$), we need the notion of *freshness*.

Definition 3.6 (Freshness-Rules). *Let π_i^s be an accepting oracle held by a party P_i with intended partner P_j. Meanwhile, let π_j^t be an oracle (if it exists), such that π_i^s and π_j^t have matching conversations. Then the oracle π_i^s is said to be τ-fresh when the adversary $\mathcal{A}$ issues its τ-th query and none of the following conditions holds:*

- *P_i or P_j has been established by the adversary $\mathcal{A}$ via the* RegCorruptParty *query;*
- *P_i is τ_i-corrupted with $\tau_i \leq \tau$ or P_j is τ_j-corrupted with $\tau_j \leq \tau$;*
- *π_i^s is τ_i-state-revealed, where $\tau_i \leq \tau$;*
- *π_i^s is τ_i-revealed, where $\tau_i \leq \tau$;*
- *If there is an oracle, π_j^t, that has matching conversation to π_i^s, then π_j^t is τ_j-state-revealed, where $\tau_j \leq \tau$;*
- *If there is an oracle, π_j^t, that has matching conversation to π_i^s, then π_j^t is τ_j-revealed, where $\tau_j \leq \tau$;*

We define the security of AKE protocols in our strongly extended Bellare-Rogaway model ($\mathsf{eBR}^{\mathsf{C}}$) in two stages: (1) secure *authentication* property using matching conversation[5], and (2) session key *indistinguishability*, i.e., an adversary can not distinguish a fresh session key from a random key except with negligible probability.

Definition 3.7 (Security Definition). *We say that a two-party* AKE *protocol Σ is (t, ϵ)-secure in $\mathsf{eBR}^{\mathsf{C}}$ model , if for all adversaries $\mathcal{A}$ running the above security game within time t, it holds that:*

1. *When $\mathcal{A}$ terminates, there exists no τ-fresh oracle π_i^s (except with probability ϵ), such that*
   - *π_i^s has internal states $\Phi_i^s =$ `accept` and $\mathsf{PID}_i^s = j$, and*
   - *there is no unique oracle π_j^t such that π_i^s and π_j^t have matching conversations.*

   Otherwise, we say that π_i^s accepts maliciously.
2. *When $\mathcal{A}$ returns b' such that*
   - *$\mathcal{A}$ has issued a* Test*-query to oracle π_i^s, and*
   - *the oracle π_i^s is τ-fresh throughout the security game,*

[5] The CK-model captures only implicit authentication.

then the probability that b' equals the bit b sampled by the Test*-query is bounded by*

$$|\Pr[b = b'] - 1/2| \leq \epsilon.$$

Remark 3.8. In contrast to previous works, we explicitly model the revelation of state information of sessions (via RevealState) and strong and practical PKI-based attacks (via RegCorruptParty) like the public key substitution attack (PKS) [BWM99c, MS04] or the duplicate-signature key selection (DSKS) attack [MS04, KM11]. We believe that the revelation of state information is much more realistic than (just) the revelation of keys. To model strong and practical PKI-related attacks we use the RegCorruptParty query into our models that allows attackers to register adversarially chosen public keys and identities. Observe that the adversary does not have to know the corresponding secret key. In practice, most certification authorities (CAs) do not require the registrant to deliver proofs of knowledge of the secret key. Using RegCorruptParty-query the adversary may easily register a public key which has already been registered by another honest party. Since the public keys are equal, all the authenticated messages that are produced by this honest party can be re-used by the adversary. Such attacks can have serious security effects [BWM99c, MS04, KM11]. Our model also formalizes perfect forward secrecy. Forward secrecy is a very strong form of security which guarantees that past sessions remain secure even if the long-term keys get exposed in later sessions. We use a formal definition of forward secrecy that is adopted from [JKSS12].

3.4.3.2 Security Definition for Tightly-secure AKE Protocol

We construct the first Authenticated Key Exchange (AKE) protocol whose security does not degrade with an increasing number of users and sessions. In order to capture the security goals for our AKE protocol with a tight reduction (tAKE), we consider a very strong extended Bellare-Rogaway security model, named $\mathsf{eBR}^{\mathsf{T}}$ model, which allows adaptive corruptions of long-term secrets, adaptive reveals of session keys, and multiple adaptive Test queries. Our Model provides *perfect forward secrecy* (PFS) [BWJM97, Kra05b] and *key-compromise impersonation* (KCI) attacks [JV96, BWM99a, GBN09]. However, an attacker is not allowed to reveal the internal states or intermediate results of computations. In the following we give a formal security definition of our security model, $\mathsf{eBR}^{\mathsf{T}}$.

Definition 3.9 (Freshness-Rules). *Oracle π_i^s is said to be τ-fresh when any adversary $\mathcal{A}$ issues its τ-th query and the following conditions holds:*

- P_i *or* P_j *has not been established by the adversary $\mathcal{A}$ via the* RegCorruptParty *query;*
- *π_i^s has τ_i-accepted, where $\tau_i \leq \tau$.*
- *π_i^s is τ_i-revealed, where $\tau_i > \tau$.*
- *If there is an oracle, π_j^t, that has matching conversation to π_i^s, then π_j^t is ∞-revealed and ∞-tested.*

- *If* $\mathsf{Pid}_i^s = j$ *then* P_j *is* $\tau^{(j)}$*-corrupted with* $\tau^{(j)} > \tau$.

Definition 3.10 (tAKE Security Definition). *We say that a two-party* tAKE *protocol* Σ *is* (t, ϵ)*-secure in* $\mathsf{eBR}^{\mathsf{T}}$ *model, if for all adversaries* $\mathcal{A}$ *running the above security game within time* t*, it holds that:*

1. *When* $\mathcal{A}$ *terminates, there exists no* τ*-fresh oracle* π_i^s *(except with probability* ϵ*), such that*
   - π_i^s *has internal states* $\Omega_i^s = \texttt{accept}$ *and* $\Psi_i^s = j$*, and*
   - *there is no unique oracle* π_j^t *such that* π_i^s *and* π_j^t *have matching conversations.*

   Otherwise, we say that π_i^s *accepts maliciously.*
2. *When* $\mathcal{A}$ *returns* b' *such that*
   - $\mathcal{A}$ *has issued a* Test*-query to oracle* π_i^s*, and*
   - *the oracle* π_i^s *is a* τ*-fresh oracle that is* ∞*-revealed throughout the security game as described above,*

   then the probability that b' *equals the bit* b *sampled by the* Test*-query is bounded by*

$$|\Pr[b = b'] - 1/2| \leq \epsilon.$$

Remark 3.11. Note that for our tAKE protocol that comes with tight security proof under the standard assumptions we allow adaptive corruptions of long-term secrets, adaptive reveals of session keys, and multiple adaptive Test queries.

We allow the adversary to also corrupt P_i before π_i^s accepts, i.e., $\tau^{(i)}$-corrupted with $\tau^{(i)} < \tau$. According to Definition 3.9, the adversary can be allowed to corrupt a τ-fresh oracle π_i^s and its corresponding oracle π_j^t. This allows us to model perfect forward secrecy (PFS) and key-compromise impersonation (KCI) attacks. Moreover, the adversary do not need to supply a proof of knowledge of a matching secret key if it issues a RegCorruptParty-query. It implies that we also model strong and practical PKI-based attacks (via a RegCorruptParty query) [BWM99c, MS04, MS04, KM11]. Finally, we note that the adversary may issue more than one Test-query throughout the security game described above. Simultaneously, it may also issue Reveal-query, even if the test oracle is tested.

3.5 Extended ACCE Security Model

In 2012, Jager et al. introduced a new formal security model for two-party authenticated and confidential channel establishment (ACCE) protocol, and provided security analysis of the cryptographic protocol "TLS_DHE" in this model. A two-party authenticated and confidential channel establishment protocol is a protocol that enables those two parties to compute a shared secret key and provide secure channels. In contrast to the security

of AKE protocols, one requires ciphertext indistinguishability for the security of ACCE protocols.

In this section, we present an extension of the formal security model for two party authenticated and confidential channel establishment protocols introduced by JKSS [JKSS12], namely eACCE model. Since the original ACCE model is unsuitable for the security analysis of TLS-PSK ciphersuites, we require an extended ACCE model. First, the original ACCE model does not consider scenarios with pre-shared, symmetric keys. We extended the ACCE model for hybrid settings. Seconds, the ACCE model dose not address the practical PKI-related attacks. Finally, we require a variant of forward secrecy called *asymmetric perfect forward secrecy*, that captures the protocol sessions of the TLS-PSK protocols. Hence, it makes sense to extend the ACCE model for these additional security requirements.

In this model, while emulating the real-world capabilities of an active adversary, we provide an "execution environment" for adversaries following the tradition of the seminal work of Bellare and Rogaway [BR93a] and its extensions [BWJM97, CK01, Kra05a, LLM07b, JKSS12, KPW13, GKS13, BDK+14].

3.5.1 Execution Environment

In the following let $\ell, d \in \mathsf{Poly}(\kappa)$ be positive integers. In the execution environment, we fix a set of ℓ honest parties $\{\mathsf{P}_1, \ldots, \mathsf{P}_\ell\}$. Each party is either identified by index i in the security experiment or a unique string PID_i with fixed length (which might appear in the protocol flows).

LONG-TERM KEYS. To cover authentication with symmetric keys, we extend the state of each party to also include pre-shared keys. Each party holds (symmetric) pre-shared keys with all other parties. We denote with $\mathsf{PSK}_{i,j} = \mathsf{PSK}_{j,i}$ the symmetric key shared between parties P_i and P_j. Each party P_i with $i \in \{1, \ldots, \ell\}$ also has access to a long-term public/private key pair (pk_i, sk_i). Formally, all parties maintain several state variables as described in Table 3.3.

Variable	Description
sk_i	stores the secret key of a public key pair (pk_i, sk_i)
$\boxed{\mathsf{PSK}_i}$	a vector which contains an entry $\mathsf{PSK}_{i,j}$ per party P_j
τ_i	denotes, that sk_i was corrupted after the τ_i-th query of $\mathcal{A}$
$\boxed{f_i}$	a vector denoting the freshness of all pre-shared keys, containing one entry $f_{i,j} \in \{\mathsf{exposed}, \mathsf{fresh}\}$ for each entry in PSK_i

Table 3.3. Internal States of Parties for extended ACCE model

Note that for better comparison with the original ACCE model we will subsequently use **boxes** to highlight state variables that are essentially new in our model.

The first two variables, sk_i and PSK_i, are used to store keys that are used in the protocol execution while the remaining variables are solely used to define security, see below. (When defining security the latter are additionally managed and updated by the challenger.) The variables of each party P_i will be initialized according to the following rules:

- The long-term key pair $(pk_i, sk_i) \xleftarrow{\$} \mathsf{KeyGen}(1^\kappa)$ and pre-shared key vector PSK_i are chosen randomly from the key space. For all parties $\mathsf{P}_i, \mathsf{P}_j$ with $i, j \in \{1, \ldots, \ell\}$ and with $i \neq j$, and pre-shared keys $\mathsf{PSK}_i \xleftarrow{\$} \{0,1\}^\kappa$ it holds that $\mathsf{PSK}_{i,j} = \mathsf{PSK}_{j,i}$ and $\mathsf{PSK}_{i,i} := \emptyset$.
- All entries in f_i are set to fresh, i.e., $f_i = \emptyset$.
- τ_i is set to $\tau_i := \infty$, which means that all parties are initially not corrupted.

In the following, we will call party P_i uncorrupted iff $\tau_i = \infty$. Thus, we do not consider a dedicated variable that holds the corruption state of the secret key sk_i.

Each honest party P_i can sequentially and concurrently execute the protocol multiple times. This is characterized by a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$. Oracle π_i^s behaves as party P_i carrying out a process to execute the s-th protocol instance with some partner P_j (which is determined during the protocol execution). All oracles of P_i have access to the long-term keys sk_i and PSK_i with $j \in \{1, \ldots, \ell\}$. Moreover, we assume each oracle π_i^s maintains a list of independent internal state variables as described in Table 3.4.

Variable	Description
Φ_i^s	denotes the execution-state $\Phi_i^s \in \{\texttt{negotiating}, \texttt{accept}, \texttt{reject}\}$
PID_i^s	stores the identity of the intended communication partner
ρ_i^s	denotes the role $\rho_i^s \in \{\mathsf{Client}, \mathsf{Server}\}$
$\mathsf{K}_i^s = (k_{\mathsf{enc}}, k_{\mathsf{dec}})$	stores the application keys K_i^s
$\mathsf{St}_i^s = (u, v, st_e, st_d, C)$	stores the current states of the authenticated encryption scheme[6].
T_i^s	records the transcript of messages sent and received by oracle π_i^s
$\boxed{\mathsf{kst}_i^s}$	denotes the freshness $\mathsf{kst}_i^s \in \{\mathsf{exposed}, \mathsf{fresh}\}$ of the session key
b_i^s	stores a bit $b \in \{0,1\}$ used to define security

Table 3.4. Internal States of Oracles for extended ACCE model

Note that for better comparison with the original ACCE model we will subsequently use **boxes** to highlight state variables that are essentially new in our model.

The variables Φ_i^s, PID_i^s, ρ_i^s, K_i^s, st_e, st_d, and T_i^s are used by the oracles to execute the protocol. The remaining variables are only used to define security. As in [JKSS12], , u and v are simple counters used for defining security, st_e and st_d hold the state informa-

tion of the symmetric encryption system. C represents a list of ciphertexts that can be indexed by u and v. The variables of each oracle π_i^s will be initialized with the following rules:

- The execution-state Φ_i^s is set to `negotiating`.
- The variable kst_i^s is set to fresh.
- The bit b_i^s is chosen at random.
- The counters u, v are initialized to 0.
- All other variables are set to only contain the empty string $\emptyset$.

At some point, π_i^s completes the execution with a state $\Phi_i^s \in \{\texttt{accept}, \texttt{reject}\}$. Furthermore, we will always assume (for simplicity) that $\mathsf{K}_i^s = \emptyset$ if an oracle has not reached `accept`-state (yet). To formalize the notion that two oracles engage in an online communication, we use *matching conversations* as proposed by Bellare and Rogaway [BR93a]. We use the variant by JKSS described above in Section 3.4.

To keep our definition of ACCE protocols general we do not consider protocol-specific definitions of partnership like for example [KPW13] who define partnership of TLS sessions using only the first three messages exchanged in the handshake phase (see Remark 3.12 below).

3.5.2 Adversarial Model

An adversary $\mathcal{A}$ in our model is a PPT taking as input the security parameter 1^κ and the public information, which may interact with these oracles by issuing the following queries. For better comparison with the original ACCE model we will subsequently use **boxes** to highlight the queries that are essentially new in our model.

$\mathsf{Send}^{\mathsf{pre}}(\pi_i^s, m)$: This query sends message m to oracle π_i^s. The oracle will respond with the next message m^* (if there is any) that should be sent according to the protocol specification and its internal states.

After answering a $\mathsf{Send}^{\mathsf{pre}}$ query, the variables $(\Phi_i^s, \mathsf{PID}_i^s, \rho_i^s, \mathsf{K}_i^s, T_i^s)$ will be updated depending on the protocol specification. This query is essentially defined as in JKSS.

$\boxed{\mathsf{RegisterParty}(\mu, pk_\mu, [\overrightarrow{\mathsf{PSK}}])}$: This query allows $\mathcal{A}$ to register a new party with a new identity μ and a static public key (pk_μ) to be used for party P_μ. In response, if the same identity μ is already registered (either via a RegisterParty-query or $\mu \in [\ell]$), a failure symbol $\perp$ is returned. Otherwise, a new party P_μ is added with the static public key pk_μ. The secret key sk_μ is set to the empty string $\emptyset$. The parties registered by this query are considered corrupted and controlled by the adversary. If RegisterParty is the τ'-th query of the adversary, P_μ is initialized with $\tau_\mu = \tau'$. If the adversary also provides a pre-shared key PSK, then this key will be implemented for every party P_i with $i \in [\ell]$ as key $\mathsf{PSK}_{i,\mu}$. Otherwise, the simulator chooses a random key

$\mathsf{PSK} \xleftarrow{\$} \{0,1\}^\kappa$ and sets $\mathsf{PSK}_{i,\mu} = \mathsf{PSK}_{\mu,i} := \mathsf{PSK}$ for all parties P_i before outputting PSK. The corresponding entries $f_{i,\mu}$ in the vectors of the other parties P_i with $i \in [\ell]$ are set to exposed. Via this query we extend the ACCE model of JKSS to also model key registration. Note that if parties were adversary-controlled, then the adversary is not allowed to issue Corrupt-queries to these parties.

$\mathsf{RevealKey}(\pi_i^s)$: Oracle π_i^s responds to a RevealKey-query with the contents of variable K_i^s, the application keys. At the same time the challenger sets $\mathsf{kst}_i^s = \mathsf{exposed}$. If at the point when $\mathcal{A}$ issues this query there exists another oracle π_j^t having matching conversation to π_i^s, then we also set $\mathsf{kst}_j^t = \mathsf{exposed}$ for π_j^t. This query slightly deviates from JKSS to cover the attacks mentioned in Remark 3.12 [7].

$\boxed{\mathsf{Corrupt}(\mathsf{P}_i, [\mathsf{P}_j])}$: Depending on the second input parameter, oracle π_i^1 responds with certain long-term secrets of party P_i. This query extends the corruption capabilities of JKSS to symmetric keys.

- If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{P}_i)$ or $\mathsf{Corrupt}(\mathsf{P}_i, \emptyset)$ [8], oracle π_i^1 returns the long-term secret key sk_i of party P_i. If this query is the τ-th query issued by $\mathcal{A}$, then we say that P_i is τ-corrupted and π_i^1 sets $\tau_i := \tau$.
- If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{P}_i, \mathsf{P}_j)$, oracle π_i^1 returns the symmetric pre-shared key $\mathsf{PSK}_{i,j}$ stored in PSK_i and sets $f_{i,j} := \mathsf{exposed}$.
- If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{P}_i, \top)$, oracle π_i^1 returns the vector PSK_i and sets $f_{i,j} := \mathsf{exposed}$ for all entries $f_{i,*} \in f_i$.

$\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \mathsf{len}, \mathsf{Hd})$: This query takes as input two messages m_0 and m_1, length parameter len, and header data Hd. If $\Phi_i^s \neq \texttt{accept}$ then π_i^s returns $\perp$. Otherwise, it proceeds as depicted in Figure 3.2, depending on the random bit $b_i^s \xleftarrow{\$} \{0,1\}$ sampled by π_i^s at the beginning of the game and the internal state variables of π_i^s.

$\mathsf{Decrypt}(\pi_i^s, C, \mathsf{Hd})$: This query takes as input a ciphertext C and header data Hd. If π_i^s has $\Phi_i^s \neq \texttt{accept}$ then π_i^s returns $\perp$. Otherwise, it proceeds as depicted in Figure 3.2. This query is essentially defined as in [JKSS12].

Note that $(u, v, b_i^s, \rho, k_{\mathsf{enc}}^\rho, k_{\mathsf{dec}}^\rho, C)$ denote the values stored in the internal variables of oracle π_i^s.

Remark 3.12. In the execution environment we need the state variable kst_i^s because we have to cope with a theoretical attack caused by the RevealKey query. Consider the situation when an uncorrupted server oracle π_i^s accepts and just sends out the encrypted finished message C_S to its partner oracle π_j^t which up to now has matching conversation (which is defined in 3.4) with π_i^s. However at this point an adversary $\mathcal{A}$ may reveal the session key of π_i^s, drop C_S and computes a ciphertext C_S' which encrypts the same finished message but differs from C_S. Next, $\mathcal{A}$ sends C_S' to π_j^t. The problem is that now π_j^t

[7] JKSS implicitly located the specification of when to set $\mathsf{kst}_j^t = \mathsf{exposed}$ into the security definition.
[8] The party P_i is not adversarially controlled.

$\underline{\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \mathsf{len}, \mathsf{Hd})}$:	$\underline{\mathsf{Decrypt}(\pi_i^s, C, H)}$:
$u := u + 1$	$v := v + 1$
$(C^{(0)}, \mathsf{st}_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k_{\mathsf{enc}}^{\rho}, \mathsf{len}, \mathsf{Hd}, m_0, \mathsf{st}_e)$	If $b_i^s = 0$, then return $\perp$
$(C^{(1)}, \mathsf{st}_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k_{\mathsf{enc}}^{\rho}, \mathsf{len}, \mathsf{Hd}, m_1, \mathsf{st}_e)$	$(m, \mathsf{st}_d) = \mathsf{StE.Dec}(k_{\mathsf{dec}}^{\rho}, \mathsf{Hd}, C, \mathsf{st}_d)$
If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return $\perp$	If $v > u$ or $C \neq C_v$ or $\mathsf{Hd} \neq \mathsf{Hd}_v$,
$(C_u, \mathsf{Hd}_u, \mathsf{st}_e) := (C^{(b)}, \mathsf{Hd}, \mathsf{st}_e^{(b)})$	then $\mathsf{phase} := 1$
Return C_u	If $\mathsf{phase} = 1$ then return m

Fig. 3.2. Encrypt and Decrypt oracles in extended ACCE security experiment.

still accepts, although it does not have matching conversation to π_i^s. To thwart this attack, we modify the RevealKey query and the corresponding security definition as compared to JKSS. In our definition, the challenger in the security game simply keeps track of which session keys have been queried by the adversary via kst_i^s (and kst_j^t) and does not allow the adversary to break the security of any oracle whose session key has been revealed. We find our solution very natural. JKSS use a very similar solution. Roughly speaking, they specify when a session key has been revealed by a partner oracle in the security definition whereas we use the definition of the RevealKey query to do so. We stress that the problem is not restricted to our analysis of TLS-PSK but rather seems fundamental to security protocols in general (similar to the problem that an adversary may always drop the last protocol message which makes one party end up accepting although the transcript of its partner oracle is actually different). We also remark that by using a distinct definition of partnership, we could seemingly avoid this problem, for example by using the definition of [KPW13] (or an adapted form) that only spans the first three messages of TLS. However, this would come at the cost of generality of our definition and we refrain from doing so. Also, we remark that when providing a new partnership definition that is specific to some protocol, for example a truncated version of the transcript, there must be some additional formal evidence that this definition actually uniquely identifies sessions.

Definition 3.13 (Correctness). *We say that an* ACCE *protocol Π is correct, if for any two oracles π_i^s, π_j^t that have matching conversations with $\mathsf{PID}_i^s = j$ and $\mathsf{PID}_j^t = i$ and $\Phi_i^s = \texttt{accept}$ and $\Phi_j^t = \texttt{accept}$ it always holds that $\mathsf{K}_i^s = \mathsf{K}_j^t$.*

3.5.3 Security Definition

We define security via an experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

SECURITY GAME. Assume there is a global variable Λ_{ACCE} which stores the role information of each party for the considered protocol Π.[9] In the game, the following steps are performed:

1. Given the security parameter κ the challenger implements the collection of oracles $\{\pi_i^s : i, j \in [\ell], s \in [d]\}$ with respect to Π, according to the protocol specification. In this process, the challenger generates long-term keys $\mathsf{PSK}_{i,j}$ for all pairs of parties. Next it additionally generates long-term key pairs (pk_i, sk_i) for all parties $i \in [\ell]$ that require them (e.g. if the corresponding party is a server in the `TLS_RSA_PSK` protocol). Finally, the challenger gives the adversary $\mathcal{A}$ all identifiers $\{\mathsf{PID}_i\}$, all public keys $\{pk_i\}$, $i \in [\ell]$, (if any) and Λ_{ACCE} as input.
2. Next the adversary may start issuing $\mathsf{Send}^{\mathsf{pre}}$, RevealKey, Corrupt, Encrypt, Decrypt, and RegisterParty queries.
3. At the end of the game, the adversary outputs a triple (i, s, b') and terminates. $\mathcal{A}$ wins if $b' = b_i^s$.

In the following, we provide a general security definition for ACCE protocols. It will subsequently be referred to when providing specific definitions for ACCE protocols that provide no forward secrecy, perfect forward secrecy or asymmetric perfect forward secrecy. We have tried to keep the details of the execution environment and the definition of security close to that of JKSS. Intuitively, our security definition mainly differs from JKSS in that it considers adversaries that also have access to the new RegisterParty query and the (extended) Corrupt query.

Definition 3.14 (ACCE Security). *We say that an adversary $\mathcal{A}$ $(t, \epsilon_{\mathsf{ACCE}})$-*breaks *an* ACCE *protocol, if $\mathcal{A}$ runs in time t, and at least one of the following two conditions holds:*

1. *When $\mathcal{A}$ terminates, then with probability at least ϵ_{ACCE} there exists an oracle π_i^s such that*
   - *π_i^s accepts with $\mathsf{PID}_i^s = j$ when $\mathcal{A}$ issues its τ_0-th query, and*
   - *both P_i and its intended partner P_j [10] are not corrupted throughout the security game described above and*
   - *π_i^s has internal state $\mathsf{kst}_i^s = \mathsf{fresh}$, and*
   - *there is no unique oracle π_j^t such that π_i^s has a matching conversation to π_j^t.*

   If π_i^s accepts in the above sense, then we say that π_i^s accepts maliciously.
2. *When $\mathcal{A}$ terminates and outputs a triple (i, s, b') such that*

[9] This information is simply used to determine which party also holds asymmetric key pairs besides the shared symmetric keys.

[10] The party P_j is not adversarially corrupted, i.e., $j \in [\ell]$. This means that P_j has not been registered by a RegisterParty query. Otherwise $\mathcal{A}$ may obtain all corresponding secure keys and trivially make oracle π_i^s accept.

- π_i^s *accepts with a unique oracle* π_j^t *such that* π_i^s *has a matching conversation to* π_j^t *when* $\mathcal{A}$ *issues its* τ_0*-th query, and*
- $\mathcal{A}$ *did not issue a* RevealKey*-query to* π_i^s *nor to* π_j^t*, i.e.,* $\mathsf{kst}_i^s = \mathsf{fresh}$*, and*
- P_i *is* τ_i*-corrupted and* P_j *is* τ_j*-corrupted,*

then the probability that b' *equals* b_i^s *is bounded by*

$$|\Pr[b_i^s = b'] - 1/2| \geq \epsilon.$$

If $\mathcal{A}$ *outputs* (i, s, b') *such that* $b' = b_i^s$ *and the above conditions are met, then we say that* $\mathcal{A}$ *answers the encryption-challenge correctly.*

Let us now define security more concretely. We consider three levels of forward secrecy. We start with a basic security definition for protocols that do not provide any form of forward secrecy.

Definition 3.15 (ACCE Security without Forward Secrecy). *We say that an* ACCE *protocol is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{NoFS}})$*-secure* without forward secrecy (NoFS)*, if it is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{NoFS}})$*-secure with respect to Definition 3.14, i.e.,* $\tau_i = \tau_j = \infty$*.*

Definition 3.16 (ACCE Security with Perfect Forward Secrecy). *We say that an* ACCE *protocol is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{PFS}})$*-secure* with perfect forward secrecy (PFS)*, if it is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{PFS}})$*-secure with respect to Definition 3.14 with* $\tau_i, \tau_j > \tau_0$*.*

In the following, we provide our new definition of asymmetric perfect forward secrecy which is similar to that of classical perfect forward secrecy except that only the client is allowed to be corrupted after it has accepted. Server oracles may not be corrupted after accepting (such that for them security holds only in the basic sense of Definition 3.15). We will later show that TLS-RSA-PSK fulfills this extended definition of security.

Definition 3.17 (ACCE Security with Asymmetric Perfect Forward Secrecy). *We say that an* ACCE *protocol is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{APFS}})$*-secure* with asymmetric perfect forward secrecy (APFS)*, if it is* $(t, \epsilon_{\mathsf{ACCE}}^{\mathsf{APFS}})$*-secure with respect to Definition 3.14 and it holds that* $\tau_i = \infty$ *and* $\tau_j > \tau_0$ *if* π_i^s *has internal state* $\rho = \mathsf{Server}$ *or* $\tau_i > \tau_0$ *and* $\tau_j = \infty$ *if* π_i^s *has internal state* $\rho = \mathsf{Client}$*.*

3.6 Relations among the Security Models

In this section, we discuss the exact relation between these security models described above. We classify the differences into two categories: security definition requirements and adversary capabilities. Figure 3.3 lists the terms and abbreviations that are used in Figure 3.4. We summarize the differences in Figure 3.4. The symbol " • " at Table 3.4 means that the model captures this property. Otherwise, "×".

Security Goals							
Abb.	**Description**	**Abb.**	**Description**	**Abb.**	**Description**	**Abb.**	**Description**
e-EA	**e**xplicit **E**ntity **A**uthentication	**i-EA**	**i**mplicit **E**ntity **A**uthentication	**K-I**	**K**ey **I**ndistinguishability	**C-I**	**C**iphertext **I**ndistinguishability

Partnership			
Abb.	**Description**	**Abb.**	**Description**
MC	**M**atching **C**onversition	**sID**	**S**ession **I**dentity

Modeled Attacks							
Abb.	**Description**	**Abb.**	**Description**	**Abb.**	**Description**	**Abb.**	**Description**
PKI-R-A	**PKI R**elated **A**ttack	**PFS**	**P**erfect **F**orward **S**ecrecy	**wPFS**	**w**eak **P**erfect **F**orward **S**ecrecy	**APFS**	**A**symmetric **P**erfect **F**orward **S**ecrecy
SS-E-R	**S**ession **S**tate **E**xpose **R**esilience	**eK-E-R**	**e**phemeral **K**ey **E**xpose **R**esilience	**T-SS-ER**	**T**est **S**ession **S**tate **Expose Resilience**	**T-eK-E-R**	**T**est **e**phemeral **K**ey **Expose Resilience**
KCI	**K**ey **C**ompromise **I**mpersonation	**SK-E**	**S**ession **K**ey **E**xpose	**UKS**	**U**nknown **K**ey **S**hare		

Fig. 3.3. Terms and Abbreviations for Relation among the Security Models

Security / Models	Security Goals				Partnership (Based On)		Modeled Attacks										
	e-EA	**i-EA**	**K-I**	**C-I**	**MC**	**sID**	**PKI-R-A**	**PFS**	**wPFS**	**APFS**	**KCI**	**SK-E**	**UKS**	**SS-E-R**	**eK-E-R**	**T-SS-E-R**	**T-eK-E-R**
BR**93**	•		•	×	•	×	×	×	×	×	×	•	•	×	×	×	×
CK**01**	×	•	•	×	×	•	×	×	×	×	×	•	•	•	×	×	×
CK_{HMQV}	×	•	•	×	×	•	×	×	•	×	•	•	•	•	×	•	×
eCK**07**	×	•	•	×	×	•	×	×	•	×	•	•	•	×	•	×	•
eBR^{C} (AKECompilers)	•	×	•	×	•	×	•	•	•	×	×	•	•	•	×	×	×
eBR^{T} **(Multi-Test)**	•	×	•	×	•	×	•	•	•	×	•	•	•	×	×	×	×
ACCE12	•	×	×	•	•	×	×	•	•	×	•	•	•	×	×	×	×
eACCE Hybird Setting	•	×	×	•	•	×	•	•	•	•	×	•	•	×	×	×	×

Fig. 3.4. Relation among all related Security Models in (active) adversarial Environments

Chapter 4

Security Analysis of TLS-PSK Ciphersuites in the Standard Model

Contents

In this chapter we prove the security of Transport Layer Security Pre-Shared Key ciphersuites (TLS-PSK). TLS-PSK is a set of cryptographic protocols that provide secure communication based on pre-shared keys (PSKs). These pre-shared keys are symmetric keys shared in advance among the communicating parties. There are several ciphersuites: The first set of ciphersuites uses only symmetric key operations for authentication. The second set uses a Diffie-Hellman key exchange authenticated with a pre-shared key. The third set combines public key authentication of the server with pre-shared key authentication of the client. In various environments, TLS-PSK handshake is an interesting alternative for remote authentication between servers and constrained clients like smart cards, for example for mobile phone authentication, EMV-based payment transactions or authentication via electronic ID cards.

The content of this chapter was brought forth in a cooperation with Sven Schäge, Zheng Yang, Jörg Schwenk and Florian Kohlar. The result is published in the proceedings of the *International Conference on Practice and Theory of Public-Key Cryptography (PKC) 2014* [LSY+14b] and in the IACR archive ePrint [LSY+14c]. The authors main contribution within this joined work is double pseudo-random function DPRF and the security analysis.

SUMMARY OF OUR CONTRIBUTIONS.

The work presented in this chapter provides security analysis of all three TLS-PSK ciphersuites. Similar to classical TLS, it is provably impossible to show that the keys

produced by TLS-PSK are indistinguishable from random keys. We describe our contribution as follows:

- **Extended ACCE model for hybrid settings**: As one of our main contributions, we introduce the first definition of ACCE security for authentication protocols with pre-shared keys as described in Section 3.5. We do not propose a separate model but rather an extension of the ACCE model of JKSS as described in [JKSS12] to also cover authentication via pre-shared keys. Our definition is naturally combined to accommodated *hybrid settings* where some types of parties authenticate based on pre-shared keys and others authenticate based on public keys. Note that eACCE model described in Section 3.5 is used for our security analysis of TLS-PSK ciphersuites.
- **Double Pseudo-Random Function**: To prove our results on `TLS_RSA_PSK` and `TLS_DHE_PSK`, we introduce a new variant of pseudo-random functions (PRFs), called *double pseudo-random function* (DPRF). Roughly speaking, a DPRF takes as input two keys only one of which is generated randomly and kept secret from the attacker (as in classical PRFs). However, when the adversary makes its queries, not only the message but also the other key can entirely be specified by the adversary. Our notion of DPRF nicely abstracts the crucial mechanism in TLS-PSK that is required to guarantee (asymmetric) perfect forward secrecy. In our security proofs, we assume that TLS's key derivation function provides a suitable DPRF in the standard model. Existing results on the security of HMAC directly support this assumption for TLS 1.1 when the pre-shared key has a specific bit length.
- **Asymmetric Perfect Forward Secrecy**: we introduce a strengthened variant of forward secrecy definition called *asymmetric perfect forward secrecy* (APFS), that captures that protocol sessions of ACCE protocols with pre-shared keys, see 3.17. APFS guarantees a strong level of confidentiality even if the long-term secrets of the client are exposed after the protocol run. Asymmetric perfect forward secrecy is a strong security notion that can hold for protocols that do not fulfill the standard notion of perfect forward secrecy. This allows us to prove the security of such protocols in a stronger security model as it was previously possible. We show that `TLS_PSK` is ACCE secure (*without forward secrecy*), `TLS_RSA_PSK` is ACCE secure with *asymmetric perfect forward secrecy* and `TLS_DHE_PSK` is secure with (classical) *perfect forward secrecy*. Informally, our results say that TLS-PSK guarantees confidentiality and integrity of all messages exchange between client and server, unless the adversary has learned the pre-shared key or corrupted one of the parties to learn the application/session key. In `TLS_DHE_PSK` the communication remains confidential even if the adversary corrupts the pre-shared secret later on. In contrast, in `TLS_RSA_PSK` the communication remains confidential even if the adversary manages to corrupt the pre-shared key or the server's long-term key later on, but not both of them.

  Note also, that for the `TLS_PSK` and `TLS_DHE_PSK` ciphersuites we neither have to rely on non-standard assumptions like the PRF-ODH assumption nor the random or-

acle model. However, if we want to prove the ACCE security of TLS_RSA_PSK with asymmetric perfect forward secrecy in the *standard model* we need to assume that the public key encryption scheme is IND-CCA secure, similar to [KPW13, KSS13]. We remark that [KPW13] were also able to prove security of the classical TLS ciphersuites based on RSA key transport in the random oracle model. We provide an overview of our results in Figure 4.1. Note that if only the standard definition of pseudo-random function (PRF) is used in the security analysis, we can prove all the ciphersuites in the eACCE model, but without forward secrecy.

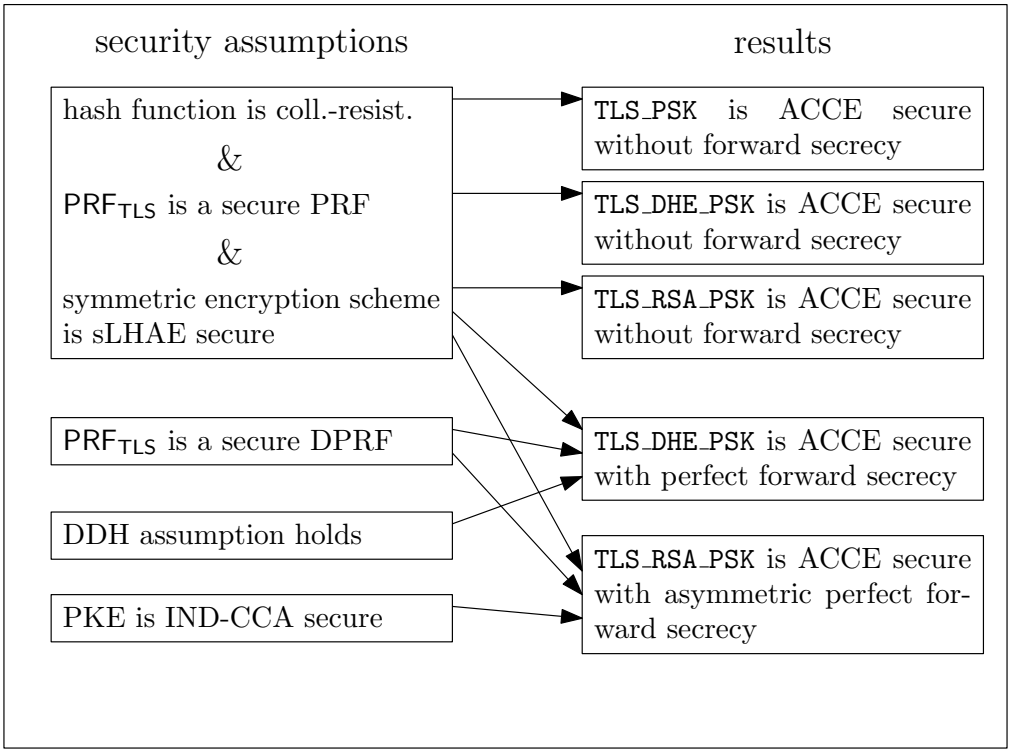


Fig. 4.1. Summary of Results for Security Analysis of TLS-PSK Ciphersuites

ORGANIZATION. First, we introduce all ciphersuite families of TLS-PSK, i.e., TLS-PSK, TLS-DHE-PSK and TLS-RSA-PSK, in Section 4.1. Next, we present a variant of pseudo-random functions (PRFs), called *double pseudo-random function* (DPRF). Finally, we prove that all ciphersuite families of TLS-PSK meet our strong notion of ACCE security in Sections 4.3, 4.4 and 4.5 respectively.

4.1 Transport Layer Security-Pre-shared Key Ciphersuites

In this section we describe a variant of TLS that assumes pre-shared symmetric keys between client and server. The corresponding ciphersuite family is termed TLS with pre-shared keys (TLS-PSK) and available in many TLS releases and libraries [Ope13, MJ, Bot13, Pau13, Pee13, Pet13, Tod13, Bou13].

Related Work On the Security of TLS. Since the introduction of its predecessor SSL, the security of TLS has often been the focus of security researchers and attackers worldwide. Over the time, several attacks on TLS have been published. Most of these attacks do not directly attack the cryptographic core of TLS, but rather exploit side-channels or vulnerabilities in associated technologies, like the famous Bleichenbacher attack [Ble98], or attacks on the domain name system or the public-key infrastructure [KSJG10, DAT12, MS13]. However, despite that no serious attacks on the cryptographic core of the current TLS protocol are known, determining exactly what security guarantees TLS provides has been an elusive problem for many years. This is partly due to the fact that the popular TLS ciphersuites provably do not provide security in the classical sense of authenticated key exchange (AKE) protocols, the classical and very strong standard notion of security of key exchange protocols [MQV95, Sho99b, LMQ$^+$03, JKL04, Ust08]. Until recently only security analyses of modified versions of TLS were published [JK02, GMP$^+$08, MSW10]. At CRYPTO 2012, Jager, et al. [JKSS12] were the first to present a detailed security analysis of the *unmodified* version of one of TLS's ciphersuite families. They showed that the cryptographic core of ephemeral Diffie-Hellman with mutual authentication is a provably secure authenticated and confidential channel establishment (ACCE) protocol in the standard model. ACCE is a new security notion that is particularly well suited to capture what protocols like TLS intuitively want to achieve: the establishment of a secure channel between client and server. Among its features, it not only formalizes confidentiality and integrity of messages exchanged between client and server, but also covers replay and re-ordering attacks. Very recently, Krawczyk, Paterson, and Wee (KPW) [KPW13] and independently Kohlar, Schäge, Schwenk [KSS13] presented, while relying on different cryptographic assumptions and security models [1], extensions of the JKSS result to the remaining ciphersuite families. In particular, they show that TLS-RSA and TLS-DH also constitute ACCE protocols when used for mutual authentication and that TLS-RSA, TLS-DH, and TLS-DHE are ACCE secure in the practically important setting of server-only authentication (for which they provide new formal security definitions). When proving security in the standard model, both KPW and KSS assume that the public key encryption system used for key exchange in TLS-RSA is IND-CCA secure. However, KPW also gave a proof of security in the random oracle model where the public key encryption is only required to be one-way cryptosystem secure under plaintext-checking attacks (OW-PCA).

[1] The security models and complexity assumptions differ mainly with respect to the capabilities granted to the adversary when corrupting and registering new parties and the application of the random oracle model.

Unfortunately, all previous results on the (ACCE) security of TLS are based on either i) new, non-standard security assumption like the PRF-ODH assumption introduced in [JKSS12] and refined in [KPW13, KSS13] or ii) strong idealizations such as modeling TLS's key derivation function as a random oracle [BR93c] or assuming that the public-key encryption scheme in TLS-RSA is substituted with a IND-CCA secure one. Looking somewhat ahead, for the TLS ciphersuites with pre-shared keys, fortunately the situation is different, i.e., security can be based on standard assumptions only.

TLS WITH PRE-SHARED KEYS. The original specifications of the TLS protocol [DA99, DR06, DR08] do not explicitly include ciphersuites that support authentication and key exchange using pre-shared keys. However, since 2005 there exists an extension called "Pre-Shared Key Ciphersuites for Transport Layer Security" (TLS-PSK) which specifically describes such ciphersuites in RFC 4279 [ET05]. (Yet another extension termed "TLS Pre-Shared Key (PSK) Ciphersuites with NULL Encryption" proposes variants of the TLS-PSK ciphersuites that can be used only for authentication, i.e., when channel encryption is disabled [BG07].) The TLS-PSK standard specifies three ciphersuites, `TLS_PSK`, `TLS_RSA_PSK` and `TLS_DHE_PSK`, each of which derives the master secret in a different way. In `TLS_PSK`, the master secret is solely based on the secret pre-shared keys. In the remaining ciphersuites the computation of the master secret is additionally dependent on freshly exchanged secrets via encrypted key transport in `TLS_RSA_PSK` or Diffie-Hellman key exchange in `TLS_DHE_PSK`. The intuition is that as long as either the pre-shared key or the freshly exchanged secret is not compromised, then the Handshake layer yields a secure application key. All three ciphersuites assume that the client only has a pre-shared key for authentication. Although it is not as widespread as TLS with RSA key transport, several interesting and important scenarios for TLS with pre-shared keys exist.

- Since November 2010, the new electronic German ID (eID) card supports online remote authentication of the eID card holder to some online service (eService). The most important network channel involved in the eID online authentication mechanism is a TLS channel, where the eID card shares symmetric keys with the authentication endpoints. Here TLS-PSK is applied to perform mutual authentication between the card holder and some online service. A technical report by the German Federal Office for Information Security (BSI) [fISB05] describes in detail how the pre-shared key known to the eID-Server and the eService is bound to the TLS channel.
- As a second example, we mention the application of TLS-PSK in the Generic Authentication Architecture, the 3GGP mobile phone standard for UMTS and LTE. According to ETSI TR 133 919 V11.0.0 (2012-11), TLS-PSK can be used to secure the communication between server and user equipment (e.g., hand held telephone or laptop with a mobile broadband adapter).

- An IETF draft from 2009 for EMV smart cards describes an authentication protocol based on TLS-PSK [PUM11]. EMV chips are widely deployed and are used commonly for secure payment transactions [CTM12]. The draft describes how the identity information and pre-shared keys stored on the EMV chip can be used to establish a TLS-PSK channel.

The main advantage of TLS-PSK over the standard ciphersuites (with self-signed certificates) is that it avoids computationally expensive public key operations. This particularly pays off in systems with energy-constrained devices like mobile phones or mobile payment stations. Often, in such systems the end points are initialized with pre-shared keys in some secure environment. At the same time, the network layout is hierarchical such that clients technically only directly communicate with a trusted central server that in turn routes their messages to the intended communication partner. This keeps the size of the key material that has to be stored in the clients small and virtually static. Its efficiency makes TLS-PSK a much more attractive alternative in these scenarios than, for example, TLS with self-signed certificates.

4.1.1 A Brief Introduction to TLS-PSK Ciphersuites

This section describes the three sets of ciphersuites specified in TLS-PSK: `TLS_PSK`, `TLS_RSA_PSK` and `TLS_DHE_PSK`. In each of these ciphersuites, the master secret is computed using pre-shared keys which are symmetric keys shared in advance among the communicating parties. The main differences are in the way the master secret is computed. As sketched before, in `TLS_RSA_PSK` the computation of the master secret is additionally dependent on a random value produced by the client that is sent to the server encrypted with its public key. In `TLS_DHE_PSK` the master secret is computed using the pre-shared keys and a fresh Diffie-Hellman key that is exchanged between client and server. The following description is valid for *all* `TLS_PSK` versions.

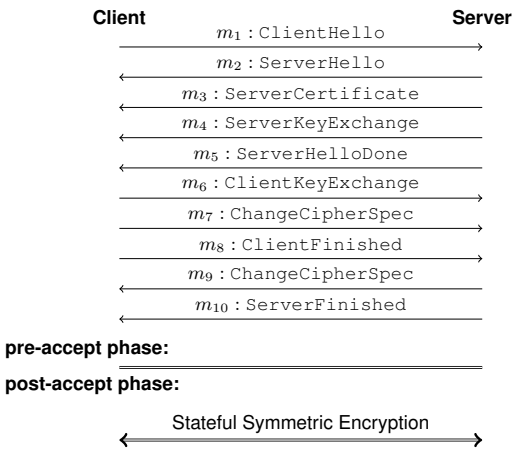


Fig. 4.2. TLS handshake for the PSK key exchange algorithm and associated ciphersuites

The TLS handshake protocol consists of 10 messages, whose content ranges from constant byte values to tuples of cryptographic values. Not all messages are relevant for our security proof, we list them merely for completeness. All messages are prepended with a numeric tag that identifies the type of message, a length value, and the version number of TLS. Also, all messages are sent through the 'TLS Record Layer', which at startup provides no encryption nor any other cryptographic transformations.

CLIENTHELLO. Message m_1 is the `ClientHello` message. In this message, one or more TLS-PSK ciphersuites that are supported by the client are included. For our analysis the only important value is r_{Client}, the random value chosen by the client. It consists of 32 bytes (256 Bits), where 4 Bytes are usually used to encode the local time of the client. The remaining 28 Bytes are chosen randomly by the client. If the client wants to resume a previous TLS session, he may optionally include a TLS *session ID* value received from the server in a previous session. (This value is not protected cryptographically and should thus not be confused with session IDs used in formal security proofs.) This is followed by a list `cs-list` of *ciphersuites*, where each ciphersuite is a tuple of key exchange method, signing, encryption and MAC algorithms, coded as two bytes. Data compression is possible before encryption and is signaled by the inclusion of zero or more compression methods.

SERVERHELLO. The `ServerHello` message m_2 has the same structure as `ClientHello`. The TLS-server can select one of the PSK ciphersuites specified by the client and includes this ciphersuite in the `ServerHello` message. In our analysis the value r_{Server} is important which is drawn randomly by the server.

SERVERCERTIFICATE. For `TLS_PSK` and `TLS_DHE_PSK`, the message is not included. In `TLS_RSA_PSK` $\mathsf{cert}_{\mathsf{Server}}$ contains a public key that is bound to the server's identity.

SERVERKEYEXCHANGE. Since clients and servers may have pre-shared keys with many different parties, in the `ServerKeyExchange` message m_4, the TLS-Server provides a `PSKIdentityHint` pointing to the PSK used for authentication. However, for ephemeral Diffie-Hellman key exchange, the Diffie-Hellman (DH) key exchange parameters are also contained in the `ServerKeyExchange` messages including information on the DH group (e.g. a large prime number p), and the DH share T_{Server} ($T_{\mathsf{Server}} = g^{t_{\mathsf{Server}}}$, where t_{Server} is a random value in $\mathbb{Z}_q$). (We implicitly assume that the client checks whether the received parameters are valid, in particular if T_{Server} is indeed in the group generated by g.)

SERVERHELLODONE. The `ServerHelloDone` message m_5 does not contain any data, but consists only of a constant tag with byte-value '14' and a length value '0'. The server sends this message in order to inform the client to proceed with the next phase of the protocol.

CLIENTKEYEXCHANGE. Message m_6 is called `ClientKeyExchange`. We describe the contents of this message for `TLS_DHE_PSK`, `TLS_PSK` and `TLS_RSA_PSK` separately:

- For `TLS_PSK`, the message is not included.
- For ephemeral Diffie-Hellman key exchange `TLS_DHE_PSK`, it contains the Diffie-Hellman share T_{Client} of the client, i.e., $T_{\mathsf{Client}} = g^{t_{\mathsf{Client}}}$.

- For the RSA-based key exchange `TLS_RSA_PSK`, it uses the public key of the server certificate of the server to authenticate the server, in addition to using a PSK. The client selects a 46-byte random value R and sends a 2-byte version number V and the 46-byte random value R encrypted under the server's RSA public key to the server.

Also, the client send an identifier for the pre-shared key it is going to use when communicating with the server. This information is called `PSK-Identity`.

CHANGECIPHERSPEC. To signal the 'start of encryption' for the server, the client sends message m_7 (`ChangeCipherSpec`) that simply contains the byte value '1' to the server.

CLIENTFINISHED. The next data to be sent is the `ClientFinished` message, m_8, which consists of an encryption C_{Client} of $\mathsf{Fin}_{\mathsf{Client}}$ concatenated with a MAC value. The messages C_{Client} and $\mathsf{Fin}_{\mathsf{Client}}$ are computed as follows:

- $\mathsf{Fin}_{\mathsf{Client}} := \mathsf{PRF}(\mathsf{ms}, \mathsf{label}_3 || \mathsf{H}(m_1 || \ldots || m_7))$;
- $C_{\mathsf{Client}} := \mathsf{StE.Enc}(\mathsf{K}_{\mathsf{ENC}}^{\mathsf{Client}}, \mathsf{len}, \mathsf{Hd}, \mathsf{Fin}_{\mathsf{Client}}, \mathsf{st}_e)$.

The application key $\mathsf{K}_{\mathsf{ENC}}^{\mathsf{Client}}$ and the master secret ms are described below.

CHANGECIPHERSPEC. To signal the 'start of encryption' to the server, message m_9 `ChangeCipherSpec` solely contains byte value '1'.

SERVERFINISHED. After the server has received messages m_8, the server can also compute pms, ms, the encryption and MAC keys, and the `ServerFinished` message $\mathsf{Fin}_{\mathsf{Server}}$. It can then decrypt m_8 and check $\mathsf{Fin}_{\mathsf{Client}}$ by computing the pseudo-random value on the messages sent and received by the server. If this check fails, the server '$rejects$' and aborts the handshake. If the check is successful, it '$accepts$' and sends the message m_9 containing C_{Server} which is the encryption of $\mathsf{Fin}_{\mathsf{Server}}$ to the client. The messages C_{Server} and $\mathsf{Fin}_{\mathsf{Server}}$ are computed as follows:

- $\mathsf{Fin}_{\mathsf{Server}} := \mathsf{PRF}(\mathsf{ms}, \mathsf{label}_4 || \mathsf{H}(m_1 || \ldots || m_9))$;
- $C_{\mathsf{Server}} := \mathsf{StE.Enc}(\mathsf{K}_{\mathsf{ENC}}^{\mathsf{Server}}, \mathsf{len}, \mathsf{Hd}, \mathsf{Fin}_{\mathsf{Server}}, \mathsf{st}_{\mathsf{ENC}})$.

COMPUTING THE PRE-MASTER SECRET. Here, we give a detailed description of pre-master secret pms of all cases (`TLS_PSK`, `TLS_DHE_PSK`, and `TLS_RSA_PSK`) as follws:

- `TLS_PSK` case: For `TLS_PSK` version, the client/server is able to compute the *master secret* ms using the pre-master secret pms, from which all further secret values are derived. If the PSK is N bytes long, the pms consists of the 2-byte representation of the integer value N, N zero bytes, the 2-byte representation of N once again, and the PSK itself,

$$\mathsf{pms} := \mathsf{N}||0...0||\mathsf{N}||\mathsf{PSK}. \tag{4.1}$$

Since the first half of pms is constant for any PSK we get for `TLS_PSK` that the entire security of $\mathsf{PRF}_{\mathsf{TLS}}$ only relies on the second half of pms.

- `TLS_DHE_PSK` case: Let Z be the value produced for DH-based ciphersuites, Z $=g^{t_{\mathsf{Server}}t_{\mathsf{Client}}} = T^{t_{\mathsf{Server}}}_{\mathsf{Client}} = T^{t_{\mathsf{Client}}}_{\mathsf{Server}}$. The pre-master key pms consists of a concatenation of four values: $\mathsf{len}_{\mathsf{Z}}$ indicating the length of Z, Z itself, $\mathsf{len}_{\mathsf{PSK}}$ showing the length of the PSK, and the PSK itself,

$$\mathsf{pms} := \mathsf{len}_{\mathsf{Z}}||\mathsf{Z}||\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}. \tag{4.2}$$

- `TLS_RSA_PSK` case: First, the pre-master secret concatenates the constant $\mathsf{C} = 48$, the 2-byte version number V and a 46-byte random value R, $\mathsf{len}_{\mathsf{PSK}}$ containing the length of the PSK, and the PSK itself,

$$\mathsf{pms} := \mathsf{C}||\mathsf{V}||\mathsf{R}||\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}. \tag{4.3}$$

COMPUTING THE MASTER SECRET. According to the original specification, released as RFC 4279 [ET05], the key derivation function of TLS, denoted here as $\mathsf{PRF}_{\mathsf{TLS}}$, is used when constructing the master secret. $\mathsf{PRF}_{\mathsf{TLS}}$ takes as input a secret, a seed, and an identifying label and produces an output of arbitrary length. We first describe the generic computation of the master secret ms for all ciphersuites using pre-shared keys. The *master secret* ms is computed as follows:

$$\mathsf{ms} := \mathsf{PRF}_{\mathsf{TLS}}(\mathsf{pms}, \mathsf{label}_1||r_{\mathsf{Client}}||r_{\mathsf{Server}}) \tag{4.4}$$

COMPUTING THE APPLICATION KEYS. After computing the *master secret* ms, it is stored for the lifetime of the TLS session, and the pre-master key pms is erased from memory. The *master secret* ms is subsequently used, together with the two random nonces and another fixed label_2, to derive all encryption and MAC keys. The following four application keys (encryption and MAC keys for each direction) are computed by using $\mathsf{PRF}_{\mathsf{TLS}}$, where the inputs are now the master secret ms, label_2 and $r_{\mathsf{Client}}, r_{\mathsf{Server}}$. More precisely, the key material $\mathsf{K}^{\mathsf{Client}}_{\mathsf{ENC}} := (\mathsf{K}^{C\to S}_{\mathsf{enc}}, \mathsf{K}^{C\to S}_{\mathsf{mac}})$ and $\mathsf{K}^{\mathsf{Client}}_{\mathsf{DEC}} := (\mathsf{K}^{S\to C}_{\mathsf{enc}}, \mathsf{K}^{S\to C}_{\mathsf{mac}})$ is computed as

$$\mathsf{K}^{C\to S}_{\mathsf{enc}}||\mathsf{K}^{S\to C}_{\mathsf{enc}}||\mathsf{K}^{C\to S}_{\mathsf{mac}}||\mathsf{K}^{S\to C}_{\mathsf{mac}} := \mathsf{PRF}_{\mathsf{TLS}}(\mathsf{ms}, \mathsf{label}_2||r_{\mathsf{Client}}||r_{\mathsf{Server}}), \tag{4.5}$$

where $\mathsf{K}^{\mathsf{Client}}_{\mathsf{ENC}}$ is used to encrypt and authenticate data sent from the client to the server, and $\mathsf{K}^{\mathsf{Client}}_{\mathsf{DEC}}$ is used to decrypt and verify data received from the server.

4.2 Double Pseudo-Random Functions

To prove our results on TLS_RSA_PSK and TLS_DHE_PSK, we introduce a new variant of pseudo-random functions, called *double pseudo-random function* (DPRF). In this subsection, we give a detailed describtion of DPRF.

Double pseudo-random functions can be thought of as a class of PRFs with two keys. Let $\mathsf{DPRF} : \mathcal{K}_{\mathsf{DPRF}_1} \times \mathcal{K}_{\mathsf{DPRF}_2} \times \mathcal{M}_{\mathsf{DPRF}} \rightarrow \mathcal{R}_{\mathsf{DPRF}}$ denote a family of deterministic functions, where $\mathcal{K}_{\mathsf{DPRF}_1}, \mathcal{K}_{\mathsf{DPRF}_2}$ is the key space, $\mathcal{M}_{\mathsf{DPRF}}$ is the domain and $\mathcal{R}_{\mathsf{DPRF}}$ is the range of PRF.

Intuitively, security requires that the output of the DPRF is indistinguishable from random as long as one key remains hidden from the adversary even if the adversary is able to adaptively specify the second key and the input message. To formalize security we consider the following security game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Let $\mathsf{RF}_{\mathsf{DPRF}}(\cdot,\cdot)$ denote an oracle implemented by $\mathcal{C}$, which takes as input a key $k_j \in \mathcal{K}_{\mathsf{DPRF}_j}$ (where j is specified by the adversary via an Initiation query) and message $m \in \mathcal{M}_{\mathsf{DPRF}}$ and outputs a value $z \in \mathcal{R}_{\mathsf{DPRF}}$.

1. The adversary first runs $\mathsf{Initiation}(j)$ with $j \in \{1,2\}$ to specify the key $k_j \in \mathcal{K}_{\mathsf{DPRF}_j}$ that he wants to manipulate.
2. The challenger $\mathcal{C}$ samples $\hat{b} \xleftarrow{\$} \{0,1\}$, and sets $u = (j \mod 2) + 1$. If $\hat{b} = 0$, the challenger samples $k_u \in_R \mathcal{K}_{\mathsf{DPRF}_u}$ and assigns $\mathsf{RF}_{\mathsf{DPRF}}(\cdot,\cdot)$ to either $\mathsf{DPRF}(\cdot, k_2, \cdot)$ or $\mathsf{DPRF}(k_1, \cdot, \cdot)$ depending on the value of u. For instance, if $u = 2$ then the random function $\mathsf{RF}_{\mathsf{DPRF}}$ is assigned to $\mathsf{DPRF}(\cdot, k_2, \cdot)$, and the $\mathcal{A}$ is allowed to specify k_1 arbitrarily in each query. If $\hat{b} = 1$, the challenger assigns $\mathsf{RF}_{\mathsf{DPRF}}$ to $\mathsf{RF}(\cdot,\cdot)$ which is a truly random function that takes as input key k_j and message m and outputs a value in the same range $\mathcal{R}_{\mathsf{DPRF}}$ as $\mathsf{DPRF}(\cdot,\cdot,\cdot)$.
3. The adversary may adaptively make queries $k_{j,i}$, m_i for $1 \leq i \leq q$ to oracle $\mathsf{RF}_{\mathsf{DPRF}}$ and receives the result of $\mathsf{RF}_{\mathsf{DPRF}}(k_{j,i}, m_i)$, where $k_{j,i}$ denotes the i-th key k_j chosen by $\mathcal{A}$.
4. Finally, the adversary outputs its guess $\hat{b}' \in \{0,1\}$ of $\hat{b}$. If $\hat{b} = \hat{b}'$ the adversary wins.

As before, we let $\Pr\left[\hat{b} = \hat{b}'\right]$ denote the success probability of the adversary and $\left|\Pr\left[\hat{b} = \hat{b}'\right] - 1/2\right|$ its advantage.

Definition 4.1. *We say that* DPRF *is a (q, t, ϵ_{DPRF})-*secure *double pseudo-random function, if any adversary running in probabilistic polynomial time t has at most an advantage of ϵ_{DPRF} to distinguish the* DPRF *from a truly random function, i.e.,*

$$\Pr\left[\hat{b} = \hat{b}'\right] \leq 1/2 + \epsilon_{\mathsf{DPRF}},$$

where the number of allowed queries q is upper bounded by t.

4.2.1 Analysis of the Relationship between DPRF and PRF

To prove (asymmetric) perfect forward secrecy in `TLS_DHE_PSK` and `TLS_RSA_PSK` we assume that $\mathsf{PRF}_{\mathsf{TLS}}$ constitutes a secure DPRF (in the standard model) where the key space of the DPRF consists of the key space of the pre-shared key PSK and the key space of the freshly generated RSA or Diffie-Hellman secret. In this section we analyze the relationship between *double pseudo-random function* (DPRF) defined in Section 4.2 and plain PRF defined in Section 2.2.2, and give a practical construction of a DPRF from PRFs.

Observe that any $(t, \epsilon_{\mathsf{DPRF}})$-secure DPRF trivially gives rise to a plain PRF: if the DPRF is secure after adaptive message queries – even when one of the keys can be specified by the adversary – it remains of course secure when both keys are chosen at random and kept secret from the adversary. However, such a function can be viewed as a PRF where the key space consist of all possible pairs of keys (k_1, k_2). Also, if the DPRF can generate output values which are indistinguishable from random if the adversary adaptively specifies keys and messages, it remains secure if the adversary is only allowed to specify messages. The following lemma holds for any message space and output space.

Lemma 4.2. *Suppose that* $\mathsf{DPRF}(k_1, k_2, m)$ *is a* $(q, t, \epsilon_{\mathsf{DPRF}})$*-secure* DPRF *with key spaces* $\mathcal{K}_{\mathsf{DPRF}_1}$ *and* $\mathcal{K}_{\mathsf{DPRF}_2}$ *according to Definition 4.1. Then* $\mathsf{DPRF}(k_1, k_2, m)$ *is a* $(q, t, \epsilon_{\mathsf{PRF}})$*-secure* PRF *for key space* $\mathcal{K}_{\mathsf{DPRF}_1}$*, key space* $\mathcal{K}_{\mathsf{DPRF}_2}$*, or* $\mathcal{K}_{\mathsf{DPRF}_1} \times \mathcal{K}_{\mathsf{DPRF}_2}$.

Proof. To show that $\mathsf{DPRF}(k_1, k_2, m)$ is a $(t, \epsilon_{\mathsf{PRF}})$-secure PRF for key space $\mathcal{K}_{\mathsf{DPRF}_1}$ (or key space $\mathcal{K}_{\mathsf{DPRF}_2}$) we can imagine a PRF simulator that simply queries $\mathsf{Initiation}(0)$ (or $\mathsf{Initiation}(1)$) to its DPRF challenger and subsequently only relays the message queries and responses. To show that $\mathsf{DPRF}(k_1, k_2, m)$ is a $(t, \epsilon_{\mathsf{PRF}})$-secure PRF for key space $\mathcal{K}_{\mathsf{DPRF}_1} \times \mathcal{K}_{\mathsf{DPRF}_2}$ we use that it is a PRF for $\mathcal{K}_{\mathsf{DPRF}_1}$ and key space $\mathcal{K}_{\mathsf{DPRF}_2}$. So as long as either k_1 or k_2 is chosen uniformly at random $\mathsf{DPRF}(k_1, k_2, m)$ is a PRF even independent of the choice of the other key. However it of course remains a PRF if the other key is chosen uniformly as well. □

4.2.2 A Practical Construction of a DPRF from PRFs.

There is a simple way to construct a DPRF from two PRFs with the same message space. Assume we have two PRFs $\mathsf{PRF}(\cdot,\cdot) : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{M}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ and $\mathsf{PRF}'(\cdot,\cdot) : \mathcal{K}_{\mathsf{PRF}'} \times \mathcal{M}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}'}$. We can then construct a DPRF with $\mathcal{K}_{\mathsf{DPRF}_1} = \mathcal{K}_{\mathsf{PRF}}$ and $\mathcal{K}_{\mathsf{DPRF}_2} = \mathcal{K}_{\mathsf{PRF}'}$ and message space $\mathcal{M}_{\mathsf{DPRF}} = \mathcal{M}_{\mathsf{PRF}}$. On input $k_1 \in \mathcal{K}_{\mathsf{DPRF}_1}$ and $k_2 \in \mathcal{K}_{\mathsf{DPRF}_2}$ and message $m \in \mathcal{M}_{\mathsf{DPRF}}$, the DPRF proceeds as follows:

$$\mathsf{DPRF}(k_1, k_2, m) := \mathsf{PRF}(k_1, m) \oplus \mathsf{PRF}'(k_2, m).$$

Lemma 4.3. *Suppose that* PRF *and* PRF′ *are* $(q, t, \epsilon_{\mathsf{PRF}})$*-secure pseudo-random functions according to Definition 2.4. Then the above* DPRF *is* $(q, t', \epsilon_{\mathsf{DPRF}})$*-secure according to Definition 4.1 with* $t \approx t'$ *and* $\epsilon_{\mathsf{DPRF}} \leq 2\epsilon_{\mathsf{PRF}}$.

Proof. The proof proceeds in a sequence of games, following [Sho04, BR06]. The first game is the real security experiment, assumed there exists an adversary $\mathcal{A}$ that breaks the security of DPRF. Then, we describe several intermediate games that step-wisely modify the original game. Finally we prove that (under the stated security assumptions), no adversary $\mathcal{A}$ can break the security of DPRF.

Let $\mathsf{break}_{\delta}^{(\mathsf{Ind})}$ denote the event that $\hat{b}' = \hat{b}$ in Game δ. Let $\mathsf{Adv}_{\delta} := |\Pr[\mathsf{break}_{\delta}^{(\mathsf{Ind})}] - 1/2|$ denote the advantage of $\mathcal{A}$ in Game δ and $\Pr[\mathsf{break}_{\delta}^{(\mathsf{Ind})}]$ its success probability. Consider the following sequence of games.

GAME 0. This game equals the DPRF security experiment. Thus, for some ϵ_{DPRF} we have

$$\Pr[\mathsf{break}_{0}^{(\mathsf{Ind})}] = 1/2 + \epsilon_{\mathsf{DPRF}}.$$

GAME 1. Assume the adversary queries $\mathsf{Initiation}(j)$ with $j \in 1, 2$. Let $u = (j \mod 2) + 1$. In this game, $\mathcal{C}$ either change the function $\mathsf{PRF}(k_1^*, \cdot)$ to a truly random function $\mathsf{RF}(\cdot)$ (if $u = 1$) or the function $\mathsf{PRF}'(k_2^*, \cdot)$ (if $u = 2$). As before the challenger loses 2 factor for target-PRF-guessing. If there exists a polynomial time adversary $\mathcal{A}$ that can distinguish this game from the previous game, we can construct an algorithm $\mathcal{B}$ using $\mathcal{A}$ that breaks the security of PRF (or PRF′). Exploiting the security of PRF (or PRF′), we have that

$$|\Pr[\mathsf{break}_{0}^{(\mathsf{Ind})}] - \Pr[\mathsf{break}_{1}^{(\mathsf{Ind})}]| \leq 2\epsilon_{\mathsf{PRF}}.$$

Since DPRF's output is computed as

$$\mathsf{DPRF}(k_1, k_2, m) := \mathsf{PRF}(k_1, m) \oplus \mathsf{PRF}'(k_2, m)$$

we also have that even when the adversary entirely and adaptively specifies either $\mathsf{PRF}(k_1, m)$ or $\mathsf{PRF}'(k_2, m)$ via the q queries granted in the DPRF game, the output $\mathsf{DPRF}(k_1, k_2, m)$ is still indistinguishable from random as the other output ($\mathsf{PRF}'(k_2, m)$ or $\mathsf{PRF}(k_1, m)$) remains random. Therefore, if only one pseudo random function (PRF or PRF′) is exchanged with a truly random function, the entire function $\mathsf{DPRF}(k_1, k_2, m)$ behaves like a truly random function. We have

$$\Pr[\mathsf{break}_{1}^{(\mathsf{Ind})}] = 1/2 \Leftrightarrow \mathsf{Adv}_1 = 0.$$

To show that this game is indistinguishable from the previous one observe that $\mathcal{B}$ can simulate the $\mathsf{RF}_{\mathsf{DPRF}}(k_j, \cdot)$ queries made by $\mathcal{A}$ as $\pi_{\mathsf{PRF}} \oplus \mathsf{PRF}(k_j, \cdot)$ where π_{PRF} is the oracle in the PRF security experiment. If $\pi_{\mathsf{PRF}} = \mathsf{RF}(\cdot)$ then $\mathcal{B}$'s simulation yields a

distribution that is equal to this game, otherwise it is equal to the previous game. Finally, $\mathcal{B}$ can simply forward $\mathcal{A}$'s response to its challenger in the PRF game.

Summing up the probabilities from Game 0 to Game 1, we proved Lemma 4.3, i.e.,

$$\epsilon_{\mathsf{DPRF}} \leq 2\epsilon_{\mathsf{PRF}}$$

□

4.2.3 Security Overview of $\mathsf{PRF}_{\mathsf{TLS}}$ in TLS Protocols

In our security proof of TLS_PSK we assume that the pseudo-random function of TLS ($\mathsf{PRF}_{\mathsf{TLS}}$) that is used for the computation of the master-secret constitutes a secure PRF in the standard model when applied with the pre-master key pms as the key. However, to prove perfect forward secrecy in TLS_DHE_PSK and *asymmetric* perfect forward secrecy in TLS_RSA_PSK we assume that $\mathsf{PRF}_{\mathsf{TLS}}$ constitutes a secure DPRF (in the standard model) where the key space of the DPRF consists of the key space of the pre-shared key and the key space of the freshly generated RSA or Diffie-Hellman secret. In the following we will analyze the plausibility of these assumptions in the light of existing results. What considerably complicates our analysis is that TLS 1.1 and TLS 1.2 specify different implementations of $\mathsf{PRF}_{\mathsf{TLS}}$. We therefore start with a detailed description of $\mathsf{PRF}_{\mathsf{TLS}}$ in TLS 1.1 and TLS 1.2.

IMPLEMENTATION OF $\mathsf{PRF}_{\mathsf{TLS}}$ IN TLS 1.1. In TLS 1.1, the output of the key derivation is computed as:

$$\mathsf{PRF}_{\mathsf{TLS}}(\mathsf{pms}, m) = \text{HMAC\_MD5}'(\mathsf{pms}_1, m) \oplus \text{HMAC\_SHA}'(\mathsf{pms}_2, m) \tag{4.6}$$

where pms_1 is the first half of the pre-master secret and pms_2 is the second half, i.e., $\mathsf{pms} = \mathsf{pms}_1 || \mathsf{pms}_2$. As described in detail before, in TLS-PSK the input message m is derived from some constant, public label and the messages exchanged in the protocol so far (depending on the ciphersuite used). HMAC_MD5′ is computed from several concatenations and iterations of HMAC_MD5 that all use the same input key pms_1.

Similarly, HMAC_SHA′ is computed from several concatenations and iterations of HMAC_SHA that again all use the same input key pms_2. In general, the data expansion function HMAC_X′(k,m) for key k and message m is defined as

$$\text{HMAC\_X}'(k, m) = \text{HMAC\_X}(k, A(1)||m) \,||\, \text{HMAC\_X}(k, A(2)||m) \,||\, \ldots, \tag{4.7}$$

where the $A(i)$ are defined as

$$A(0) = m, \tag{4.8}$$
$$A(i) = \text{HMAC\_X}(k, A(i-1)). \tag{4.9}$$

In the above, we use HMAC_X to refer to the standard HMAC algorithm that uses X as the underlying hash function [KBC97]. TLS allows to generate arbitrary output lengths for HMAC_X′ (that are not necessarily multiples of the output length of HMAC_X). To this end, one simply computes an HMAC_X′ output that is (slightly) larger than the target length via Equation 4.7. Next, the last output bits of this result are just discarded (until we meet the target length).

IMPLEMENTATION OF $\mathsf{PRF}_{\mathsf{TLS}}$ IN TLS 1.2. The definition of $\mathsf{PRF}_{\mathsf{TLS}}$ in TLS 1.2 is different from TLS 1.1. First, the new standard allows client and server to negotiate the underlying hash function. However, SHA-256 is used in all pre-defined ciphersuites specified in the TLS standard and generally recommended. Second, the computation of $\mathsf{PRF}_{\mathsf{TLS}}$ no longer relies on two different hash functions but only on a single one. For SHA-256 the function thus simply looks like

$$\mathsf{PRF}_{\mathsf{TLS}}(\mathsf{pms}, m) = \text{HMAC\_SHA-256}'(\mathsf{pms}, m). \tag{4.10}$$

EXISTING RESULTS ON THE SECURITY OF $\mathsf{PRF}_{\mathsf{TLS}}$. Let now us give a brief summary of the most important theoretical results. In [Bel06], Bellare proved that HMAC is a pseudo-random function when the underlying compression function of the hash function is a PRF when keyed by either the data input or the chaining value. In 2008, Foque, Pointcheval, and Zimmer (FPZ) showed that, while relying on [Bel06], for *any* key distribution with high min-entropy, HMAC [DGH+04] is a good strong randomness extractor under security assumptions that are related to the fact that the compression function of the underlying hash function behaves like a pseudo-random function [FPZ08]. In 2010, Fischlin, Lehmann, and Wagner (FLW) [FLW10] specifically analyzed the key derivation function of TLS 1.1. In their analysis, FLW show that HMAC_X′ is a secure PRF if HMAC_X is a secure PRF. FLW rely on [Bel06] (who showed that HMAC_X is pseudo-random) to base the security of $\mathsf{PRF}_{\mathsf{TLS}}$ on the security of the compression functions of the underlying hash functions. Very recently, Koblitz and Menezes gave a separation result showing that the proof of [Bel06] actually does not apply to the standardized version of HMAC [KBC97] without modifications (of the security assumptions). They also present a new proof of security of HMAC as standardized in [Nat02] that holds in the uniform model of complexity. However, due to its large tightness loss, Koblitz and Menezes doubt that even their new and strengthened security proof is "good enough to serve as a convincing real-world guarantee of security of HMAC".

APPLICATION TO $\mathsf{PRF}_{\mathsf{TLS}}$. Unfortunately, none of these results *directly* proves that $\mathsf{PRF}_{\mathsf{TLS}}$ as used in TLS-PSK behaves like a DPRF. Nevertheless, they might in some cases serve as a strong indicator of the security of $\mathsf{PRF}_{\mathsf{TLS}}$.

- When using `TLS_PSK`, the security of $\mathsf{PRF}_{\mathsf{TLS}}$ only relies on the secret key PSK, which is located in the last bits of pms. The results of FLW on the pseudo-

randomness of HMAC_X′ do not only make it appear plausible that $\mathsf{PRF}_{\mathsf{TLS}}$ constitutes a PRF in TLS 1.1 but also in TLS 1.2.[2]

- As another important example, consider the security of $\mathsf{PRF}_{\mathsf{TLS}}$ as specified in TLS 1.1 when using TLS_DHE_PSK and TLS_RSA_PSK. Recall that in TLS_DHE_PSK the pre-master secret is computed as $\mathsf{pms} := \mathsf{len}_{\mathsf{Z}}||\mathsf{Z}||\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}$. Assume that the length of PSK is such that $\mathsf{len}_{\mathsf{Z}}||\mathsf{Z}$ and $\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}$ have equal bit length. When splitting pms in two halves, we now get that $\mathsf{pms}_1 = \mathsf{len}_{\mathsf{Z}}||\mathsf{Z}$ and $\mathsf{pms}_2 = \mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}$ are independent random keys. At this point, we may again rely on the results of FLW (and Lemma 4.2) to deduce that if an adversary may not reveal both, Z and PSK, the output of $\mathsf{PRF}_{\mathsf{TLS}}$ remains indistinguishable from a random value. In this case, $\mathsf{PRF}_{\mathsf{TLS}}$ practically constitutes a DPRF if HMAC_MD5 is a secure PRF for the key space $\mathsf{len}_{\mathsf{Z}}||\mathsf{Z}$ (with random Z) and HMAC_SHA is a secure PRF with key space $\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}$ (for random PSK). We obtain an analogous result for TLS_RSA_PSK if the length of PSK is such that $\mathsf{pms}_1 = \mathsf{C}||\mathsf{V}||\mathsf{R}$ and $\mathsf{pms}_2 = \mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}$.

However, in general PSK may have arbitrary size. In TLS_DHE_PSK for example, if PSK is relatively small it is likely that pms_2 also consists of several bits of Z. In this case, HMAC_SHA′ is used with a key that not only consists of random bits of PSK but also of possibly adversarially manipulated bits of Z. Thus parts of the key bits of HMAC_SHA′ may be specified by the adversary. It is not clear if the results of FLW transfer to these situations as well.

4.3 Security Analysis of TLS-PSK Ciphersuite

In this section we prove that the proposed TLS-PSK ciphersuite (TLS_PSK) as described in 4.1 is secure in the sense of the security guarantees as specified in Section 3.15 in the standard model.

Theorem 4.4. *Let μ be the output length of pseudo-random function of TLS protocol $\mathsf{PRF}_{\mathsf{TLS}}$ and let λ be the length of the nonces r_{Client} and r_{Server}. Assume that $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{PRF}})$-secure pseudo-random function PRF with respect to Definition 2.4 when keyed with the pre-master secret pms described in 4.1 or the master secret ms described in 4.4. Suppose that the hash function is $(t, \epsilon_{\mathsf{CRHF}})$-collision-resistant with respect to Definition 2.2.1.1, and the sLHAE scheme is $(t, \epsilon_{\mathsf{StE}})$-secure with respect to Definition 2.2.7.*

Then for any adversary $\mathcal{A}$ that $(t', \epsilon^{\mathsf{psk}}_{\mathsf{tls}})$-breaks the TLS with pre-shared key protocol (TLS-PSK) in the sense of Definition 3.15 with $t \approx t'$ it holds that

$$\epsilon^{\mathsf{psk}}_{\mathsf{tls}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + 6 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu-1}} + 6 \cdot \epsilon_{\mathsf{StE}} \right).$$

[2] Technically, the key spaces of HMAC_X′ need to be defined distinctly. In TLS 1.2, HMAC_X′ is required to be a PRF for the key space that consist of all $\mathsf{N}||0...0||\mathsf{N}||\mathsf{PSK}$ with random PSK while in TLS 1.1 the key space would consist of all $\mathsf{N}||\mathsf{PSK}$.

We prove this theorem 4.4 in two stages. First, we partition the set of all adversaries into two categories:

1. Adversaries that succeed in making an oracle accept maliciously. We call such an adversary an authentication-adversary $\mathcal{A}^{\mathsf{Auth}}$.
2. Adversaries that do not succeed in making any oracle accept maliciously, but which answer the encryption-challenge. We call such an adversary an encryption-adversary $\mathcal{A}^{\mathsf{Enc}}$.

We show that the TLS-PSK protocol is a secure authenticated and confidential channel establishment (ACCE) protocol except for probability ϵ_{auth}, that is, the protocol fulfills security property 1.) of the ACCE definition 3.15. In the next step, we show that the ciphertext of the ACCE protocol is secure except for probability ϵ_{enc} in the sense of the Property 2.) of the ACCE definition 3.15.

We prove Theorem 4.4 by the following two lemmas. Lemma 4.5 bounds the probability ϵ_{Auth} that an authentication-adversary succeeds, Lemma 4.7 bounds the probability ϵ_{Enc} that an encryption-adversary succeeds. Then we have

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Enc}}.$$

Lemma 4.5. *For any adversary $\mathcal{A}^{\mathsf{Auth}}$ running in time $t' \approx t$, the probability that there exists an oracle π_i^s that accepts maliciously is at most*

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^\lambda} + 2\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^\mu} + 2\epsilon_{\mathsf{StE}} \right),$$

where all quantities are defined as stated in Theorem 4.4.

Note that ϵ_{Auth} is an upper bound on the probability that there exists an oracle that accepts maliciously.

Proof. Let $\mathsf{break}_\delta^{(\mathsf{Auth})}$ be the event that there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has internal state $\Phi_{i^*}^{s^*} = \texttt{accept}$ and $\mathsf{PID}_{i^*}^{s^*} = j^*$, but there is no unique oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations, i.e., it accepts maliciously in the sense of Definition 3.15, in Game δ.

GAME 0. This game equals the ACCE security experiment described in Section 3.15. Thus, we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}.$$

GAME 1. This game is similar to the previous game. However, in this game we add an abort rule. The challenger aborts, if there exists any oracle $\pi_{i^*}^{s^*}$ that chooses a random nonce $r_{i^*}^{s^*}$ or $r_{j^*}^{t^*}$ which is not unique.

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_1^{(\mathsf{Auth})}] + \frac{(d\ell)^2}{2^\lambda}.$$

Note that now each oracle has a unique nonce $r_{i^*}^{s^*}$ or $r_{j^*}^{t^*}$, which is included in the signatures. We will use this to ensure that each oracle that accepts with non-corrupted partner has a *unique* partner oracle.

GAME 2. In this game, the challenger tries to guess which oracle will be the first oracle to accept maliciously and its partner oracle. If its guess is incorrect, then the challenger aborts this game. Technically, this game is identical to previous game, except for the following modifications. The challenger guesses the random indices $((i^*, j^*), (s^*, t^*)) \xleftarrow{\$} [\ell]^2 \times [d]^2$. If there exists an oracle π_i^s that 'accepts' maliciously with intended communication partner PID_j (i.e., oracle π_j^t), and $(i, s) \neq (i^*, s^*)$ and $(j, t) \neq (j^*, t^*)$, then the challenger aborts the game. Note that if π_i^s is the first oracle that 'accepts' maliciously, then with probability $1/(d\ell)^2$ we have $(i, s) = (i^*, s^*)$ and $(j, t) = (j^*, t^*)$, and thus

$$\Pr[\mathsf{break}_1^{(\mathsf{Auth})}] \leq (d\ell)^2 \cdot \Pr[\mathsf{break}_2^{(\mathsf{Auth})}].$$

GAME 3. In this game, we replace the master secret ms^* computed by $\pi_{i^*}^{s^*}$ with an independent random value $\widetilde{\mathsf{ms}^*}$. Moreover, if $\pi_{j^*}^{t^*}$ computes the master key using the same nonces $r_{i^*}^{s^*}||r_{j^*}^{t^*}$ as $\pi_{i^*}^{s^*}$, then we set its master key to $\widetilde{\mathsf{ms}^*}$ as well. We make use of the fact that the pre-shared keys PSK are chosen uniformly at random from the key space of $\mathsf{PRF}_{\mathsf{TLS}}$. Distinguishing Game 3 from Game 2 implies an algorithm breaking the security of the pseudo-random function $\mathsf{PRF}_{\mathsf{TLS}}$, thus

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_3^{(\mathsf{Auth})}] + \epsilon_{\mathsf{PRF}}.$$

GAME 4. In this game, we replace the pseudo-random function PRF used by the test oracle $\pi_{i^*}^{s^*}$ with a truly random function RF. Note that if oracle $\pi_{i^*}^{s^*}$ and oracle $\pi_{j^*}^{t^*}$ use the same master secret $\widetilde{\mathsf{ms}^*}$, then PRF used by $\pi_{j^*}^{t^*}$ is replaced as well. Distinguishing Game 4 from Game 3 implies an algorithm breaking the security of the pseudo-random function $\mathsf{PRF}_{\mathsf{TLS}}$, thus

$$\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_4^{(\mathsf{Auth})}] + \epsilon_{\mathsf{PRF}}.$$

GAME 5. In Game 4 we have replaced the function $\mathsf{PRF}(\widetilde{\mathsf{ms}^*}, \cdot)$ used by $\pi_{i^*}^{s^*}$ with a random function RF. Thus, the `Finished` message is $\mathsf{RF}(\widetilde{\mathsf{ms}^*}, \mathsf{label}_3||\mathsf{CRHF}(m_1||\ldots))$, where $(m_1||\ldots)$ denotes the transcript of all messages sent and received by oracle $\pi_{i^*}^{s^*}$. In this game, the challenger proceeds exactly like the challenger in Game 4, except that we add an abort rule. We abort the game, if oracle $\pi_{i^*}^{s^*}$ ever evaluates the random function RF on an input m^* such that $\mathsf{CRHF}(m^*) = \mathsf{CRHF}(m_1||\ldots)$, where $(m^* \neq m_1||\ldots)$. Since $\mathsf{CRHF}(m^*) = \mathsf{CRHF}(m_1||\ldots)$ implies that a collision for the hash function CRHF is found, we have

$$\Pr[\mathsf{break}_4^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_5^{(\mathsf{Auth})}] + \epsilon_{\mathsf{CRHF}}.$$

GAME 6. Finally we use that the full transcript of all messages sent and received is used to compute the `Finished` messages, and that `Finished` messages are computed by evaluating a truly random function that is only accessible to $\pi_{i^*}^{s^*}$ and (possibly) $\pi_{j^*}^{t^*}$ due to Game 5. This allows to show that any adversary has probability at most $\frac{1}{2^\mu}$ of making oracle $\pi_{i^*}^{s^*}$ accept without having a matching conversation to $\pi_{j^*}^{t^*}$. The `Finished` messages are computed by evaluating a truly random function $\mathsf{RF}(\widetilde{\mathsf{ms}^*}, \cdot)$, which is only accessible to oracles sharing $\widetilde{\mathsf{ms}^*}$, and the full transcript containing all previous messages is used to compute the `Finished` messages. If there is no oracle having a matching conversation to $\pi_{i^*}^{s^*}$, the adversary receives no information about $\mathsf{RF}(\widetilde{\mathsf{ms}^*}, \cdot)$. Therefore we have

$$\Pr[\mathsf{break}_5^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_6^{(\mathsf{Auth})}] + \frac{1}{2^\mu}.$$

GAME 7. In this game we show that any successful adversary can be used to break the sLHAE-security of the encryption system. This step is necessary, as an adversary can violate the matching conversations definition (and thus make an oracle accept maliciously) by creating a new, valid encryption (C_{Client} or C_{Server}) of one of the `Finished` messages ($\mathsf{Fin}_{\mathsf{Client}}$ or $\mathsf{Fin}_{\mathsf{Server}}$), which is distinct from the ciphertext output by the corresponding oracle (client or Server) or of any other messages sent later. Therefore, we need to make sure that the adversary is not able to generate new, valid symmetric encryptions of the `Finished` messages. To this end we exploit the properties of the sLHAE scheme. The forged ciphertexts produced by the adversary are either computed using $\mathsf{K}_{\mathsf{ENC}}^{\mathsf{Client}}$ or using $\mathsf{K}_{\mathsf{ENC}}^{\mathsf{Server}}$. The challenger can guess which of the two keys are used with probability at least $1/2$. On failure, it simply aborts. On success, the challenger can embed the sLHAE challenge into this key.

$$\Pr[\mathsf{break}_6^{(\mathsf{Auth})}] \leq 2\Pr[\mathsf{break}_7^{(\mathsf{Auth})}].$$

Claim 4.6.

$$\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] = \epsilon_{\mathsf{StE}}.$$

Proof. According to the sLHAE security of the symmetric encryption scheme, $\mathcal{A}$ has advantage at most ϵ_{StE} in breaking the sLHAE security. The access to the oracles in the sLHAE security game can directly be used to implement the Encrypt and Decrypt queries of the ACCE security game. Observe that the values generated in this game are exactly distributed as in the previous game. We have $\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] = \epsilon_{\mathsf{StE}}$.

□

Summing up the probabilities from Game 0 to Game 7, we proved Lemma 4.5, i.e.,

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^\lambda} + 2\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^\mu} + 2\epsilon_{\mathsf{StE}} \right).$$

□

Lemma 4.7. *For any adversary $\mathcal{A}^{\mathsf{Enc}}$ running in time $t' \approx t$, the probability that $\mathcal{A}^{\mathsf{Enc}}$ answers the encryption-challenge correctly in the sense of Definition 3.15 is at most $1/2 + \epsilon_{\mathsf{Enc}}$ with*

$$\epsilon_{\mathsf{Enc}} \leq \epsilon_{\mathsf{Auth}} + (d\ell)^2 \left(2\epsilon_{\mathsf{PRF}} + 2\epsilon_{\mathsf{StE}}\right),$$

where ϵ_{Auth} is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 3.15 (cf. Lemma 4.5) and all other quantities are defined as stated in Theorem 4.4.

Proof. Assume that $\mathcal{A}^{\mathsf{Enc}}$ always outputs (i^*, s^*, b') such that all conditions in Property 2 of Definition 3.15 are satisfied. Let $\mathsf{break}_\delta^{(\mathsf{Enc})}$ denote the event that $b' = b_{i^*}^{s^*}$ in Game δ, where $b_{i^*}^{s^*}$ is the random bit sampled by the test oracle $\pi_{i^*}^{s^*}$, and b' is the bit output by $\mathcal{A}^{\mathsf{Enc}}$. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta^{(\mathsf{Enc})}] - 1/2$ denote the advantage of the adversary $\mathcal{A}^{\mathsf{Enc}}$ in Game δ.

GAME 0. This game equals the ACCE security experiment described in Section 3.15. We have

$$\Pr[\mathsf{break}_0^{(\mathsf{Enc})}] = \frac{1}{2} + \epsilon_{\mathsf{Enc}} = 1/2 + \mathsf{Adv}_0.$$

GAME 1. The challenger in this game proceeds as before, but it aborts if the test oracle accepts without a unique partner oracle. In other words, in this game, we make the same modifications as in Game 0 to Game 7 in the proof of Lemma 4.5. Thus we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{Auth}}.$$

We note that at this this point we have now excluded active adversaries between and, moreover, for all τ and any τ-fresh oracle $\pi_{i^*}^{s^*}$ there is a unique oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations. Therefore, any accepting oracle has an uniquely identified partner oracle.

GAME 2. Technically, this game is identical to the previous game, except for the following modifications. The challenger aborts if it fails to guess the oracle $\pi_{i^*}^{s^*}$ (and its partner $\pi_{j^*}^{t^*}$) that the adversary attacks. The probability that the challenger guesses correctly is at least $1/(d\ell)^2$ we have

$$\mathsf{Adv}_1 \leq (d\ell)^2 \cdot \mathsf{Adv}_2.$$

GAME 3. In this game we replace the master secret ms^* computed by $\pi_{i^*}^{s^*}$ with an independent random value $\widetilde{ms^*}$. Moreover, if $\pi_{j^*}^{t^*}$ compute the master key using same nonces $r_{i^*}^{s^*}||r_{j^*}^{t^*}$ as $\pi_{i^*}^{s^*}$, then we set its master key as $\widetilde{ms^*}$. We make use of the fact

that each pre-shared key is chosen uniformly at random from the key space of $\mathsf{PRF}_{\mathsf{TLS}}$. Distinguishing Game 3 from Game 2 implies an algorithm breaking the security of the pseudo-random function $\mathsf{PRF}_{\mathsf{TLS}}$, thus

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{PRF}}.$$

GAME 4. As in Game 4 in the proof of Lemma 4.5, we replace the function $\mathsf{PRF}(\widetilde{ms^*}, \cdot)$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ to compute the application keys with a random function $\mathsf{RF}(\widetilde{ms^*}, \cdot)$. In particular, this function is used to compute the key material as

$$\mathsf{K}_{\mathsf{enc}}^{C\to S}||\mathsf{K}_{\mathsf{enc}}^{S\to C}||\mathsf{K}_{\mathsf{mac}}^{C\to S}||\mathsf{K}_{\mathsf{mac}}^{S\to C} := \mathsf{RF}(\widetilde{ms^*}, \mathsf{label}_2||r_{i^*}^{s^*}||r_{j^*}^{t^*})$$

Distinguishing Game 4 from Game 3 again implies an algorithm breaking the security of the pseudo-random function $\mathsf{PRF}_{\mathsf{TLS}}$, thus we have

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{PRF}}.$$

GAME 5. In this game, we use that the key material $\mathsf{K}_{\mathsf{enc}}^{C\to S}||\mathsf{K}_{\mathsf{enc}}^{S\to C}||\mathsf{K}_{\mathsf{mac}}^{C\to S}||\mathsf{K}_{\mathsf{mac}}^{S\to C}$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ is uniformly random and independent of all TLS Handshake messages exchanged. The challenger can again guess (with probability at least $1/2$) the key that is used to create the ciphertexts which the adversary attacks (either $\mathsf{K}_{\mathsf{enc}}^{\mathsf{Client}}$ or $\mathsf{K}_{\mathsf{enc}}^{\mathsf{Server}}$) and embeds the sLHAE challenge in this key, similar to Game 7 of Lemma 4.5. If there exists a successful ACCE adversary $\mathcal{A}$, then we can construct an algorithm $\mathcal{B}_{\mathsf{sLHAE}}$ that uses $\mathcal{A}$ to break the security of the sLHAE scheme (see Section 2.2.7). Informally, $\mathcal{B}_{\mathsf{sLHAE}}$ using its encryption oracle $\mathcal{O}^{\mathsf{ENC}}$ and decryption oracle $\mathcal{O}^{\mathsf{DEC}}$ interacts with the sLHAE challenger $\mathcal{C}_{\mathsf{sLHAE}}$ described in 2.2.7. Simultaneously, $\mathcal{B}_{\mathsf{sLHAE}}$ acts as an ACCE challenger for $\mathcal{A}$ in this game. $\mathcal{B}_{\mathsf{sLHAE}}$ embeds the sLHAE experiment by simply forwarding all $\mathsf{Encrypt}(\pi_{i^*}^{s^*}, \cdot)$ queries to the encryption oracle $\mathcal{O}^{\mathsf{ENC}}$, and all $\mathsf{Decrypt}(\pi_{j^*}^{t^*}, \cdot)$ queries to the decryption oracle $\mathcal{O}^{\mathsf{DEC}}$. As before we have

$$\mathsf{Adv}_4 = 2 \cdot \mathsf{Adv}_5.$$

Claim 4.8.

$$\Pr[\mathsf{break}_5^{(\mathsf{Auth})}] = \epsilon_{\mathsf{StE}}.$$

Proof. If $\mathcal{A}$ outputs a triple (i^*, s^*, b'), then $\mathcal{B}_{\mathsf{sLHAE}}$ forwards b' to the sLHAE challenger $\mathcal{C}_{\mathsf{sLHAE}}$. Otherwise it selects a random bit $b' \xleftarrow{\$} \{0,1\}$ and outputs b'. $\mathcal{B}_{\mathsf{sLHAE}}$ essentially forwards all messages supplied by $\mathcal{A}$. Therefore, if $\mathcal{A}$ has advantage ϵ in winning the above game against the challenger, then $\mathcal{B}_{\mathsf{sLHAE}}$ can succeed in breaking the sLHAE security with probability at least $1/2 + \epsilon$. Since by assumption in 2.2.7 any polynomial-time attacker has advantage at most ϵ_{StE} in breaking the sLHAE security, we have $\mathsf{Adv}_5 \leq \epsilon_{\mathsf{StE}}$.

□

Summing up the probabilities from Game 0 to Game 5, we proved Lemma 4.7. Finally, Summing up the probabilities from Lemmas 4.5 and 4.7, we obtain

$$\epsilon_{\mathsf{tls}}^{\mathsf{PSK}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + 6 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu-1}} + 6 \cdot \epsilon_{\mathsf{StE}} \right),$$

which proves Theorem 4.4. □

4.4 Security Analysis of TLS-DHE-PSK Ciphersuite

In this section we prove that the proposed TLS-DHE-PSK ciphersuite (`TLS_DHE_PSK`) as described in 4.1 is secure in the sense of the security guarantees as specified in Section 3.16 in the standard model.

Theorem 4.9. *Let μ be the output length of pseudo-random function of TLS protocol $\mathsf{PRF}_{\mathsf{TLS}}$ and let λ be the length of the nonces r_{Client} and r_{Server}. Assume that the key derivation function $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{DPRF}})$-secure DPRF with respect to Definition 4.1 when keyed with the pre-master secret pms described in 4.2 Assume that $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{PRF}})$-secure PRF with respect to Definition 2.4 when keyed with the master secret ms described in 4.4. Suppose that the collision resistant hash function is $(t, \epsilon_{\mathsf{CRHF}})$-secure with respect to Definition 2.2.1.1, the DDH-problem is $(t, \epsilon_{\mathsf{DDH}})$-hard in the group $\mathcal{G}$ used to compute the TLS pre-master secret pms, and that the sLHAE scheme is $(t, \epsilon_{\mathsf{StE}})$-secure with respect to Definition 2.2.7.*

Then for any adversary $\mathcal{A}$ that $(t', \epsilon_{\mathsf{tls}}^{\mathsf{psk-dhe}})$-breaks the TLS-DHE with pre-shared key protocol (TLS-DHE-PSK) in the sense of Definition 3.16 with $t \approx t'$ holds that

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-dhe}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + 3 \cdot \epsilon_{\mathsf{DPRF}} + 3 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu-1}} + \epsilon_{\mathsf{DDH}} + 6 \cdot \epsilon_{\mathsf{StE}} \right).$$

Similar to the previous proof given in Section 4.3, we prove Theorem 4.9 via two lemmas. Lemma 4.10 bounds the probability ϵ_{Auth} that an authentication-adversary $\mathcal{A}^{\mathsf{Auth}}$ succeeds, Lemma 4.12 bounds the probability ϵ_{Enc} that an encryption-adversary $\mathcal{A}^{\mathsf{Enc}}$ succeeds. Then we have

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-dhe}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Enc}}.$$

Lemma 4.10. *For any adversary $\mathcal{A}^{\mathsf{Auth}}$ running in time $t' \approx t$, the probability that there exists an oracle π_i^s that accepts maliciously in the sense of Definition 3.16 is at most*

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu}} + 2\epsilon_{\mathsf{StE}} \right),$$

where all quantities are defined as stated in Theorem 4.9.

Proof. We use the same notions as defined in the proof of Lemma 4.5. The proof proceeds in a *sequence of games* as in [Sho04, BR06].

GAME 0. This game equals the ACCE security experiment described in Section 3.16. Thus, for some ϵ_{Auth} we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}.$$

GAME 1. In this game, the challenger aborts, if there exists any oracle $\pi_{i^*}^{s^*}$ that chooses a random nonce $r_{i^*}^{s^*}$ or $r_{j^*}^{t^*}$ which is not unique, i.e., nonce collision. This game corresponds to Game 1 in the proof of Lemma 4.5. With the same arguments we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})} \leq \Pr[\mathsf{break}_1^{(\mathsf{Auth})}] + \frac{(d\ell)^2}{2^\lambda}.$$

GAME 2. In this game, the challenger tries to guess which oracle will be the oracle to accept maliciously and its partner oracle. This game corresponds to Game 2 in the proof of Lemma 4.5. With the same arguments we have

$$\Pr[\mathsf{break}_1^{(\mathsf{Auth})}] \leq (d\ell)^2 \cdot \Pr[\mathsf{break}_2^{(\mathsf{Auth})}].$$

GAME 3. In this game, we replace the master secret ms^* that is generated by $\pi_{i^*}^{s^*}$ with an independent random value $\widetilde{ms^*}$. Moreover, if $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have computed the same random nonces and the same Diffie-Hellman value Z, we set the master secret of $\pi_{j^*}^{t^*}$ to $\widetilde{ms^*}$ as well. Otherwise we compute the master secret of $\pi_{j^*}^{t^*}$ as specified in the protocol. We exploit that $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{DPRF}})$-secure DPRF by showing that any adversary $\mathcal{A}$ that recognizes our modification can be used to build a successful attacker $\mathcal{B}_{\mathsf{DPRF}}$ against the DPRF properties of $\mathsf{PRF}_{\mathsf{TLS}}$ as described in 4.1. $\mathcal{B}_{\mathsf{DPRF}}$ acts as a challenger for $\mathcal{A}$ in this game.

Suppose that in the DPRF security game the simulator first calls $\mathsf{Initiation}(1)$, indicating that it wants to specify the Diffie-Hellman value (Z) as its target key when making its DPRF queries. Furthermore it will use the queries granted in the DPRF security game to compute the outputs of $\mathsf{PRF}_{\mathsf{TLS}}$. Now, if $\hat{b} = 0$ in the DPRF security game, we are in the previous game of the proof. In case $\hat{b} = 1$ we are in the current game of the proof. Note that according to the security definition described in Section 3.16 $\mathcal{A}$ is allowed to obtain pre-shared key PSK by calling Corrupt-query. So, any adversary $\mathcal{A}$ that distinguishes Game 3 from Game 2 implies a DPRF-adversary $\mathcal{B}_{\mathsf{DPRF}}$ that breaks the DPRF-security of $\mathsf{PRF}_{\mathsf{TLS}}$. We get that

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_3^{(\mathsf{Auth})}] + \epsilon_{\mathsf{DPRF}}.$$

GAME 4. In this game, we replace this pseudo-random function PRF with a truly random function RF. This game corresponds to Game 4 in the proof of Lemma 4.5. With the same arguments we have

$$\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_4^{(\mathsf{Auth})}] + \epsilon_{\mathsf{PRF}}.$$

GAME 5. This game is similar to the previous game except that we add an abortion rule, i.e., we aborts whenever hash-collisions occurs. With the same arguments described in Game 5 we have

$$\Pr[\mathsf{break}_4^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_5^{(\mathsf{Auth})}] + \epsilon_{\mathsf{CRHF}}.$$

GAME 6. This game proceeds as Game 6 in the proof of Lemma 4.5. With same arguments, we have that the adversary cannot successfully compute finished messages except for some negligible probability. We have

$$\Pr[\mathsf{break}_5^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_6^{(\mathsf{Auth})}] + \frac{1}{2^{\mu}}.$$

GAME 7. In this game we will show that if there exists a successful adversary $\mathcal{A}$ breaking the ACCE protocol, we can use it to construct an algorithm to break the sLHAE-security of the encryption system as defined in 2.2.7. This game proceeds as Game 7 in the proof of Lemma 4.5. With the same arguments, we have

$$\Pr[\mathsf{break}_6^{(\mathsf{Auth})}] \leq 2 \cdot \Pr[\mathsf{break}_7^{(\mathsf{Auth})}]$$

Claim 4.11.

$$\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] \leq \epsilon_{\mathsf{StE}}.$$

Proof. With the same arguments as 7, we have $\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] \leq \epsilon_{\mathsf{StE}}$. □

Summing up the probabilities from Game 0 to Game 7, we proved Lemma 4.10, i.e.,

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu}} + 2\epsilon_{\mathsf{StE}} \right).$$

□

Lemma 4.12. *For any adversary $\mathcal{A}^{\mathsf{Enc}}$ running in time $t' \approx t$, the probability that $\mathcal{A}^{\mathsf{Enc}}$ answers the encryption-challenge correctly in the sense of Definition 3.16 is at most $1/2 + \epsilon_{\mathsf{Enc}}$ with*

$$\epsilon_{\mathsf{Enc}} \leq \epsilon_{\mathsf{Auth}} + (d\ell)^2 \left(\epsilon_{\mathsf{DDH}} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + 2\epsilon_{\mathsf{StE}} \right),$$

where ϵ_{Auth} is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 3.16 (cf. Lemma 4.10) and all other quantities are defined as stated in Theorem 4.9.

Note that Definition 3.16 for the TLS-DHE with pre-shared key protocol (TLS-DHE-PSK) captures perfect forward secrecy (PFS).

Proof. We use the same notions as defined in the proof of Lemma 4.7. Consider the following sequence of games.

GAME 0. This game equals the ACCE security experiment described in Section 3.16. For some ϵ_{Enc} we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Enc})}] = \frac{1}{2} + \epsilon_{\mathsf{Enc}} = 1/2 + \mathsf{Adv}_0.$$

GAME 1. The challenger in this game proceeds as before, but it aborts and chooses b' uniformly random, if there exists any oracle that accepts maliciously in the sense of Definition 3.16. Thus we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{Auth}},$$

where ϵ_{Auth} an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 3.16 (cf. Lemma 4.10).

Recall that we assume that $\mathcal{A}^{\mathsf{Enc}}$ always outputs (i^*, s^*, b') such that all conditions in Property 2 of Definition 3.16 are satisfied. In particular it outputs (i^*, s^*, b') such that $\pi_{i^*}^{s^*}$ 'accepts' after the τ_0-th query of $\mathcal{A}^{\mathsf{Enc}}$ with intended partner PID_{j^*}, where PID_{j^*} is τ_{j^*}-corrupted with $\tau_{j^*} > \tau_0$. Note that in Game 1 for any such oracle $\pi_{i^*}^{s^*}$ there exists a unique 'partner oracle' $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ has a matching conversation to $\pi_{j^*}^{t^*}$. Otherwise, this game is aborted.

GAME 2. The challenger in this game proceeds as before, but in addition guesses the indices $((i^*, j^*), (s^*, t^*)) \xleftarrow{\$} [\ell]^2 \times [d]^2$ of the oracle $\pi_{i^*}^{s^*}$ for which the adversary will correctly answer the encryption challenge and its corresponding partner oracle $\pi_{j^*}^{t^*}$. It aborts on failure and proceeds otherwise. Thus,

$$\mathsf{Adv}_1 \leq (d\ell)^2 \cdot \mathsf{Adv}_2.$$

GAME 3. Let $T_{i^*}^{s^*} = g^u$ denote the Diffie-Hellman share chosen by $\pi_{i^*}^{s^*}$, and let $T_{j^*}^{t^*} = g^v$ denote the share chosen by its partner $\pi_{j^*}^{t^*}$. Both oracles compute a shared secret $\mathsf{Z} = (T_{i^*}^{s^*})^v = (T_{j^*}^{t^*})^u = g^{uv}$. Thus, the pre-master secret is computed as $\mathsf{pms}^* = \mathsf{len}_{\mathsf{Z}} \,||\mathsf{Z}||\, \mathsf{len}_{\mathsf{PSK}}||\, \mathsf{PSK}_{i^*,j^*}$. The challenger in this game proceeds as before, but it replaces the Diffie-Hellman shared key Z of $\pi_{i^*}^{s^*}$ and $\pi_{j^*,i^*}^{t^*}$ with a random element $\mathsf{Z}^* = g^w$ for $w \xleftarrow{\$} \mathbb{Z}_q$. We then have that $\widetilde{pms}^* = \mathsf{len}_{\mathsf{Z}} \,||\mathsf{Z}^*||\, \mathsf{len}_{\mathsf{PSK}} \,||\mathsf{PSK}_{i^*,j^*}$. Recall that by assumption the Diffie-Hellman shared key will at no time be revealed by the adversary (in contrast to the pre-shared keys). In order to model perfect forward secrecy property, the adversary is allowed to make Corrupt-query to test oracle $\pi_{i^*}^{s^*}$ after it accepts. [3]

[3] In the previous proof we excluded active adversaries. The result shows that as long as the pre-shared keys remain uncorrupted before the oracles accept, authentication can be guaranteed. In this proof we show that even if the

In this game, we will show that if there exists an adversary $\mathcal{A}$ distinguishing this game from the previous game, we then can construct an algorithm $\mathcal{B}_{\mathsf{DDH}}$ solving the DDH problem as follows.

Assume that $\mathcal{B}_{\mathsf{DDH}}$ receives as input (g, g^u, g^v, g^z) from the DDH challenger. Then, $\mathcal{B}_{\mathsf{DDH}}$ acts as the challenger for $\mathcal{A}_{\mathsf{Enc}}$, and implements the challenger as in Game 2, except that it sets $T_{i^*,s^*} := g^u$, $T_{j^*,t^*} := g^v$, and the pre-master secret of $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ equal to $\mathsf{pms} := \mathsf{len}_{\mathsf{Z}}||g^z||\mathsf{len}_{\mathsf{PSK}}||\mathsf{PSK}_{i^*,j^*}$. Note that $\mathcal{B}_{\mathsf{DDH}}$ can simulate all messages exchanged between $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ properly, in particular the finished messages using knowledge of pms. Since all other oracles are not modified, $\mathcal{B}_{\mathsf{DDH}}$ can simulate these oracles properly as well. If $z = uv$, then the view of $\mathcal{A}$ when interacting with $\mathcal{B}_{\mathsf{DDH}}$ is identical to the previous game, while if $z \xleftarrow{\$} \mathbb{Z}_q$ (i.e., $z \neq uv$) then it is identical to this game. If $\mathcal{A}$ can distinguish this Game from the previous game, $\mathcal{B}_{\mathsf{DDH}}$ can break the DDH problem. Due to the DDH assumption, the advantage of $\mathcal{A}$ in distinguishing between Game 3 and Game 2 is bound by ϵ_{DDH}. Thus, we have

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{DDH}}.$$

GAME 4. In this game we replace the master secret ms^* computed by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ with an independent random value $\widetilde{ms^*}$. Recall that $\pi_{i^*}^{s^*}$ computes the master secret as ms^* = $\mathsf{PRF}_{\mathsf{TLS}}(\mathsf{pms}^*, \mathsf{label}_1 \,||r_{i^*}^{s^*}||r_{j^*}^{t^*})$, where $\mathsf{pms}^* = \mathsf{len}_{\mathsf{Z}^*}\,||\mathsf{Z}^*\,||\mathsf{len}_{\mathsf{PSK}}\,||\mathsf{PSK}_{i^*,j^*}$ is the pre-master key, and Z^* is a random group element of DH group. By security assumption we have that $\mathsf{PRF}_{\mathsf{TLS}}$ constitutes a secure double pseudo random function DPRF when at least one of the values Z^* or PSK_{i^*,j^*} is random and not revealed. If any adversary $\mathcal{A}$ can distinguish this Game from the previous game, we can break the security of DPRF. Due to the security of $\mathsf{PRF}_{\mathsf{TLS}}$, we have

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{DPRF}}.$$

GAME 5. We now replace the function $\mathsf{PRF}_{\mathsf{TLS}}(\widetilde{ms^*}, \cdot)$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ to derive the application keys with a random function $\mathsf{RF}(\widetilde{ms^*}, \cdot)$. Of course the same random function is used for both oracles $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$. In particular, this function is used to compute the key material as

$$\mathsf{K}_{\mathsf{enc}}^{C\to S}||\mathsf{K}_{\mathsf{enc}}^{S\to C}||\mathsf{K}_{\mathsf{mac}}^{C\to S}||\mathsf{K}_{\mathsf{mac}}^{S\to C} := \mathsf{RF}(\widetilde{ms^*}, \mathsf{label}_2||r_{i^*}^{s^*}||r_{j^*}^{t^*}).$$

Distinguishing Game 5 from Game 4 again implies an algorithm breaking the security of the pseudo-random function $\mathsf{PRF}_{\mathsf{TLS}}$, thus we have

$$\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{PRF}}.$$

---

pre-shared keys are revealed after the oracles have accepted, no adversary can break the perfect forward secrecy (PFS) of encryption-challenge.

Note that in Game 5 the key material $\mathsf{K}_{\mathsf{enc}}^{C\to S}||\mathsf{K}_{\mathsf{enc}}^{S\to C}||\mathsf{K}_{\mathsf{mac}}^{C\to S}||\mathsf{K}_{\mathsf{mac}}^{S\to C}$ of oracles $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ is uniformly random and independent of all TLS Handshake messages exchanged in the pre-accept phase.

GAME 6. From Game 5 we know that the key material $\mathsf{K}_{\mathsf{enc}}^{C\to S}||\mathsf{K}_{\mathsf{enc}}^{S\to C}||\mathsf{K}_{\mathsf{mac}}^{C\to S}||\mathsf{K}_{\mathsf{mac}}^{S\to C}$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ in the stateful symmetric encryption scheme is uniformly random and independent of all TLS Handshake messages.

In this game, we construct a simulator $\mathcal{B}_{\mathsf{sLHAE}}$ that uses a successful ACCE attacker $\mathcal{A}$ to break the security of the underlying sLHAE secure symmetric encryption scheme as defined in 2.2.7. This game proceeds as Game 5 in the proof of Lemma 4.7. With the same arguments, we have

$$\Pr[\mathsf{break}_5^{(\mathsf{Enc})}] \leq 2 \cdot \Pr[\mathsf{break}_6^{(\mathsf{Enc})}],$$

Claim 4.13.

$$\Pr[\mathsf{break}_6^{(\mathsf{Enc})}] \leq \epsilon_{\mathsf{StE}}.$$

Proof. With the same arguments as 5, we have $\Pr[\mathsf{break}_6^{(\mathsf{Enc})}] \leq \epsilon_{\mathsf{StE}}$. □

Summing up the probabilities from Game 0 to Game 6, we proved Lemma 4.12. Finally, summing up probabilities from Lemmas 4.10 and 4.12, we obtain

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-dhe}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + 3 \cdot \epsilon_{\mathsf{DPRF}} + 3 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu-1}} + \epsilon_{\mathsf{DDH}} + 6 \cdot \epsilon_{\mathsf{StE}} \right),$$

which proves Theorem 4.9. □

4.5 Security Analysis of TLS-RSA-PSK Ciphersuite

In this section we will analyze the security of TLS-RSA-PSK ciphersuite with the following theorem.

Theorem 4.14. *Let μ be the output length of pseudo-random function of TLS protocol $\mathsf{PRF}_{\mathsf{TLS}}$ and let λ be the length of the nonces r_{Client} and r_{Server}. Assume that the key derivation function $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{DPRF}})$-secure double pseudo-random function with respect to 4.1 when keyed with the pre-master secret pms described in 4.3. Assume that $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{PRF}})$-secure PRF with respect to 2.4 when keyed with the master secret ms described in 4.4. Suppose the collision resistance hash function is $(t, \epsilon_{\mathsf{CRHF}})$-secure with respect to Definition 2.2.1.1, the public key encryption scheme PKE is $(t, \epsilon_{\mathsf{PKE}})$-secure with respect to Definition 2.7. Suppose that the sLHAE scheme is $(t, \epsilon_{\mathsf{StE}})$-secure.*

Then for any adversary $\mathcal{A}$ that $(t', \epsilon_{\mathsf{tls}}^{\mathsf{psk-rsa}})$-breaks TLS-RSA with pre-shared key protocol (TLS-RSA-PSK) (where the key transport mechanism is implemented via PKE) in the sense of Definition 3.17 with $t \approx t'$ it holds that

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-rsa}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + \epsilon_{\mathsf{PKE}} + 3 \cdot \epsilon_{\mathsf{DPRF}} + 3 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{H}} + \frac{1}{2^{\mu-1}} + 6 \cdot \epsilon_{\mathsf{StE}} \right).$$

We show that the TLS-RSA-PSK protocol is a secure ACCE protocol, and give a formal security analysis in ACCE model as described in 3.5. Similar to the previous proofs described in Section 4.3 and in Section 4.4, we also prove Theorem 4.14 via the following lemmas:

- Lemma 4.15 bounds the probability ϵ_{Auth} that an authentication-adversary $\mathcal{A}^{\mathsf{Auth}}$ succeeds;
- Lemma 4.17 bounds the probability ϵ_{Enc} that an encryption-adversary $\mathcal{A}^{\mathsf{Enc}}$ succeeds.

Then we have

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-rsa}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Enc}}.$$

Lemma 4.15. *For any adversary $\mathcal{A}^{\mathsf{Auth}}$ running in time $t' \approx t$, the probability that there exists an oracle π_i^s that accepts maliciously in the sense of Definition 3.17 is at most*

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^{\lambda}} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^{\mu}} + 2\epsilon_{\mathsf{StE}} \right)$$

where all quantities are defined as stated in Theorem 4.14.

The bound on ϵ_{Auth} is derived almost exactly like in the proof of Lemma 4.10, and therefore we only give a sketch of the proof.

Proof. We use the same notions as defined in the proof of Lemma 4.5.

GAME 0. This game equals the ACCE security experiment described in Section 3.17. Thus, for some ϵ_{Auth} we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}.$$

GAME 1. In this game, the challenger aborts, if there exists any oracle $\pi_{i*}^{s^*}$ that chooses a random nonce $r_{i*}^{s^*}$ or $r_{j*}^{t^*}$ which is not unique, i.e., nonce collision. With the same arguments described in Game 1, we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})} \leq \Pr[\mathsf{break}_1^{(\mathsf{Auth})}] + \frac{(d\ell)^2}{2^{\lambda}}.$$

GAME 2. In this game, the challenger tries to guess which oracle will be the oracle to accept maliciously and its partner oracle. Therefore, we have

$$\Pr[\mathsf{break}_1^{(\mathsf{Auth})}] \leq (d\ell)^2 \cdot \Pr[\mathsf{break}_2^{(\mathsf{Auth})}].$$

GAME 3. In this game, we replace the master secret ms^* by an independent random $\widetilde{\mathsf{ms}^*}$. Note that in contrast to TLS-DHE-PSK ciphersuite, RSA-encrypted key transport scheme with freshly chosen key material R is used in TLS-RSA-PSK ciphersuite. This game corresponds to Game 3 in the proof of Lemma 4.10. With the same arguments we have

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_3^{(\mathsf{Auth})}] + \epsilon_{\mathsf{DPRF}}.$$

GAME 4. In this game, we replace this pseudo-random function PRF with a truly random function RF. This game corresponds to Game 4 in the proof of Lemma 4.10. With the same arguments we have

$$\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_4^{(\mathsf{Auth})}] + \epsilon_{\mathsf{PRF}}.$$

GAME 5. This game is similar to the previous game except that we add an abortion rule, i.e., hash-collision. Therefore, we have

$$\Pr[\mathsf{break}_4^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_5^{(\mathsf{Auth})}] + \epsilon_{\mathsf{CRHF}}.$$

GAME 6. This game proceeds as Game 6 in the proof of Lemma 4.10. With same arguments, we have that the adversary cannot successfully compute finished messages except for some negligible probability. We have

$$\Pr[\mathsf{break}_5^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_6^{(\mathsf{Auth})}] + \frac{1}{2^{\mu}}.$$

GAME 7. In this game we will show that if there exists a successful adversary $\mathcal{A}$ breaking the ACCE protocol, we can use it to construct an algorithm to break the sLHAE-security of the encryption system as defined in 2.2.7. With the same arguments as described in Game 7 of Lemma 4.12, we have

$$\Pr[\mathsf{break}_6^{(\mathsf{Auth})}] \leq 2 \cdot \Pr[\mathsf{break}_7^{(\mathsf{Auth})}]$$

Claim 4.16.

$$\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] \leq \epsilon_{\mathsf{StE}}.$$

Proof. With the same arguments as 7, we have $\Pr[\mathsf{break}_7^{(\mathsf{Auth})}] \leq \epsilon_{\mathsf{StE}}$. □

Summing up the probabilities from Game 0 to Game 7, we proved Lemma 4.15, i.e.,

$$\epsilon_{\mathsf{Auth}} \leq (d\ell)^2 \cdot \left(\frac{1}{2^\lambda} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CRHF}} + \frac{1}{2^\mu} + 2\epsilon_{\mathsf{StE}} \right).$$

□

Lemma 4.17. *For any adversary $\mathcal{A}^{\mathsf{Enc}}$ running in time $t' \approx t$, the probability that $\mathcal{A}$ answers the encryption-challenge correctly in the sense of Definition 3.17 is at most $1/2 + \epsilon_{\mathsf{enc}}$ with*

$$\epsilon_{\mathsf{enc}} \leq \epsilon_{\mathsf{auth}} + (d\ell)^2 \left(\epsilon_{\mathsf{PKE}} + \epsilon_{\mathsf{DPRF}} + \epsilon_{\mathsf{PRF}} + 2\epsilon_{\mathsf{StE}}\right)$$

where ϵ_{auth} is an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 3.17 (cf. Lemma 4.15) and all other quantities are defined as stated in Theorem 4.14.

Note that Definition 3.17 for the TLS-RSA with pre-shared key protocol (TLS-RSA-PSK) captures asymmetric perfect forward secrecy (APFS).

Proof. We use the same notions as defined in the proof of Lemma 4.7. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta^{(\mathsf{Enc})}] - 1/2$ denote the advantage of $\mathcal{A}^{\mathsf{Enc}}$ in Game δ. Consider the following sequence of games.

GAME 0. This game equals the ACCE security experiment. For some ϵ_{Enc} we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Enc})}] = \frac{1}{2} + \epsilon_{\mathsf{Enc}} = 1/2 + \mathsf{Adv}_0.$$

GAME 1. The challenger in this game proceeds as before, but it aborts and chooses b' uniformly random, if there exists any oracle that accepts maliciously in the sense of Definition 3.17. Thus we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{Auth}},$$

where ϵ_{auth} an upper bound on the probability that there exists an oracle that accepts maliciously in the sense of Definition 3.17 (cf. Lemma 4.15).

Recall that we assume that $\mathcal{A}^{\mathsf{Enc}}$ always outputs (i, s, b') such that all conditions in Property 2 of Definition 3.17 are satisfied. Note that in Game 1 for any such oracle π_i^s there exists a unique 'partner oracle' π_j^t such that π_i^s has a matching conversation to π_j^t, as the game is aborted otherwise.

GAME 2. The challenger in this game proceeds as before, but in addition guesses the oracle $\pi_{i^*}^{s^*}$ (and its partner oracle $\pi_{j^*}^{t^*}$) for which the adversary breaks the encryption

challenge by drawing random the indices $(i,j) \times (s,t) \xleftarrow{\$} [\ell]^2 \times [d]^2$. With probability $1/(d\ell)^2$ we have $(i,s) \times (j,t) = (i^*,s^*) \times (j^*,t^*)$, and thus

$$\mathsf{Adv}_1 \leq (d\ell)^2 \cdot \mathsf{Adv}_2.$$

GAME 3. In this game we replace the ciphertext c^* sent by the client oracle $\pi_{i^*}^{s^*}$, by a random ciphertext c' of a truly random message. However, the oracle $\pi_{i^*}^{s^*}$ and its partner oracle (if it exists) use a random nonce R^* which is independent of c' to compute the master key. More precisely, since the challenger implements all server oracles it can, whenever the ciphertext c' is received by any server oracle of P_{j^*}, make it use R^*. We show that this modification is indistinguishable from the previous game when the PKE is secure. If there exists an adversary $\mathcal{A}$ who can distinguish these two games, we can use it to construct an algorithm $\mathcal{B}_{\mathsf{PKE}}$ to break the security of the PKE scheme as described in 2.7.

Assuming that an adversary $\mathcal{B}_{\mathsf{PKE}}$ using its encryption oracle $\mathcal{O}^{\mathsf{ENC}}$ and decryption oracle $\mathcal{O}^{\mathsf{DEC}}$ interacts with the challenger $\mathcal{C}_{\mathsf{PKE}}$. $\mathcal{B}_{\mathsf{PKE}}$ acts as a challenger for $\mathcal{A}$ in this game. We show that the simulation of $\mathcal{B}_{\mathsf{PKE}}$ perfectly simulates the challenger in this game from the adversary's point of view. At the beginning of the simulation game, we embed the challenge public key of the PKE challenger in pk_{j^*}. $\mathcal{B}_{\mathsf{PKE}}$ obtains the public parameters and the public key pk' of the challenger $\mathcal{C}_{\mathsf{PKE}}$ and sets $pk_{j^*} = pk'$. For all other oracles $\pi_{j^*}^{t}$ of PID_{j^*} with $t \in [d]$ and $t \neq t^*$, $\mathcal{B}_{\mathsf{PKE}}$ can use its decryption oracle to simulate the protocol executing, i.e., $\mathcal{B}_{\mathsf{PKE}}$ use the decryption queries granted by the PKE challenger to decrypt the ciphertexts. Further, $\mathcal{B}_{\mathsf{PKE}}$ obtains R^* after it is drawn by the client and selects a random message R sends (R^*, R) to the PKE challenger who returns a ciphertext c'. Next $\mathcal{B}_{\mathsf{PKE}}$ send c' to the server oracle $\pi_{j^*}^{t^*}$. Observe that if c' is an encryption of R^* we are in the previous game. However, if c' encrypts an independently drawn random message R we are in the current game. So any attacker that distinguishes these two games can directly be used to break the security of the PKE scheme.

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{PKE}}.$$

GAME 4. Recall that $\pi_{i^*}^{s^*}$ computes the master secret as $\mathsf{ms}^* = \mathsf{PRF}_{\mathsf{TLS}}(\mathsf{pms}^*, \mathsf{label}_1 \,||r_{i^*}^{s^*} \,||r_{j^*}^{t^*})$, where pms^* is the pre-master key. In this game we replace the master secret ms^* computed by $\pi_{i^*}^{s^*}$ with an independent random value $\widetilde{ms^*}$. Moreover, as $\pi_{j^*}^{t^*}$ receives as input the same ciphertext c' that was sent from $\pi_{i^*}^{s^*}$, we set the master secret of $\pi_{j^*}^{t^*}$ equal to $\widetilde{ms^*}$ as well. We exploit that by assumption $\mathsf{PRF}_{\mathsf{TLS}}$ is a $(t, \epsilon_{\mathsf{DPRF}})$-secure DPRF. Therefore, as long as at least one of the values R^* and PSK_{i^*} are chosen at random and are not revealed, the master secret is indistinguishable from random. Any adversary that recognizes our modification can be used to break the security of $\mathsf{PRF}_{\mathsf{TLS}}$. Therefore, we have

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{DPRF}}.$$

GAME 5. We now replace the function $\mathsf{PRF}_{\mathsf{TLS}}(\widetilde{ms^*}, \cdot)$ used by $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ to derive the application keys with a truly random function RF. This game proceeds as the same as the Game 5 in the proof of Lemma 4.12. With the same arguments, we have

$$\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{PRF}}.$$

GAME 6. In this game, we prove the security by showing that, if there is a successful ACCE attacker $\mathcal{A}$, then there exists a probabilistic polynomial-time algorithm $\mathcal{B}_{\mathsf{sLHAE}}$ that can break the security of sLHAE scheme as defined in 2.2.7. This game proceeds as the same as the Game 6 in the proof of Lemma 4.12. With the same arguments, we have

$$\mathsf{Adv}_5 \leq 2\mathsf{Adv}_6$$

Claim 4.18.

$$\mathsf{Adv}_6 = \epsilon_{\mathsf{StE}}.$$

Proof. With the same arguments as 5, we have $\mathsf{Adv}_6 = \epsilon_{\mathsf{StE}}$. □

Summing up the probabilities from Game 0 to Game 6, we proved Lemma 4.17. Finally, summing up probabilities from Lemmas 4.15 and 4.17, we obtain

$$\epsilon_{\mathsf{tls}}^{\mathsf{psk-rsa}} \leq (d\ell)^2 \left(\frac{1}{2^{\lambda-1}} + \epsilon_{\mathsf{PKE}} + 3 \cdot \epsilon_{\mathsf{DPRF}} + 3 \cdot \epsilon_{\mathsf{PRF}} + 2 \cdot \epsilon_{\mathsf{H}} + \frac{1}{2^{\mu-1}} + 6 \cdot \epsilon_{\mathsf{StE}} \right),$$

which proves Theorem 4.9. □

Chapter 5

Tightly-Secure Authenticated Key Exchange in the Standard Model

Contents

In this chapter, we construct the first secure Authenticated Key Exchange (AKE) protocol, called tAKE, in standard model whose security does not degrade with an increasing number of participating parties and sessions. We describe a generic three-pass AKE protocol and prove its security in an enhanced Bellare-Rogaway security model under the standard assumption. Our construction is modular and enjoys a tight security reduction.

The content of this chapter was brought forth in a cooperation with Christoph Bader, Dennis Hofheinz, Tibor Jager and Eike Kiltz. The result is published in the proceedings of the International Conference on Theory of Cryptography Conference (TCC) 2015 [BHJ+15] and in the IACR archive ePrint [BHJ+14]. The authors main contribution within this joined work is authenticated key exchange protocol and the security analysis. The content of this chapter was brought forth in a cooperation with Christoph Bader and Tibor Jager.

SUMMARY OF OUR CONTRIBUTIONS.

The work presented in this chapter provides the first tightly-secure authenticated key exchange protocol whose security is proved in an enhanced Bellare-Rogaway security model described in 3.4.3.2, which allows adaptive corruptions of long-term secret keys, adaptive reveals of session keys and multiple adaptive Test queries. We describe our contribution as follows:

- **The first provably-secure AKE with tight reduction**: We construct the first authenticated key exchange protocol whose security does not degrade with an increasing

number of users and sessions. We give a three-message AKE protocol and prove its security in an enhanced version of the classical Bellare-Rogaway-93 security model. Our construction is generic and modular, moreover, can be instantiated efficiently from standard assumptions. For instance, we give an SXDH-based protocol whose communication complexity is only 14 group elements and 4 exponents.

- **The new, strong security definitions for digital signature and KEM**: We develop new security definitions for digital signatures and key encapsulation mechanisms, i.e existential unforgeability under adaptive chosen-message attacks in the multi-user setting with adaptive corruptions (MU-C-EUF-CMA), and indistinguishability under chosen plain-text attacks in the multi-user setting with corruptions (MU-C-IND-CPA). Moreover, in paper [BHJ$^+$15] we also show how to construct efficient schemes that satisfy the new definitions under standard assumptions.

ORGANIZATION. First, we will elaborate on the importance and difficulty of constructing tightly-secure AKE in Section 5.1. Then, in Section 5.2 we describe our AKE-protocol. Finally, we give a tight proof of security in an enhanced Bellare-Rogaway-93 security model in Section 5.3. Note that the security model $\mathsf{eBR}^{\mathsf{T}}$ described in Section 3.4.3.2 is used for our security analysis of our tightly-secure AKE protocol.

5.1 Importance and Difficulty of Constructing Tightly-secure AKE

The security reduction for AKE protocols only guarantees that no poly-time adversary can break the security properties of AKE protocols with sufficiently high probability. The comparisons of the efficiency of AKE protocols must take into account the efficiency of the security reduction. The quality of a reduction can be measured by its efficiency: the running time and success probability. The reduction is considered as *tight*, if the reduction in which an adversary who breaks an AKE protocol with probability ε in time t can be used to break the underlying hard problem with probability $\varepsilon' \approx \varepsilon$ in time $t' \approx t$. It has been well known that besides theoretical interest, a tight reduction is of utmost practical importance. An AKE protocol with a non-tight reduction has to necessarily require larger parameters sizes (e.g. key sizes) to provide the same security level as a protocol with a tight security reduction. Nature, as we often say, obtaining a reasonable level of security from an AKE protocol with a non-tight reduction is completely impractical. We used a simplified example as in paper [GJKW07] to illustrate the point. For simplicity, we only consider DLOG based public key schemes. Assume that any poly-time adversary requires time t to break an AKE protocol with probability at most $\varepsilon = \mathsf{R} \cdot (2^{-c})^{\mathsf{L}}$, where $\mathsf{R} > 0$ is a factor of the reduction, L is the bit-length of the system parameters for AKE protocol and $c > 0$ is a constant and implies the exoteric factors, including e.g. the attacker computational power and it's memory. If an AKE system requires a security level of probability ε in time t against a poly-time adversary, we compute the bit-length of the system parameters $\mathsf{L} = \frac{[\log_2\left(\frac{\varepsilon^{-1}}{\mathsf{R}}\right)]}{-c}$.

For simplicity of exposition, suppose that $\varepsilon = 2^{-60}$ against any poly-time adversary should require 1 month of attack effort, for $\varepsilon = 2^{-70}$ required approximate 1 year and $\varepsilon = 2^{-80}$ required approximate 10 year. Suppose that $c = \frac{1}{15}$. If an AKE system requires a security level of 2^{60} (i.e., probability of adversary $\varepsilon = 2^{-60}$) against any poly-time adversary investing time t = 1 month, for a tight-reduction $\mathsf{R} = \mathcal{O}(1)$ we need to set $\mathsf{L} \approx 900$. However, for a non-tight reduction, e.g. $\mathsf{R} = 2^{60}$, $\mathsf{L} \approx 1800$. Table 5.1 shows a comparison of the length of system parameters between tight reduction and non-tight reduction. From this table it is clear that in contrast to a tight reduction an AKE system with a non-tight reduction implies a huge decrease in efficiency (time and space) to the same security complexity. Therefore, developing efficient AKE protocols with security

Security Level	**Time: t**	**Reduction Factor: R**	**Constant: c**	**Length: L**	**Security Level**	**Time: t**	**Reduction Factor: R**	**Constant: c**	**Length: L**
2^{60}	1 month	2^{30}	$\frac{1}{15}$	≈ 1350	2^{60}	1 month	$\mathcal{O}(1)$	$\frac{1}{15}$	≈ 900
		2^{60}		≈ 1800					
		2^{90}		≈ 2250					
2^{70}	1 year	2^{30}	$\frac{1}{15}$	≈ 1500	2^{70}	1 year	$\mathcal{O}(1)$	$\frac{1}{15}$	≈ 1050
		2^{60}		≈ 1950					
		2^{90}		≈ 2400					
2^{80}	10 years	2^{30}	$\frac{1}{15}$	≈ 1650	2^{80}	10 years	$\mathcal{O}(1)$	$\frac{1}{15}$	≈ 1200
		2^{60}		≈ 2100					
		2^{90}		≈ 2550					

Fig. 5.1. Comparison of the Length of System Parameters between Tight Reduction and Non-tight Reduction

reduction as tight as possible is the most important for many practical applications of key exchange.

Tightly-secure authenticated key exchange protocol is a challenging problem. To the best of our knowledge, there was no tightly-secure AKE scheme under standard assumptions, though there are multiple well-known AKE schemes in the literature. In this subsection, we introduce two main difficulties with proving tight security of authenticated key exchange protocols, which we would like to explain with concrete example.

- To illustrate the first main difficulty, suppose that a set of honest parties in an AKE scheme, where each honest party P_i, $i \in [\ell]$, has long-term public/secret key pair (pk_i, sk_i) for an authentication scheme, e.g. a digital signature scheme. In the security proof of AKE, we must use the security of the authentication scheme as a security argument for a reduction to the security of the AKE protocol. In other words, we give a reduction from forging an authentication scheme to breaking the AKE protocol.

  In general, in security proof for AKE protocol the reduction implements the challenger $\mathcal{C}_{\mathsf{AKE}}$ in order to take advantage of the adversary $\mathcal{A}$. In the most commonly used security models, e.g. BR, CK or eCK models, the adversary $\mathcal{A}$ is allowed to learn the long-term secret of all parties, except for communication partner of the test oracle selected by $\mathcal{A}$. Assume that $\mathcal{C}_{\mathsf{AKE}}$ firstly guesses the partner identity of

the test oracle P_j, $j \in [\ell]$. The challenger takes a security parameter and runs the setup algorithm to generate public parameter pk_i, $i \in [\ell]$ and $i \neq j$. In oder to use the security of the authentication scheme as an argument, $\mathcal{C}_{\mathsf{AKE}}$ can get a challenge public key pk^* from the security experiment of the authentication scheme and embed the public key $pk_j = pk^*$ in the AKE security experiment. Finally, $\mathcal{C}_{\mathsf{AKE}}$ gives all the public keys to $\mathcal{A}$. During the security experiment $\mathcal{A}$ can learn the long-term secrets of all parties (except for the partner of its test session) using $\mathsf{Corrupt}$ queries. Note that $\mathcal{C}_{\mathsf{AKE}}$ can not answer the $\mathsf{Corrupt}(\mathsf{P}_j)$-query made by the adversary. Therefore, this strategy works only if $\mathcal{C}_{\mathsf{AKE}}$ can correctly guess the partner identity of test oracle selected by $\mathcal{A}$, which occurs with probability $\geq \frac{1}{\ell}$. In our tightly-secure AKE scheme we used *double-key* technique with a non-interactive proof system [Gro06, GOS06, GS08, Gro10, AFG+10, HJ12] in a way somewhat related to the Naor-Yung paradigm [NY90] to avoid this guessing.

- To clarify the second main difficulty, we use signed Diffie-Hellman (signed-DH) protocol as an example. We first give a sketch of signed-DH protocol in Figure 5.2. Then, we explain the second main difficulty for a tightly-secure AKE protocol.

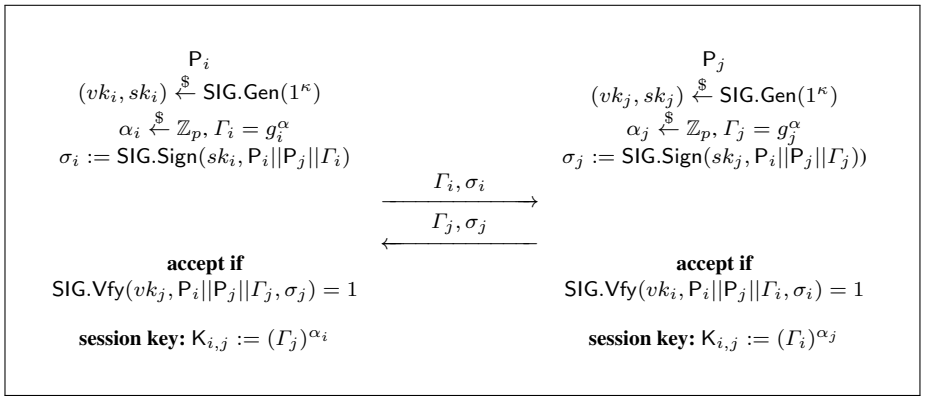


Fig. 5.2. Signed-DH Protocol

In Paper [CK01], Canetti and Krawczyk proved the security of Signed-DH protocol based on the decisional Diffie-Hellman assumption and the security of signature. Even though the DDH problem can be random self-reducible, However, it seems impossible to avoid guessing the test oracle. To explain this, suppose that an adversary $\mathcal{A}$ in the AKE security model described in Section 3.4 establishes a session on parties P_i and P_j. For simplicity, we describe the protocol execution by oracles π_i^s and π_j^t. According to the protocol specification in Figure 5.2, π_i^s sends the messages (Γ_i^s, σ_i^s) to π_j^t. In the AKE security proof, the challenger wants to reduce the AKE security to the DDH assumption, it has to embedded the given DDH-instance (g, g^x, g^y, g^z) in the protocol messages. If the DDH-instance (g^x or g^y) is not embedded in Γ_i^s. However, $\mathcal{A}$ selects oracle π_i^s as its test oracle, then the challenger is not able to take advantage of $\mathcal{A}$. The reason is simple: the DDH-instance is not embedded in the test

session. Otherwise, if the challenger embeds the DDH-instance (g^x or g^y) in Γ_i^s, but, $\mathcal{A}$ does not select π_i^s as its test oracle, it is able to make the simulation to fail. First, $\mathcal{A}$ issues a Corrupt-query to P_j to learn the long-term secret sk_j. Note that since $\mathcal{A}$ does not select π_i^s as its test oracle, according to the security definition 3.10 it is allowed to issue a Corrupt-query to party P_j. $\mathcal{A}$ selects $\alpha_j \xleftarrow{\$} \mathbb{Z}_p$, and computes g_j^α and a (valid) corresponding signature σ_i^s using sk_j. Then, it sends (g^{α_j}, σ_i^s) to π_i^s. π_i^s receives (g^{α_j}, σ_i^s) and verifies the signature σ_i^s. Since σ_i^s is valid, oracle π_i^s accepts. Because π_i^s is not selected as test oracle, $\mathcal{A}$ issues a Reveal(π_i^s)-query to π_i^s. The challenger does not know the ephemeral secrets x or α_j, it can not respond with a correct session key $\mathsf{K}_{i,j} = (g)^{x\alpha_j}$ to $\mathcal{A}$. However, $\mathcal{A}$ is able to correctly compute $\mathsf{K}_{i,j} = (\Gamma_i)^{\alpha_j}$ with α_j. If the challenger responds with an incorrect $\mathsf{K}_{i,j}$, $\mathcal{A}$ can check the validity of the challenger's response whether $(\Gamma_i)^{\alpha_j} \stackrel{?}{=} \mathsf{K}_{i,j}$. The simulation will fail to run.

From the above analysis it seems that the challenger has to guess in advance (at least) one oracle that participates in the test-session. It leads to a loss factor of ($\frac{1}{d\ell}$) in the reduction. Note that the loss factor of ($\frac{1}{d\ell}$) indicates the number of sessions. In practice, it can become very large.

5.2 Authenticated Key Exchange Protocol

In this section, we describe an AKE-protocol which is proved with a tight reduction, named here as tAKE. Our tAKE protocol uses a key transport mechanism that consists of three messages to authenticate both participating parties and to establish a shared session key between both parties. It takes as input the following building blocks:

- A key encapsulation mechanism scheme in the multi-user setting with corruptions described in 2.2.5.1;
- A digital signature scheme in the multi-user setting with corruptions described in 2.2.6.3;
- A strong one-time signature scheme in the multi-user setting without corruptions described in 2.12.

Roughly speaking, the key encapsulation mechanism guarantees that session keys are indistinguishable from random keys. The signature scheme is used to guarantee authentication: the long-term keys of all parties consist of verification keys of the signature scheme. Finally, the one-time signature scheme prevents oracles from accepting without having a unique partner oracle. In our paper [BHJ+15], we showed how to construct efficient schemes that satisfy the new definitions with tight security proofs under standard assumptions.

5.2.1 Protocol Description

We construct a tight AKE protocol tAKE, which is based on three building blocks: a key encapsulation mechanism as specified in Section 2.9 $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$=($\mathsf{KEM.Gen}^{\mathsf{C}}_{\mathsf{MU}}$, $\mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}$, $\mathsf{KEM.Decap}^{\mathsf{C}}_{\mathsf{MU}}$), a signature scheme as specified in Section 2.11 $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ = ($\mathsf{SIG.Gen}_{\mathsf{MU}}$, $\mathsf{SIG.Sign}_{\mathsf{MU}}$, $\mathsf{SIG.Vfy}_{\mathsf{MU}}$) and a one-time signature scheme as specified in Section 2.12 $\mathsf{OTSIG}_{\mathsf{MU}}$= ($\mathsf{OTSIG.Gen}_{\mathsf{MU}}$, $\mathsf{OTSIG.Sign}_{\mathsf{MU}}$, $\mathsf{OTSIG.Vfy}_{\mathsf{MU}}$).

The tAKE protocol between two parties P_i and P_j proceeds as follows, which is also depicted in Figure 5.3.

1. **The first step:** Party P_i generates the key encapsulation mechanism keys and one time signature keys by calling $(sk^i_{\mathsf{KEM}}, pk^i_{\mathsf{KEM}}) \xleftarrow{\$} \mathsf{KEM.Gen}^{\mathsf{C}}_{\mathsf{MU}}(1^\kappa)$ and $(vk^i_{\mathsf{OTS}}, sk^i_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTSIG.Gen}_{\mathsf{MU}}(1^\kappa)$ and computes the signature of vk^i_{OTS}: $\sigma^i = \mathsf{SIG.Sign}_{\mathsf{MU}}(sk^i, vk^i_{\mathsf{OTS}})$. It defines the partner identity $\mathsf{PID}^{\mathsf{Partner}}_i = j$ and $\mathsf{T}_1 := (vk^i_{\mathsf{OTS}}, \sigma^i, pk^i_{\mathsf{KEM}}, \mathsf{PID}^{\mathsf{Partner}}_i, i)$ and transmits T_1 to P_j.
2. **The second step:** upon receiving T_1, party P_j parses T_1 as the tuple $(vk^i_{\mathsf{OTS}}, \sigma^i, pk^i_{\mathsf{KEM}}, \mathsf{PID}^{\mathsf{Partner}}_i, i)$. Then it checks whether $\mathsf{PID}^{\mathsf{Partner}}_i = j$ and $\mathsf{SIG.Vfy}_{\mathsf{MU}}(vk^i, vk^i_{\mathsf{OTS}}, \sigma^i) = 1$. If at least one of both check is not passed, then P_j outputs $\perp$ and rejects. Otherwise it generates one-time signature keys by calling $(vk^j_{\mathsf{OTS}}, sk^j_{\mathsf{OTS}}) \xleftarrow{\$} \mathsf{OTSIG.Gen}_{\mathsf{MU}}(1^\kappa)$ and $(\mathsf{K}, \mathsf{C}_{\mathsf{KEM}}) \xleftarrow{\$} \mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}(pk^i_{\mathsf{KEM}})$ and computes $\sigma^j := \mathsf{SIG.Sign}_{\mathsf{MU}}(sk^j, vk^j_{\mathsf{OTS}})$. Then it sets $\mathsf{T}_2 := (vk^j_{\mathsf{OTS}}, \sigma^j, \mathsf{C}_{\mathsf{KEM}})$ and computes a one-time signature $\sigma^j_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}_{\mathsf{MU}}(sk^j_{\mathsf{OTS}}, (\mathsf{T}_1, \mathsf{T}_2))$ and transmits the tuple $(\mathsf{T}_2, \sigma^j_{\mathsf{OTS}})$ to party P_i.
3. **The third step:** upon receiving $(\mathsf{T}_2, \sigma^j_{\mathsf{OTS}})$, party P_i parses T_2 as $(vk^j_{\mathsf{OTS}}, \sigma^j, \mathsf{C}_{\mathsf{KEM}})$ and checks whether $\mathsf{SIG.Vfy}_{\mathsf{MU}}(vk^j, vk^j_{\mathsf{OTS}}, \sigma^j) = 1$ and $\mathsf{OTSIG.Vfy}_{\mathsf{MU}}(vk^j_{\mathsf{OTS}}, (\mathsf{T}_1, \mathsf{T}_2), \sigma^j_{\mathsf{OTS}}) = 1$. If at least one of both check is not passed, then party P_i outputs $\perp$ and rejects. Otherwise it computes $\sigma^i_{\mathsf{OTS}} := \mathsf{OTSIG.Sign}_{\mathsf{MU}}(sk^i_{\mathsf{OTS}}, (\mathsf{T}_1, \mathsf{T}_2))$ and sends σ^i_{OTS} to party P_j. Finally, party P_i computes and outputs the session key $\mathsf{K}_{i,j} := \mathsf{KEM.Decap}^{\mathsf{C}}_{\mathsf{MU}}(sk^i_{\mathsf{KEM}}, \mathsf{C}_{\mathsf{KEM}})$.
4. **The final step:** upon receiving σ^i_{OTS}, party P_j checks whether $\mathsf{OTSIG.Vfy}_{\mathsf{MU}}(vk^i_{\mathsf{OTS}}, (\mathsf{T}_1, \mathsf{T}_2), \sigma^i_{\mathsf{OTS}}) = 1$. If this fails, then $\perp$ is returned. Otherwise party P_j outputs the session key $\mathsf{K}_{i,j} := \mathsf{K}$.

5.3 Security Analysis of Tightly-Secure AKE Protocol

In this section we prove that the proposed authenticated key exchange protocol as described in 5.2 is secure in the sense of the security guarantees as specified in Section 3.10 in the standard model.

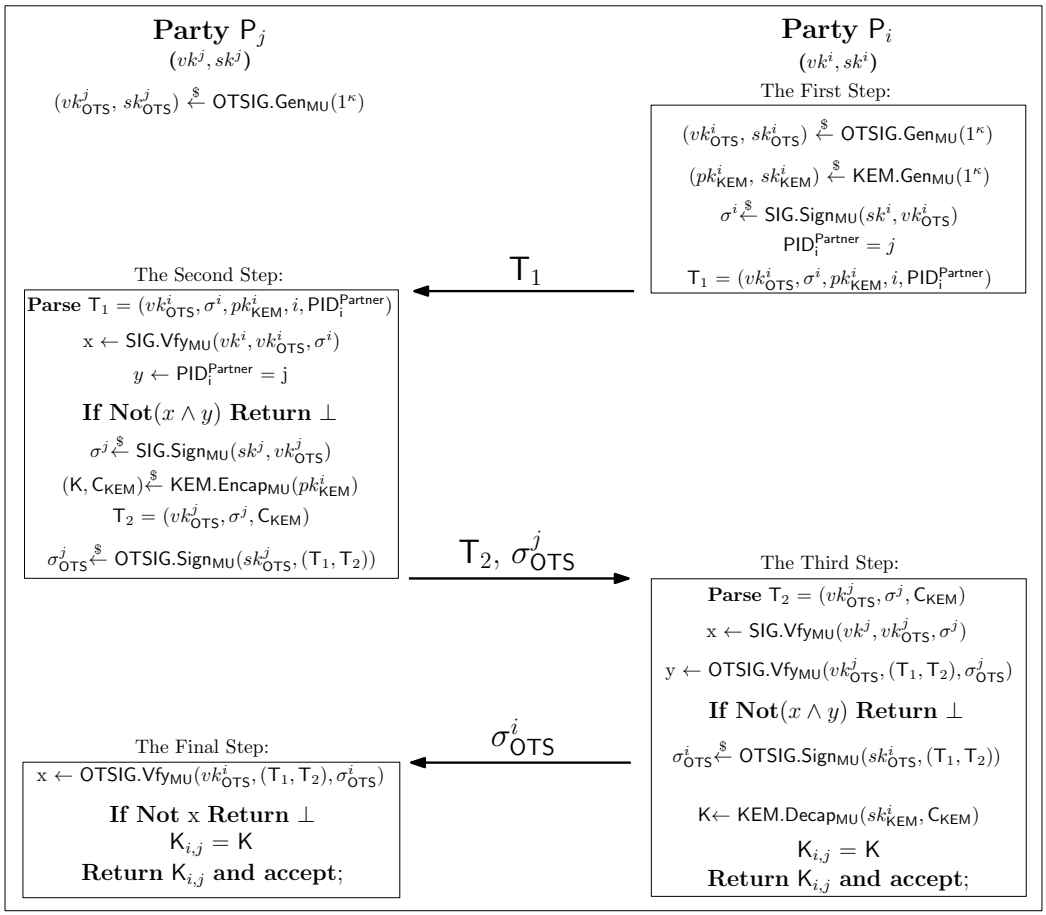


Fig. 5.3. Three-Message AKE-Construction for Extended BR-Security

Theorem 5.1. *Assume that the key encapsulation mechanism* $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ *is a (*d*,* ℓ*,* t*,* $\epsilon_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$*)-secure with respect to Definition 2.9, the signature scheme* $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ *is a (*q_{SIG}*,* ℓ*,* t*,* $\epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$*)-secure with respect to Definition 2.11, and the one-time signature is a (1,* ℓ*,* t*,* $\epsilon_{\mathsf{OTSIG_{MU}}}$*)-secure with respect to Definition 2.12. Then the protocol described in Section 5.2.1 is a* $(t', \epsilon_{\mathsf{tAKE}})$*-secure authenticated key exchange protocol* tAKE *in the sense of Definition 3.10 with* $t' \approx t$ *and*

$$\epsilon_{\mathsf{tAKE}} \leq 4\epsilon_{\mathsf{OTSIG_{MU}}} + 2\epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}} + \epsilon_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}.$$

We prove Theorem 5.1 in two stages. First, we show that the AKE protocol is a secure authentication protocol except for probability ϵ_{auth}, that is, the protocol fulfills security property 1.) of the AKE definition 3.10 by Lemma 5.2. In the next step, we show that the session key of the AKE protocol is secure except for probability ϵ_{Ind} in the sense of the Property 2.) of the AKE definition 3.10 by Lemma 5.4. Thus, we have

$$\epsilon_{\mathsf{tAKE}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Ind}}.$$

5.3.1 Authentication Property

Lemma 5.2. *For any adversary $\mathcal{A}^{\mathsf{Auth}}$ running in time $t' \approx t$, the probability that there exists an oracle π_i^s that accepts maliciously in the sense of Definition 3.10 is at most*

$$\epsilon_{\mathsf{Auth}} \leq \epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}} + 2\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}},$$

where all quantities are defined as stated in Theorem 5.1.

Proof. Let $\mathsf{break}_{\delta}^{(\mathsf{Auth})}$ be the event that there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has internal state $\Phi_{i^*}^{s^*} = \mathtt{accept}$ and $\mathsf{PID}_{i^*}^{s^*} = j^*$, but there is no unique oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations, in Game δ.

GAME 0. This is the original security game. Thus we have that

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}.$$

GAME 1. In this game, the challenger proceeds exactly like the challenger in Game 0, except that we add an abortion rule. The challenger raises event $\mathsf{abort}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ and aborts if the following condition holds:

- there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has $\mathsf{PID}_{i^*}^{s^*} = j^*$ and $\Phi_{i^*}^{s^*} = \mathtt{accept}$,
- the signature $\sigma_{j^*}^{t^*}$ received by $\pi_{i^*}^{s^*}$ that is computed over the one-time public key $vk_{j^*,\mathsf{OTS}}^{t^*}$ and verified correctly under the long-term public key vk_{j^*},
- but there is no unique oracle $\pi_{j^*}^{t^*}$ which has previously output a valid signature $\sigma_{j^*}^{t^*}$ over this one-time public key $vk_{j^*,\mathsf{OTS}}^{t^*}$.

Clearly, we have

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_1^{(\mathsf{Auth})}] + \Pr[\mathsf{abort}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}].$$

If the event $\mathsf{abort}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ happens, then we construct a signature forger $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ against the MU-C-EUF-CMA security of signature scheme $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ as follows.

Firstly, $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ receives the public keys $\{\ vk^i_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}},\ i \in [\ell]\ \}$ from the $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ as input, and runs the adversary $\mathcal{A}^{\mathsf{Auth}}$ as a subroutine simulating the challenger $\mathcal{C}_{\mathsf{AKE}}$ for $\mathcal{A}^{\mathsf{Auth}}$. It sets $vk^i_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ as public key for party with the identity $i \in [\ell]$, and sends the public keys to the adversary $\mathcal{A}^{\mathsf{Auth}}$. If $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ needs to sign a message under $vk^i_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$, it can use its signature oracle $\mathcal{O}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ to generate a signature of that message. If the adversary issues Corrupt-query to $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$, it can forward this query to its $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ challenger, and forwards the response back to $\mathcal{A}^{\mathsf{Auth}}$. Except for this, $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ proceeds exactly like the challenger in Game 0. Using its $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$ challenger, $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ can answer

all Corrupt queries made by $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$, and does not need to guess which party will be corrupted by $\mathcal{A}^{\mathsf{Auth}}$ beforehand. If the event $\mathsf{abort}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ is raised, then this means that $\mathcal{A}^{\mathsf{Auth}}$ has a valid forgery $(vk^{t^*}_{j^*,\mathsf{OTS}}, \sigma^{t^*}_{j^*})$ on behalf of an uncorrupted party P_{j^*}, and it was not output by any oracle $\pi^{t^*}_{j^*}$. Since $\mathcal{F}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$ has never requested a signature for this one-time public key $vk^{t^*}_{j^*,\mathsf{OTS}}$ from its challenger $\mathcal{C}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$, it can use this forgery to break the MU-C-EUF-CMA security of signature scheme $\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}$. Therefore, we have $\Pr[\mathsf{abort}_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}] \leq \epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}$, i.e.,

$$\Pr[\mathsf{break}^{(\mathsf{Auth})}_0] \leq \Pr[\mathsf{break}^{(\mathsf{Auth})}_1] + \epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}}.$$

GAME 2. In this game, the challenger $\mathcal{C}_{\mathsf{AKE}}$ proceeds exactly like the challenger in Game 1, except that we add an abortion rule $\mathsf{abort}_{\mathsf{collision}}$. We let $\mathsf{abort}_{\mathsf{collision}}$ be the event that two oracles sample the same verification key vk_{OTS} for the one-time signature in multi-user setting without corruptions (MU-OTS-EUF-CMA).

Clearly, we have

$$\Pr[\mathsf{break}^{(\mathsf{Auth})}_1] \leq \Pr[\mathsf{break}^{(\mathsf{Auth})}_2] + \Pr[\mathsf{abort}_{\mathsf{collision}}].$$

If the event $\mathsf{abort}_{\mathsf{collision}}$ happens, then we could construct a one-time signature forger $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ against the MU-OTS-EUF-CMA security of $\mathsf{OTSIG}_{\mathsf{MU}}$ as described in Section 2.12 as follows. Note that $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ proceeds exactly like the challenger in Game 1, except that it does not generate the one-time verification keys $\{ vk^1_{\mathsf{OTS}}, \cdots, vk^{d\ell}_{\mathsf{OTS}} \}$ on its own, but instead receives them from its $\mathsf{OTSIG}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$. With probability $\Pr[\mathsf{abort}_{\mathsf{collision}}]$, there exists two oracles which have the same verification key, $vk^a_{\mathsf{OTS}} = vk^b_{\mathsf{OTS}}$ for $a \neq b$ and $a, b \in [d\ell]$. If this event $\mathsf{abort}_{\mathsf{collision}}$ is raised, then $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ sends a sign query $\mathsf{OTSIG}_{\mathsf{MU}}$ with (a, T) to its challenger $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$, and then obtains a signature σ_{OTS} from $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$. Since $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ has never requested a one-time signature for T from its challenger $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$, it outputs $(b, \mathsf{T}, \sigma_{\mathsf{OTS}})$ as its valid forgery against the MU-OTS-EUF-CMA security of signature scheme $\mathsf{OTSIG}_{\mathsf{MU}}$. Therefore, we have $\Pr[\mathsf{abort}_{\mathsf{collision}}] \leq \epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}$, i.e.,

$$\Pr[\mathsf{break}^{(\mathsf{Auth})}_1] \leq \Pr[\mathsf{break}^{(\mathsf{Auth})}_2] + \epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}.$$

GAME 3. In this game, the challenger proceeds exactly like the challenger in Game 2, except that we add an abortion rule. The challenger raises event $\mathsf{abort}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ and aborts if the following condition holds:

- there exists a τ and a τ-fresh oracle $\pi^{s^*}_{i^*}$ that has $\mathsf{PID}^{s^*}_{i^*} = j^*$ and $\Phi^{s^*}_{i^*} = \texttt{accept}$,
- the one-time signature $\sigma^{t^*}_{j^*,\mathsf{OTS}}$ received by $\pi^{s^*}_{i^*}$ that is computed over the transcript $\mathsf{T} = (\mathsf{T}_1, \mathsf{T}_2)$ and verified correctly under the long-term public key $vk_{j^*,\mathsf{OTS}}$,
- but there is no unique oracle $\pi^{t^*}_{j^*}$ which has previously output a valid signature $\sigma^{t^*}_{j^*,\mathsf{OTS}}$ over the transcript $\mathsf{T} = (\mathsf{T}_1, \mathsf{T}_2)$.

Clearly, we have

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_3^{(\mathsf{Auth})}] + \Pr[\mathsf{abort}_{\mathsf{OTSIG}_{\mathsf{MU}}}].$$

If the event $\mathsf{abort}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ happens, then we could construct a one-time signature forger $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ against the MU-OTS-EUF-CMA security of signature scheme $\mathsf{OTSIG}_{\mathsf{MU}}$ as described in Section 2.12. Firstly, $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ receives the one-time verification key from its $\mathsf{OTSIG}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ and sends it to the adversary. If $\mathcal{F}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ needs to compute a one-time signature, it can ask its $\mathsf{OTSIG}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{OTSIG}_{\mathsf{MU}}}$. If the event $\mathsf{abort}_{\mathsf{OTSIG}_{\mathsf{MU}}}$ is raised, then it means that the adversary $\mathcal{A}^{\mathsf{Auth}}$ has a valid forgery $\sigma_{j^*,\mathsf{OTS}}^{t^*}$ of the transcript T on behalf of an uncorrupted party P_{j^*}, and it was not output by any oracle $\pi_{j^*}^{t^*}$. Since $\mathcal{F}_{\mathsf{SIG}_{\mathsf{MU}}^{\mathsf{C}}}$ has never requested a one-time signature for T from its challenger $\mathcal{C}_{\mathsf{SIG}_{\mathsf{MU}}^{\mathsf{C}}}$, it can use this forgery to break the MU-OTS-EUF-CMA security of the one-time signature scheme $\mathsf{OTSIG}_{\mathsf{MU}}$. Therefore, we have $\Pr[\mathsf{abort}_{\mathsf{OTSIG}_{\mathsf{MU}}}] \leq \epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}$, i.e.,

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_3^{(\mathsf{Auth})}] + \epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}.$$

Claim 5.3. $\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] = 0$

Proof. Note that $\mathsf{break}_3^{(\mathsf{Auth})}$ occurs only if there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has internal state $\Phi_{i^*}^{s^*} = \mathtt{accept}$ and $\mathsf{PID}_{i^*}^{s^*} = j^*$, but there is no *unique* oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations. Due to Game 1 there exists at least one uncorrupted oracle $\pi_{j^*}^{t^*}$ which has output a valid one-time signature $\sigma_{j^*,\mathsf{OTS}}^{t^*}$ over $vk_{j^*,\mathsf{OTS}}^{t^*}$ received by oracle $\pi_{i^*}^{s^*}$, otherwise the execution is aborted. Due to Game 2 the one-time verification key $vk_{j^*,\mathsf{OTS}}^{t^*}$ generated by $\pi_{j^*}^{t^*}$ is unique. If $\pi_{i^*}^{s^*}$ accepts, it muss receive a valid one-time signature $\sigma_{j^*,\mathsf{OTS}}^{t^*}$ over the transcript $\mathsf{T} = (\mathsf{T}_1, \mathsf{T}_2)$. From Game 3, we know that there must exist an oracle which has generated $\sigma_{j^*,\mathsf{OTS}}^{t^*}$ over T. Since T contains $vk_{j^*,\mathsf{OTS}}^{t^*}$. Therefore, if $\pi_{i^*}^{s^*}$ accepts, then it must have a matching conversation to oracle $\pi_{j^*}^{t^*}$. Therefore we have

$$\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] = 0.$$

□

Summing up the probabilities from Game 0 to Game 3, we proved Lemma 5.2, i.e.,

$$\epsilon_{\mathsf{Auth}} \leq \epsilon_{\mathsf{SIG}_{\mathsf{MU}}^{\mathsf{C}}} + 2\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}.$$

□

5.3.2 Key Indistinguishability

Lemma 5.4. *For any adversary $\mathcal{A}^{\mathsf{Ind}}$ running in time $t' \approx t$, the probability that $\mathcal{A}^{\mathsf{Ind}}$ answers the test-challenge correctly in the sense of Definition 3.10 is at most $(1/2+\epsilon_{\mathsf{Ind}})$ with*

$$\epsilon_{\mathsf{Ind}} \leq \epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}} + 2\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}} + \epsilon_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}.$$

All quantities are defined as stated in Theorem 5.1.

Proof. Let $\mathsf{break}^{(\mathsf{Ind})}_{\delta}$ denote the event that the $\mathcal{A}_{\mathsf{AKE}}$ correctly guesses the bit b' sampled by the Test-query in Game δ, and $\mathsf{Test}(\pi^{s^*}_{i^*})$ is the τ-th query of $\mathcal{A}_{\mathsf{AKE}}$, and $\pi^{s^*}_{i^*}$ is a τ-fresh oracle that is ∞-revealed throughout the security game. Let $\mathsf{Adv}_{\delta} := |\Pr[\mathsf{break}^{(\mathsf{Ind})}_{\delta}] - 1/2|$ denote the advantage of $\mathcal{A}_{\mathsf{AKE}}$ in Game δ[1] We proceed in games as in [Sho04, BR06].

GAME 0. This is the original security game. Thus we have that

$$\Pr[\mathsf{break}^{(\mathsf{Ind})}_{0}] = \epsilon_{\mathsf{Ind}} + 1/2 = \mathsf{Adv}_0 + 1/2.$$

GAME 1. The challenger $\mathcal{C}_{\mathsf{AKE}}$ in this game proceeds as before, which aborts if the test oracle $\pi^{s^*}_{i^*}$ accepts, i.e., $\Phi^{s^*}_{i^*} = \texttt{accept}$ and $\mathsf{PID}^{s^*}_{i^*} = j^*$, but there is no unique partner oracle $\pi^{t^*}_{j^*}$ such that $\pi^{s^*}_{i^*}$ and $\pi^{t^*}_{j^*}$ have matching conversations. This is exactly the event $\mathsf{break}^{(\mathsf{Auth})}_{0}$ from the proof of 5.3.1. Thus we have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{Auth}} \leq \mathsf{Adv}_1 + (\epsilon_{\mathsf{SIG}^{\mathsf{C}}_{\mathsf{MU}}} + 2\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}}),$$

where ϵ_{Auth} is an upper bound on the probability that there exists an oracle that accepts without unique partner oracle in the sense of Definition 3.10 (cf. Lemma 5.2).

Lemma 5.5. $\mathsf{Adv}_1 \leq \epsilon_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.

Proof. We show in this game if there exists an adversary $\mathcal{A}_{\mathsf{AKE}}$ who can correctly answer the Test-query, then we use it to construct an algorithm $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ to break the MU-IND-CPA security of key encapsulation mechanism $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ as described in Section 2.9.

Assuming that an adversary $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ using its Corrupt and Encaps oracles interacts with $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that is a challenger in the security game of the $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ scheme as in Section 2.9. Then, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ acts as a challenger for the AKE adversary $\mathcal{A}_{\mathsf{AKE}}$ in this game. We show that the simulation of $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ perfectly simulates the challenger $\mathcal{C}_{\mathsf{AKE}}$ in this game from the adversary's point of view, i.e., (I) showing how $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates protocol execution environment by a polynomial number of oracles; (II) showing how $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can perfectly answer to all queries issued by $\mathcal{A}_{\mathsf{AKE}}$; (III) Finally, if the adversary $\mathcal{A}_{\mathsf{AKE}}$

[1] For simplicity, assume that $\Pr[\mathsf{break}^{(\mathsf{Ind})}_{\delta}] \geq \frac{1}{2}$.

can correctly output a bit b' for Test-query, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can use it to break the security of $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$.

Simulation-Step I:

We firstly describe this simulation for protocol ***execution environment***. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ uses oracles to simulate the execution environment as follows:

- At the beginning of the simulation game, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, and honestly generates all system parameters and all long-term verification/private key pairs $(vk_{\mathsf{PID}_i}, sk_{\mathsf{PID}_i})$ for each honest party PID_i, $i \in [\ell]$ according to the protocol specification, and sends all public parameters as well as all long-term verification keys to the adversary.
- In case of the computation of ephemeral $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ key pairs for each session, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ asks its challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and gets $d\ell$ ephemeral public keys and sets ($\tilde{pk}^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $\tilde{sk}^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \emptyset$) as the ephemeral $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ key pairs. We denote $\mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \{ \tilde{pk}^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, i \in [\ell] \text{ and } s \in [d] \}$ as the set of ephemeral public keys from the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.
- Then, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates the protocol process as follows.
  - Firstly, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates **the first step** of the protocol specification as described in Section 5.2.1, denoted as the initiator oracle π_i^s. It proceeds exactly like in previous game, except for in case of the computation of ephemeral $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ key pairs for each session. I.e. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ randomly selects $\tilde{pk}^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ as ephemeral public key for each session. Note that since $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ does not know the corresponding ephemeral secret keys, it sets $\tilde{sk}^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \emptyset$. This will not affect our simulation since all legitimate queries supplied by the adversary $\mathcal{A}_{\mathsf{AKE}}$ can be correctly answered by $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.
  - Then $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates **the second step** of the protocol specification as described in 5.2.1, denoted as the response oracle π_j^t. It proceeds exactly like in previous game, except for in case of the computation of $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ ciphertext for each session.

    Let $\mathsf{T}_1 = (vk^s_{i,\mathsf{OTS}}, \sigma^i, pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \mathsf{PID}_i^{\mathsf{Partner}}, i)$ be the message that oracle π_j^t receives. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates the oracle π_j^t as follows. We consider two cases:
    - **Case 1:** The ephemeral public key $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is generated by the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ asks its challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ to compute (K, C) as $(\tilde{\mathsf{K}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \tilde{\mathsf{C}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}) \xleftarrow{\$} \mathcal{O}^{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}(pk^i_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$. We denote $\mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \{ \tilde{\mathsf{C}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, j \in [\ell] \text{ and } t \in [d] \}$ as the set of $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ ci-

phertexts from the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Note that $\tilde{\mathsf{C}}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is a valid ciphertext for $pk^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. However, $\tilde{\mathsf{K}}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is either a random session key or a corresponding (valid) $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ session key. The $\tilde{\mathsf{K}}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is used to respond to Test-query.

- **Case 2:** The ephemeral public key $pk^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is generated by the adversary $\mathcal{A}_{\mathsf{AKE}}$, i.e., $pk^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \notin \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can generate a ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ along with a $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ session key $\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ by calling the encapsulation algorithms, $(\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}},\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}) \xleftarrow{\$} \mathsf{KEM.Encap}^{\mathsf{C}}_{\mathsf{MU}}(pk^{s}_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$. Note that $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ knows the corresponding session key $\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ for the ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.

– Next, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates **the third step** of the protocol specification as described in 5.2.1, i.e., π^{s}_{i}.

$\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ has received $(\mathsf{T}_2, \sigma^{j}_{\mathsf{OTS}})$, where $\mathsf{T}_2 := (vk^{j}_{\mathsf{OTS}}, \sigma^{j}, \mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$ and $\sigma^{j}_{\mathsf{OTS}}$ is a one-time signature over $(\mathsf{T}_1, \mathsf{T}_2)$, it first checks the validity of the signatures. If all signatures are verified successfully, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ will compute a one-time signature according to the protocol specification and accept. For $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ received by oracle π^{s}_{i} we also consider the following two cases:

- **Case 1:** The ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ received by oracle π^{s}_{i} is generated by the adversary $\mathcal{A}_{\mathsf{AKE}}$, $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \notin \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can perform Corrupt-query and obtain the corresponding secure $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ key $sk^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Then, it can decrypt the ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and compute the session key $\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Note that $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ knows the corresponding session key $\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ for the ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. This implies that $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ performs a ciphertext validity check for $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \notin \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. If the ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is not valid, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ aborts and terminates the simulation.
- **Case 2:** The ciphertext $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ received by oracle π^{s}_{i} is generated by the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $\mathsf{C}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ does not know the corresponding session key and sets $\mathsf{K}^{t}_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \emptyset$. However, this will not affect our simulation since all legitimate queries supplied by the adversary $\mathcal{A}_{\mathsf{AKE}}$ can be correctly answered by $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. We will later describe that $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can perfectly simulate the tAKE challenger for the adversary $\mathcal{A}_{\mathsf{AKE}}$.

– Finally, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ simulates **the final step** of the protocol specification as described in 5.2.1, i.e., π^{t}_{j}.

$\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ has received one-time signature $\sigma^{i}_{\mathsf{OTS}}$ over $(\mathsf{T}_1, \mathsf{T}_2)$, it first checks the validity of the signatures. If the one-time signature is verified successfully, then

it accepts. Recall that $\mathsf{T}_1 = (vk^s_{i,\mathsf{OTS}}, \sigma^i, pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \mathsf{PID}_i^{\mathsf{Partner}}, i)$ and $\mathsf{T}_2 := (vk^j_{\mathsf{OTS}}, \sigma^j, \mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$. We also consider the following two cases:

- **Case 1:** The ephemeral public key $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that is received by oracle π^t_j in the second step is generated by the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ does not know the corresponding session key and sets $\mathsf{K}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} = \emptyset$. However, this will not impact our simulation since all legitimate queries issued by $\mathcal{A}_{\mathsf{AKE}}$ can be correctly answered by $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.
- **Case 2:** The ephemeral public key $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that is received by oracle π^t_j in the second step is generated by the adversary $\mathcal{A}_{\mathsf{AKE}}$, $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \notin \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. In this case $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ knows the corresponding session key $\mathsf{K}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ for the ciphertext $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and sets $\mathsf{K}^t_j = \mathsf{K}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.

Simulation-Step II:

Next, we describe ***the simulation of the challenger*** of AKE security experiment for the adversary $\mathcal{A}_{\mathsf{AKE}}$, and show that $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can perfectly answer all queries issued by $\mathcal{A}_{\mathsf{AKE}}$ as follows:

- $\mathsf{Corrupt}(\mathsf{PID}_i)$: $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ answers to this query exactly as the previous game. It responds with the long-term private key sk_i of party PID_i.
- $\mathsf{RegCorruptParty}(pk_c, \mathsf{P}_c)$: $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ answers to this query exactly as the previous game.
- $\mathsf{Reveal}(\pi^s_i)$: If the adversary $\mathcal{A}_{\mathsf{AKE}}$ issues a $\mathsf{Reveal}(\pi^s_i)$-query and oracle π^s_i has internal state $\Phi^s_i = \mathtt{accept}$, then $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ responds to this query with the contents of variable K^s_i to $\mathcal{A}_{\mathsf{AKE}}$. For this query we consider the following two cases:
  - **Case 1:** $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$: In this case the ciphertext $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is computed by the challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ does not know the corresponding session key. In order to respond to this query, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can issue a Corrupt-query to its $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and receive its corresponding secret key from $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Then it can decapsulate the ciphertext $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ and compute the session key, i.e., $sk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \xleftarrow{\$} \mathcal{O}^{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}_{\mathsf{Corrupt}}(pk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$ and $\mathsf{K}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \xleftarrow{\$} \mathsf{KEM}.\mathsf{Decap}^{\mathsf{C}}_{\mathsf{MU}}(sk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$. Finally, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ responds to this query with $\mathsf{K}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ to $\mathcal{A}_{\mathsf{AKE}}$.
  - **Case 2:** $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \notin \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$: In this case, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ knows the contents of the session keys, and can simply returns the session key to $\mathcal{A}_{\mathsf{AKE}}$.
- $\mathsf{Test}(\pi^s_i)$: In our security model we allow the adversary $\mathcal{A}_{\mathsf{AKE}}$ to issue this query if oracle π^s_i τ-fresh at the point in time that this query is issued by $\mathcal{A}_{\mathsf{AKE}}$. Due to

Game 0 we can conclude that π_i^s has a unique partner oracle π_j^t. By the definition of freshness, the party PID_j is $\tau^{(j)}$-corrupted with $\tau^{(j)} > \tau$. Moreover, from the simulation of protocol execution of **the second step** we can conclude that $\mathsf{C}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{CS}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. The $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ responds with $(\tilde{\mathsf{K}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \tilde{\mathsf{C}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$ to $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Note that $\tilde{\mathsf{K}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is a challenge key computed by the $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ in $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ experiment.

Simulation-Step III:

Finally, $\mathcal{A}_{\mathsf{AKE}}$ terminates. We show that if there exists an adversary $\mathcal{A}_{\mathsf{AKE}}$ that can correctly guess a bit b of Test-query, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is able to use it to break the security of $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ scheme. We denote that (b', i, s) is the output of the adversary $\mathcal{A}_{\mathsf{AKE}}$ for the Test-query $\mathsf{Test}(\pi_i^s)$. According to the security definition as described in Section 3.4.3.2, $\mathcal{A}_{\mathsf{AKE}}$ wins only if there is τ such that test oracle π_i^s is τ-fresh and $b' = b$. We also consider the following two cases:

- **Case 1:** π_i^s is an initiator oracle: In this case it means that it's $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ public key $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that is used to encapsulate the session key by its partner oracle π_j^t is generated by the $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $pk^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. Recall that the simulation of oracle π_j^t, $\tilde{\mathsf{C}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is sent to the initiator oracle π_i^s. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is able to respond with the session key $\tilde{\mathsf{K}}^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that was generated by its $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger to $\mathcal{A}_{\mathsf{AKE}}$. If $\mathcal{A}_{\mathsf{AKE}}$ terminates with outputting b', $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can simply forward $\mathcal{A}_{\mathsf{AKE}}$'s response to its challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.
- **Case 2:** π_i^s is an response oracle: In this case π_i^s generates a ciphertext for a $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ public key $pk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that was sampled by its partner oracle π_j^t. Due to freshness-rules as described in 3.9, party PID_j is τ-uncorrupted. Therefore $pk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is generated by the $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$, $pk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}} \in \mathsf{EPK}^{\mathsf{Set}}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can issue an Encaps-query and ask its challenger to get a pair $(\tilde{\mathsf{K}}^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}, \tilde{\mathsf{C}}^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}) \xleftarrow{\$} \mathcal{O}^{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}(pk^t_{j,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}})$. $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is able to respond a Test-query $\mathsf{Test}(\pi_i^s)$ with the session key $\tilde{\mathsf{K}}^s_{i,\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ that was generated by its $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ challenger to $\mathcal{A}_{\mathsf{AKE}}$. If $\mathcal{A}_{\mathsf{AKE}}$ terminates with outputting b', $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ can simply forward $\mathcal{A}_{\mathsf{AKE}}$'s response to its challenger $\mathcal{C}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$.

Observe that in either case, if $\mathcal{A}_{\mathsf{AKE}}$ can correctly guess the value b used to answer the Test-query, $\mathcal{F}_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}$ is able to break the security of $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$ scheme. Exploiting the security of the $\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}$, we obtain that

$$\mathsf{Adv}_1 \leq \epsilon_{\mathsf{KEM}^{\mathsf{C}}_{\mathsf{MU}}}.$$

Summing up the probabilities, we yield Lemma 5.4, i.e.,

$$\epsilon_{\mathsf{Ind}} \leq \epsilon_{\mathsf{SIG}^{\mathsf{c}}_{\mathsf{MU}}} + 2\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}} + \epsilon_{\mathsf{KEM}^{\mathsf{c}}_{\mathsf{MU}}}.$$

□

Collecting probabilities from Lemma 5.2 and Lemma 5.4, we conclude that

$$\epsilon_{\mathsf{tAKE}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Ind}} \leq 4\epsilon_{\mathsf{OTSIG}_{\mathsf{MU}}} + 2\epsilon_{\mathsf{SIG}^{\mathsf{c}}_{\mathsf{MU}}} + \epsilon_{\mathsf{KEM}^{\mathsf{c}}_{\mathsf{MU}}}.$$

This is the same advantage as in Theorem 5.1 and hence Theorem 5.1 follows. This completes the proof of the theorem.

□

Chapter 6

New Efficient Compilers for Authenticated Key Exchange

Contents

In this chapter we present two new efficient compilers that generically turn passively secure key exchange protocols (KE) into authenticated key exchange protocols (AKE) where security also holds in the presence of active adversaries.

The content of this chapter was brought forth in a cooperation with Sven Schäge, Zheng Yang, Jörg Schwenk and Christoph Bader. The result is published in the proceedings of the *International Conference on Applied Cryptography and Network Security (ACNS) 2014* [LSY+14a]. The results from [LSY+14a] were also published in Zheng's Dissertation [Yan13]. The authors main contribution with this joined work was the passive key exchange model and the $\mathsf{eBR}^{\mathsf{C}}$ model and the PKE-based compiler. Therefore, we present this compiler here again.

This chapter is structured as follows: We first give a short overview of related authenticated key exchange compilers in Section 6.1. In Section 6.2, we show collision probability for ephemeral public keys of a passively-secure key exchange protocol. The result is useful for the security analysis of our compilers. The corresponding definition of passively-secure key exchange and the security model are described in Section 3.3. In Section 6.3 we present a PKE based authenticated key exchange compiler, and close the chapter with a comparison of the efficiency of popular AKE compilers to our compilers in Section 6.4.

SUMMARY OF OUR CONTRIBUTIONS.

We present two efficient compilers that construct secure AKE systems from authentication protocols (AP) and passively secure key exchange protocols (KE). It relies on

signature schemes and only requires two additional moves in which signatures are exchanged. The second compiler relies on public key encryption systems. Although the first compiler is more efficient, the second compiler accounts for scenarios where the parties do not have (certified) signature keys but only encryption keys. This can often occur in practice. The security model $\mathsf{eBR}^{\mathsf{C}}$ described in Section3.4.3.1 is used for our security analysis of our AKE compilers.

- **Modular Design.** Our compilers only require the public transcript of the key exchange protocol, and do not require any modifications in the underlying (passive) KE protocols. Previous compilers as described in [BCK98, KY07, JKSS10] require costly modifications on the key exchange protocol. Namely, either the messages have to be modified or the secret session key K of the underlying KE must be input to the compiler. Thus, our compilers are easily applicable to existing systems, what makes them very useful in practice. Figure 6.1 shows modular construction of our compilers.

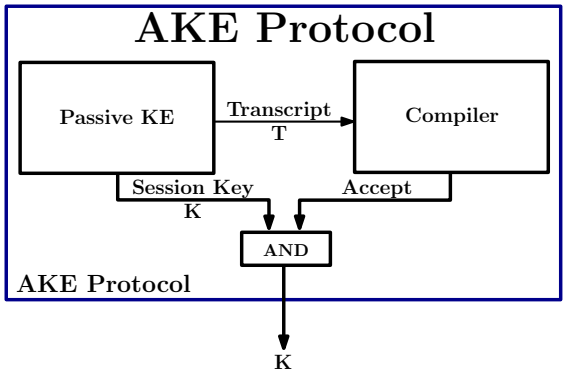


Fig. 6.1. Generic Construction of the AKE Compilers

- **Efficiency.** Our efficiency improvements rely on the following techniques:
  - We use a form of *implicit key confirmation* instead of *explicit key confirmation*.
  - As our second efficiency improvement, we formally show that for security we do not have to exchange uniformly random nonces after the key exchange protocol.
  - Finally, our compilers do not need to compute a new session key for AKE session. Our approach helps us to save the additional computation of a new session key for an AKE system.

6.1 Related Work of AKE Compilers

In 1998, Bellare, Canetti and Krawczyk were the first to consider a modular way for the development of AKE protocols [BCK98]. In this paper, they deal with two kinds

of network, i.e., a ideal authenticated network and a more realistic unauthenticated network. They propose to first design a protocol in the authenticated link model (AM) in which the adversary is limited to manage the delivery of the messages exchanged by the communication parties, and to corrupt some of them. But it can not inject, delete or manipulate messages. It implies that all messages generated by the party are authenticated. Then they transform a key exchange protocol proved secure in AM into an AKE protocol which is also secure in the unauthenticated link model (UM), in which the adversary has full powers on the communication channels, so it can change and forge new message, i.e., a real adversary environment. They used a so-called authenticator.

In 2007, Katz and Yung presented a generic compiler for group key agreement [KY07]. The KY compiler first adds an initial round to a passively secure group key exchange protocol, i.e., each party selects a nonce and broadcasts it to its communication partner. Then the compiler computes a signature for every message of the original group key exchange protocol and the random values that have been changed in the initial phase. In contrast to the BCK compiler, each message sent does not need to be authenticated interactively. The KY compiler only accounts for a single change-nonce phase that is added to the original protocol. However, the compiler still modifies each message sent in the original key exchange protocol by basically adding a signature to that message. It leads to the additional signature generation and verification operations, inefficient.

In 2010, Jäger et al. [JKSS10] also presented an elegant compiler in standard model which is independent of the key exchange protocol (a constant number of additional messages to be exchanged). Informally, the communication parties first exchange random nonces. Then, they compute signatures over the nonces and the transcript of key exchange protocol. Finally, two MAC values are computed over all the previous messages. JKSS10 compiler needs to use the session key from the passively secure key exchange in order to generate the corresponding MAC key.

6.2 Passively Secure Key Exchange Protocols

Passively-secure key exchange protocol (KE) is a central building block for our AKE compilers. In Section 3.3, we give a formal definition of KE protocol and a corresponding security model. In this section, we show collision probability for ephemeral public keys of a passively-secure key exchange protocol. The result is useful in the security proofs of our compilers to show that our compilers do not have to exchange additional random values after the passively-secure key exchange protocol run is finished.

6.2.1 Collision Property for Passively Secure KE Protocols

Our compilers require a passively-secure key exchange KE protocol. Before we now give the proof, we need to show that for every passively-secure key exchange protocol after polynomially calls to KE.EphemeralKeyGen there cannot be any collisions among

the ephemeral public keys generated by certain type of KE.EphemeralKeyGen. Note that the following lemma will be useful in the security proofs of our compilers to show that a compiler does not have to exchange additional random values after the KE run to guarantee that the transcripts which are authenticated with the authentication mechanism are unique. We can therefore discard the random values which are used in the JKSS compiler [JKSS10]. For simplicity reasons, we only describe a simple case, i.e., two-move and two-party passive KE scheme. Note that for a two-move and two-party ($\mathsf{PID_A}$ and $\mathsf{PID_B}$) KE-protocol there exist at most two types of KE.EphemeralKeyGen algorithms which may be determined by input messages $\mathsf{M_{in}^{PID_A}}$ and $\mathsf{M_{in}^{PID_B}}$. We here explicitly classify the algorithm KE.EphemeralKeyGen into two types denoted by $\mathsf{KE.EphemeralKeyGen_{PID_A}}$ for party $\mathsf{PID_A}$ and $\mathsf{KE.EphemeralKeyGen_{PID_B}}$ for party $\mathsf{PID_B}$.

Let collision denote the event that: after a polynomial number q of executions of KE.EphemeralKeyGen there exist at least two ephemeral public keys epk and epk^* generated by KE.EphemeralKeyGen are identical, where the number q is determined by time t_{KE}. Let $\epsilon_{\mathsf{collision}}$ denote the probability of the event collision occurred within time t_{KE}. We say all ephemeral keys generated by KE.EphemeralKeyGen are $(q, t_{\mathsf{KE}}, \epsilon_{\mathsf{collision}})$-distinct if those ephemeral keys are generated by KE.EphemeralKeyGen after q times execution of KE.EphemeralKeyGen within time t_{KE} and there exists no collision among those ephemeral keys except for probability $\epsilon_{\mathsf{collision}}$.

Lemma 6.1. *Assume* KE *is a* $(t_{\mathsf{KE}}, \epsilon_{\mathsf{KE}})$*-passively secure key exchange scheme without long-term key as defined above. Then all ephemeral public keys generated by* KE.EphemeralKeyGen *in the runs of* KE *scheme are* $(q, t_{\mathsf{KE}}, \epsilon_{\mathsf{collision}})$*-distinct such that* $\epsilon_{\mathsf{collision}} \leq q \cdot \epsilon_{\mathsf{KE}}$.

Proof. We consider the case that the ephemeral keys are generated by different types of ephemeral key algorithms. Obviously, in this case there is no collision (except negligible probability) between ephemeral keys $epk^s_{\mathsf{PID_A}}$ and $epk^t_{\mathsf{PID_B}}$, $s, t \in [q]$, because those keys are assumed to be generated from different key spaces. For simplicity, we say that $\epsilon_{\mathsf{collision}}$ is negligible.

We evaluate the collision probability $\epsilon_{\mathsf{collision}}$ for ephemeral keys generated by the same type of ephemeral key algorithms. We assume that with probability $\epsilon_{\mathsf{collision}}$ there will be a collision among the $epk^s_{\mathsf{PID_A}}$ (or $epk^t_{\mathsf{PID_B}}$), $s, t \in [q]$, after q protocol runs. According to the protocol specification, all values $epk^s_{\mathsf{PID_A}}$ are computed by randomized runs of $\mathsf{KE.EphemeralKeyGen_{PID_A}}$ while the $epk^t_{\mathsf{PID_B}}$ values have been computed by randomized runs of $\mathsf{KE.EphemeralKeyGen_{PID_B}}$. In particular, the computation of the $epk^s_{\mathsf{PID_A}}$ and $epk^t_{\mathsf{PID_B}}$ are deterministic in system parameters Π^{KE}, message $\mathsf{M_{in}}^{\mathsf{PID_A},s}$ (resp. $\mathsf{M_{in}}^{\mathsf{PID_B},t}$) and the random value $\omega^s_{\mathsf{PID_A}}$ (resp. $\omega^t_{\mathsf{PID_B}}$) used by $\mathsf{KE.EphemeralKeyGen_{PID_A}}$ (resp. $\mathsf{KE.EphemeralKeyGen_{PID_B}}$). The $\omega^s_{\mathsf{PID_A}}$ and $\omega^t_{\mathsf{PID_B}}$ are selected uniformly random and in particular independently.

Let $epk^*_{\mathsf{PID_A}}$ and $epk^*_{\mathsf{PID_B}}$ be the ephemeral public keys that are exchanged in the test session and given, together with the challenge key k^*_b and transcript T^*, to the ad-

versary. Let $esk^*_{\mathsf{PID_A}}$ and $esk^*_{\mathsf{PID_B}}$ be the corresponding ephemeral secret keys. These ephemeral public/secret keys can be computed using the ephemeral key algorithms $\mathsf{KE.EphemeralKeyGen}_{\mathsf{PID_A}}$ (resp. $\mathsf{KE.EphemeralKeyGen}_{\mathsf{PID_B}}$) with the public parameters Π^{KE}, the random value $\omega^*_{\mathsf{PID_A}}$ (resp. $\omega^*_{\mathsf{PID_B}}$) and $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A},*}$ (resp. $\mathsf{M}_{\mathsf{in}}^{\mathsf{PID_B},*}$). The adversary first guesses whether the collision occurs among the set of ephemeral public keys $\{epk_{\mathsf{PID_A}}\}$ or $\{epk_{\mathsf{PID_B}}\}$ with probability $\geq 1/2$. In the first case, the adversary can re-run $\mathsf{KE.EphemeralKeyGen}_{\mathsf{PID_A}}$ $(q-1)$ times with $\omega^s_{\mathsf{PID_A}}$ and $\mathsf{M_{in}}^{\mathsf{PID_A},s}$ to output $\{esk^s_{\mathsf{PID_A}}, epk^s_{\mathsf{PID_A}}\}$ for $s \in [1; q-1]$ in time less than t_{KE}. With the same probability $\epsilon_{\mathsf{collision}}$ it obtains two values $epk'_{\mathsf{PID_A}}, epk''_{\mathsf{PID_A}}$ among the q values $epk^*_{\mathsf{PID_A}}$, $epk^1_{\mathsf{PID_A}}$, …, $epk^{(q-1)}_{\mathsf{PID_A}}$ with $epk'_{\mathsf{PID_A}} = epk''_{\mathsf{PID_A}}$. Since it holds with probability $\geq 2/q$ that either $epk'_{\mathsf{PID_A}} = epk^*_{\mathsf{PID_A}}$ or $epk''_{\mathsf{PID_A}} = epk^*_{\mathsf{PID_A}}$. In this case the adversary knows one pair $(\omega'_{\mathsf{PID_A}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A},'})$ or $(\omega''_{\mathsf{PID_A}}, \mathsf{M}_{\mathsf{in}}^{\mathsf{PID_A},''})$ that maps to $epk^*_{\mathsf{PID_A}}$. Let $esk'_{\mathsf{PID_A}}$ be the corresponding ephemeral secret key. We now have to show that $esk'_{\mathsf{PID_A}}$ helps us to break the passive security. This simply follows from the determinism of $\mathsf{KE.SessionKeyGen}$ and correctness of KE. Since we have perfect correctness the adversary $\mathcal{A}$ can compute the session key k' by using the ephemeral secret key $esk'_{\mathsf{PID_A}}$ and transcript T^*. Next the adversary $\mathcal{A}$ can compare whether $k^*_b = k'$ and correctly guess the value b. In case there is a collision among the set $\{epk_{\mathsf{PID_B}}\}$ the situation is similar. Hence, due to the security of KE protocol, we have that the probability bound $\epsilon_{\mathsf{collision}} \leq q \cdot \epsilon_{\mathsf{KE}}$.

□

6.3 AKE Compiler from Public Key Encryption

In this section we will present a public key encryption based AKE compiler that turns passively-secure key exchange protocols as defined above to AKE protocols fulfilling the security guarantees as specified in Section 3.4. The compiler accounts for scenarios where the parties do not have (certified) signature keys but only encryption keys. This can often occur in practice. For example, the most efficient (for the client) and most wide-spread key exchange mechanism in TLS is RSA key transport. Here the server certificate only contains an RSA encryption key. In Section 6.3.1, we firstly give a description of our PKE-Based AKE compiler, and in Section 6.3.2 we analyzed its security in the $\mathsf{eBR}^{\mathsf{C}}$ model described in Section 3.4.3.1.

6.3.1 Description of PKE-Based AKE Compiler

The PKE-compiler takes the following building blocks as input: a passively-secure key exchange protocol KE, a public encryption scheme PKE, a collision resistant hash function CRHF and a one-time message authentication scheme OTM. During the initialization phase, the hash key is generated as $hk_{\mathsf{CRHF}} \xleftarrow{\$} \mathsf{CRHF.KG}(1^\kappa)$. Each party P is assumed to possess a pair of long-term private and public keys generated as

$(sk_{\mathsf{P}}, pk_{\mathsf{P}}) \xleftarrow{\$} \mathsf{PKE.KGen}(1^\kappa)$, $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$. In the sequel, we use the superscript "P" to highlight the message recored at party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$.

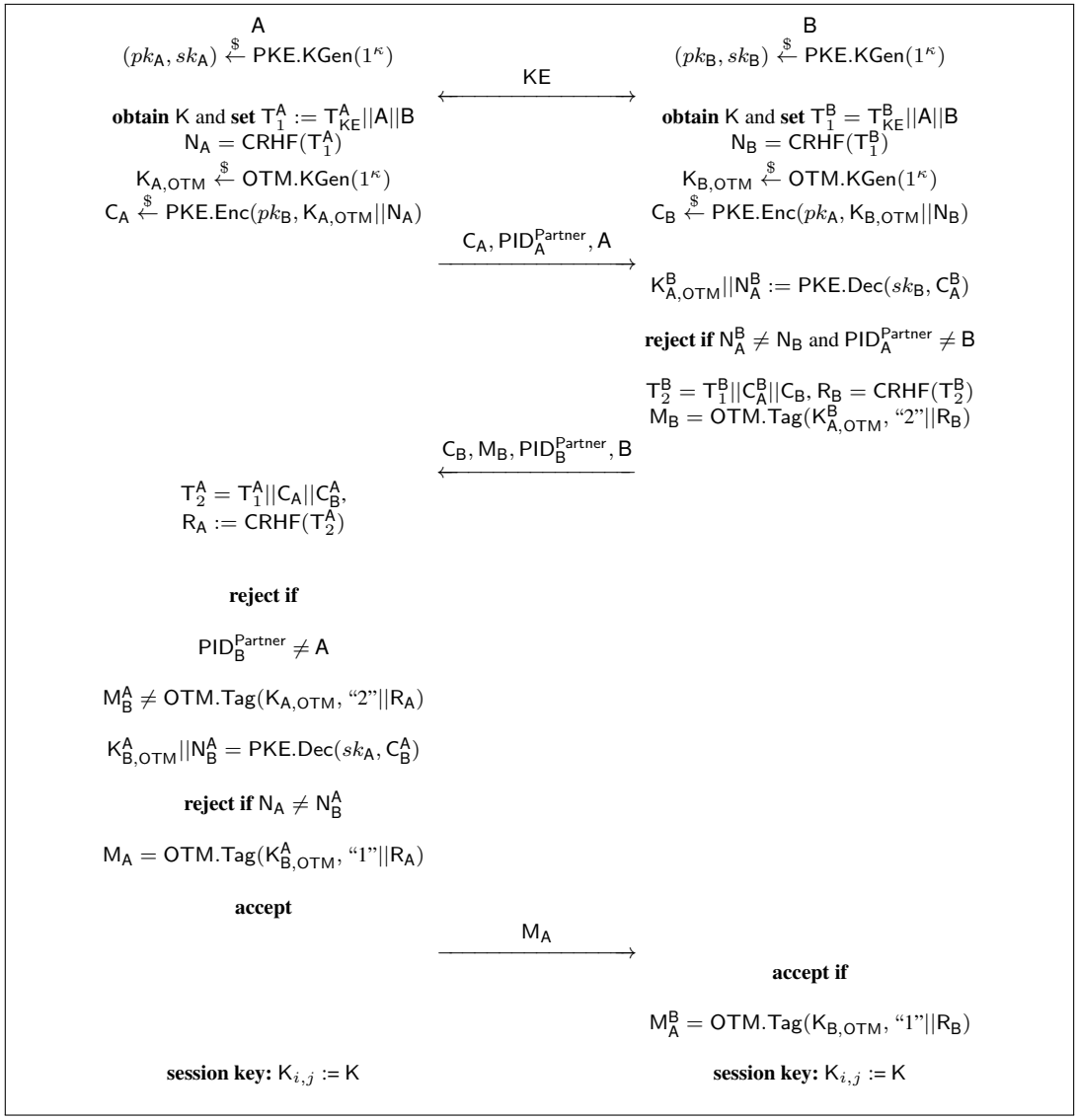


Fig. 6.2. AKE Compiler from PKE and OTM

Protocol Execution: The compiled protocol between two parties A and B proceeds as follows. It is also depicted in Figure 6.2.

- First, A and B run the passive key exchange protocol KE, then both parties obtain the key K from the key exchange phase (as the session key of AKE protocol) and record

the transcripts as $\mathsf{T}^{\mathsf{A}}_{\mathsf{KE}}$ and $\mathsf{T}^{\mathsf{B}}_{\mathsf{KE}}$, where $\mathsf{T}^{\mathsf{P}}_{\mathsf{KE}}$ consists of the list of all messages sent and received by party $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}\}$.

- A sets the transcript $\mathsf{T}^{\mathsf{A}}_1 := \mathsf{T}^{\mathsf{A}}_{\mathsf{KE}} \,||\, \mathsf{A} \,||\, \mathsf{B}$ and computes $\mathsf{N}_{\mathsf{A}} := \mathsf{CRHF}(\mathsf{T}^{\mathsf{A}}_1)$. Then, it runs $\mathsf{K}_{\mathsf{A,OTM}} \xleftarrow{\$} \mathsf{OTM.KGen}(1^\kappa)$ and computes a ciphertext $\mathsf{C}_{\mathsf{A}} \xleftarrow{\$} \mathsf{PKE.Enc}(pk_{\mathsf{B}}, \mathsf{K}_{\mathsf{A,OTM}} \,||\, \mathsf{N}_{\mathsf{A}})$ under B's public key pk_{B}. Finally, A defines the partner identity $\mathsf{PID}^{\mathsf{Partner}}_{\mathsf{A}} = \mathsf{B}$ and transmits $(\mathsf{C}_{\mathsf{A}}, \mathsf{PID}^{\mathsf{Partner}}_{\mathsf{A}}, \mathsf{A})$ to B.
- Meanwhile, B sets $\mathsf{T}^{\mathsf{B}}_1 := \mathsf{T}^{\mathsf{B}}_{\mathsf{KE}} \,||\, \mathsf{A} \,||\, \mathsf{B}$ and computes $\mathsf{N}_{\mathsf{B}} := \mathsf{CRHF}(\mathsf{T}^{\mathsf{B}}_1)$. It runs $\mathsf{K}_{\mathsf{B,OTM}} \xleftarrow{\$} \mathsf{OTM.KGen}(1^\kappa)$ and computes $\mathsf{C}_{\mathsf{B}} \xleftarrow{\$} \mathsf{PKE.Enc}(pk_{\mathsf{A}}, \mathsf{K}_{\mathsf{B,OTM}} \,||\, \mathsf{N}_{\mathsf{B}})$ under A's public key pk_{A}.
- Upon receiving the ciphertext $(\mathsf{C}^{\mathsf{B}}_{\mathsf{A}}, \mathsf{PID}^{\mathsf{Partner}}_{\mathsf{A}}, \mathsf{A})$, B sets $\mathsf{T}^{\mathsf{B}}_2 := \mathsf{T}^{\mathsf{B}}_1 \,||\, \mathsf{C}^{\mathsf{B}}_{\mathsf{A}} \,||\, \mathsf{C}_{\mathsf{B}}$ and computes $\mathsf{R}_{\mathsf{B}} := \mathsf{CRHF}(\mathsf{T}^{\mathsf{B}}_2)$. It decrypts $\mathsf{C}^{\mathsf{B}}_{\mathsf{A}}$ (i.e., $\mathsf{K}^{\mathsf{B}}_{\mathsf{A,OTM}} \,||\, \mathsf{N}^{\mathsf{B}}_{\mathsf{A}} := \mathsf{PKE.Dec}(sk_{\mathsf{B}}, \mathsf{C}^{\mathsf{B}}_{\mathsf{A}})$). Then B checks whether $\mathsf{PID}^{\mathsf{Partner}}_{\mathsf{A}} = \mathsf{B}$ and $\mathsf{N}^{\mathsf{B}}_{\mathsf{A}} = \mathsf{N}_{\mathsf{B}}$. If the check is not passed, then B rejects. Otherwise, it sets the partner identity $\mathsf{PID}^{\mathsf{Partner}}_{\mathsf{B}} = \mathsf{A}$ and computes $\mathsf{M}_{\mathsf{B}} := \mathsf{OTM.Tag}(\mathsf{K}^{\mathsf{B}}_{\mathsf{A,OTM}}, \text{“2”} \,||\, \mathsf{R}_{\mathsf{B}})$ and transmits $(\mathsf{C}_{\mathsf{B}}, \mathsf{M}_{\mathsf{B}}, \mathsf{PID}^{\mathsf{Partner}}_{\mathsf{B}}, \mathsf{B})$ to A.
- Upon receiving messages $(\mathsf{M}^{\mathsf{A}}_{\mathsf{B}}, \mathsf{C}^{\mathsf{A}}_{\mathsf{B}}, \mathsf{PID}^{\mathsf{Partner}}_{\mathsf{B}}, \mathsf{B})$, A sets $\mathsf{T}^{\mathsf{A}}_2 := \mathsf{T}^{\mathsf{A}}_1 \,||\, \mathsf{C}_{\mathsf{A}} \,||\, \mathsf{C}^{\mathsf{A}}_{\mathsf{B}}$ and computes $\mathsf{R}_{\mathsf{A}} := \mathsf{CRHF}(\mathsf{T}^{\mathsf{A}}_2)$. A rejects if $\mathsf{M}^{\mathsf{A}}_{\mathsf{B}} \neq \mathsf{OTM.Tag}(\mathsf{K}_{\mathsf{A,OTM}}, \text{“2”} \,||\, \mathsf{R}_{\mathsf{A}})$ and $\mathsf{PID}^{\mathsf{Partner}}_{\mathsf{B}} \neq \mathsf{A}$. If the check is passed, then it decrypts the ciphertext $\mathsf{C}^{\mathsf{A}}_{\mathsf{B}}$ (i.e., $\mathsf{K}^{\mathsf{A}}_{\mathsf{B,OTM}} \,||\, \mathsf{N}^{\mathsf{A}}_{\mathsf{B}} := \mathsf{PKE.Dec}(sk_{\mathsf{B}}, \mathsf{C}^{\mathsf{A}}_{\mathsf{B}})$) and checks whether $\mathsf{N}_{\mathsf{A}} = \mathsf{N}^{\mathsf{A}}_{\mathsf{B}}$. If the check is not passed, then A rejects. Otherwise, A computes $\mathsf{M}_{\mathsf{A}} := \mathsf{OTM.Tag}(\mathsf{K}^{\mathsf{A}}_{\mathsf{B,OTM}}, \text{“1”} \,||\, \mathsf{R}_{\mathsf{A}})$, and sends M_{A} to B. Finally, A accepts the session and outputs the session key $\mathsf{K}_{i,j} := \mathsf{K}$.
- Upon receiving $\mathsf{M}^{\mathsf{B}}_{\mathsf{A}}$, B accepts if and only if $\mathsf{M}^{\mathsf{B}}_{\mathsf{A}} = \mathsf{OTM.Tag}(\mathsf{K}_{\mathsf{B,OTM}}, \text{“1”}||\mathsf{R}_{\mathsf{B}})$, and outputs the session key $\mathsf{K}_{i,j} := \mathsf{K}$.

Session States: We assume that the session state of a session owned by A contains ephemeral secret keys esk_{A} used in each KE protocol instance and random one-time MAC-key $\mathsf{K}_{\mathsf{A,OTM}}$. The intermediate values (esk_{A} and $\mathsf{K}_{\mathsf{A,OTM}}$) will be stored in the variable State. Similarly, the session state of a session owned by B contains ephemeral secret keys esk_{B} and random one-time MAC-key $\mathsf{K}_{\mathsf{B,OTM}}$.

6.3.2 Security Analysis of PKE-Based AKE Compiler

In this section we prove that the proposed public key encryption based authenticated key exchange compiler is secure in the sense of the security guarantees as specified in Section 3.4 in the standard model.

Theorem 6.2. *Assume that the key exchange protocol* KE *without long-term key is* $(t, \epsilon_{\mathsf{KE}})$*-secure with respect to Definition 6.2, the public key encryption scheme is* $(q_{\mathsf{PKE}}, t, \epsilon_{\mathsf{PKE}})$*-secure (IND-CCA2) with respect to Definition 2.2.4, the hash function is* $(t, \epsilon_{\mathsf{CRHF}})$*-secure with respect to Definition 2.2.1.1 and the one-time message authentication code scheme is* $(1, t, \epsilon_{\mathsf{OTM}})$*-secure with respect to Definition 2.2.3.1. Then the*

above protocol is a $(t', \epsilon_{\mathsf{AKE}})$*-secure authenticated key exchange protocol* AKE *in the sense of Definition 3.7 with* $t' \approx t$ *and*

$$\epsilon_{\mathsf{AKE}} \leq \frac{(d\ell)^2}{2^{(\lambda-1)}} + 2d\ell\epsilon_{\mathsf{KE}} + d\ell^2(2\epsilon_{\mathsf{CRHF}} + 2\epsilon_{\mathsf{PKE}} + 2\epsilon_{\mathsf{OTM}}) + (d\ell)^2\epsilon_{\mathsf{KE}}.$$

We prove Theorem 6.2 in two stages. First, we show that the AKE protocol is a secure authentication protocol except for probability ϵ_{Auth}, that is, the protocol fulfills security property 1.) of the AKE definition 3.7. In the next step, we show that the session key of the AKE protocol is secure except for probability ϵ_{Ind} in the sense of the Property 2.) of the AKE definition 3.7. Then we have the overall probability ϵ_{AKE} that an adversary breaking the protocol is at most $\epsilon_{\mathsf{AKE}} \leq \epsilon_{\mathsf{Auth}} + \epsilon_{\mathsf{Ind}}$. To prove the following lemmas, we proceed in games as in [Sho04, BR06].

Let $\mathcal{A}_{\mathsf{AKE}}$ be an adversary against the security of the AKE compiler scheme. We consider a sequence of games and each game involving an adversary $\mathcal{A}_{\mathsf{AKE}}$ and a challenger $\mathcal{C}_{\mathsf{AKE}}$. Then, we modify Game 0 (i.e., a real security experiment) step-by-step, and argue that each game is computationally indistinguishable from the previous one. Roughly speaking, if an adversary can distinguish Game (i-1) from Game i with a non-negligible advantage, then we can use it to construct a (polynomial-time) algorithm to break some complexity assumptions. Finally, we argue that any adversary cannot win the real security experiment denoted by Game 0 with non-negligible probability.

6.3.2.1 Authentication Property

Lemma 6.3. *For any adversary* $\mathcal{A}$ *running in time* $t' \approx t$*, the probability that there exists an oracle* π_i^s *that accepts maliciously in the sense of Definition 3.7 is at most*

$$\epsilon_{\mathsf{Auth}} \leq \frac{(d\ell)^2}{2^{\lambda}} + d\ell\epsilon_{\mathsf{KE}} + d\ell^2(\epsilon_{\mathsf{CRHF}} + \epsilon_{\mathsf{PKE}} + \epsilon_{\mathsf{OTM}}),$$

where all quantities are as the same as stated in the Theorem 6.2.

Proof. Let $\mathsf{break}_{\delta}^{(\mathsf{Auth})}$ be the event that there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has internal state $\Phi_{i^*}^{s^*} = \mathtt{accept}$ and $\mathsf{PID}_{i^*}^{s^*} = j^*$, but there is no unique oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations, in Game δ.

GAME 0. This is the original security game. Thus, we have that

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] = \epsilon_{\mathsf{Auth}}.$$

GAME 1. In this game, the challenger proceeds exactly like the challenger in Game 0, except that we add an abortion rule. The challenger raises event $\mathsf{abort}_{\mathsf{ephemeral}}$ and aborts, if two oracles output the same ephemeral keys. Thus, the event $\mathsf{abort}_{\mathsf{ephemeral}}$ occurs with

probability $\Pr[\mathsf{abort}_{\mathsf{ephemeral}}] \leq d\ell\epsilon_{\mathsf{KE}}$ by Lemma 6.1. From the result of Lemma 6.1, we know that the collision probability among ephemeral keys is related to the polynomial number of execution of KE.EphemeralKeyGen and the probability ϵ_{KE}. Since there are $d\ell$ oracles, therefore the event $\mathsf{abort}_{\mathsf{eph}}$ occurs with probability $\Pr[\mathsf{abort}_{\mathsf{eph}}] \leq d\ell \cdot \epsilon_{\mathsf{KE}}$. We have that

$$\Pr[\mathsf{break}_0^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_1^{(\mathsf{Auth})}] + d\ell \cdot \epsilon_{\mathsf{KE}}.$$

GAME 2. In this game, the challenger $\mathcal{C}_{\mathsf{AKE}}$ proceeds exactly like the challenger in Game 1, except that we add an abortion rule $\mathsf{abort}_{\mathsf{collision}}^{\mathsf{OTM}}$. We let $\mathsf{abort}_{\mathsf{collision}}^{\mathsf{OTM}}$ be the event that two oracles sample the same one-time MAC key $\mathsf{K}_{\mathsf{OTM}}$ for the one-time message authentication code as described in Section 2.6. Assume that each one-time MAC key $\mathsf{K}_{\mathsf{OTM}}$ from $\{0,1\}^\lambda$. Thus, the probability that a collision occurs is bounded by $(\frac{(d\ell)^2}{2^\lambda})$. Clearly, we have

$$\Pr[\mathsf{break}_1^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_2^{(\mathsf{Auth})}] + \frac{(d\ell)^2}{2^\lambda}.$$

GAME 3. This game proceeds exactly as before, but the challenger aborts if it fails to guess the first fresh oracle $\pi_{i^*}^{s^*}$ and its intended partner PID_{j^*} which accepts without matching conversation, where $(i^*, j^*) \in [\ell]^2$ and $s^* \in [d]$. According to the definition described in 3.4.2, we allow the adversary to register a new party P_c, with a valid public key pk_c on behalf of P_c [1]. However, note that by our freshness definition $\pi_{i*}^{s^*}$ is fresh and thus PID_{i^*} and PID_{j^*} must not be adversarially controlled. Thus, we have that

$$\Pr[\mathsf{break}_2^{(\mathsf{Auth})}] \leq d\ell^2 \cdot \Pr[\mathsf{break}_3^{(\mathsf{Auth})}].$$

GAME 4. This game proceeds as the previous game, except that we add an abort condition $\mathsf{abort}_{\mathsf{collision}}^{\mathsf{CRHF}}$ and the challenger aborts if there are two distinct inputs to the CRHF that map to the same output value. Obviously we have the $\Pr[\mathsf{abort}_{\mathsf{collision}}^{\mathsf{CRHF}}] \leq \epsilon_{\mathsf{CRHF}}$ and

$$\Pr[\mathsf{break}_3^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_4^{(\mathsf{Auth})}] + \epsilon_{\mathsf{CRHF}}.$$

GAME 5. This game proceeds exactly as before, but instead of encrypting the message $(\mathsf{K}_{i^*,\mathsf{OTM}}^{s^*} \parallel \mathsf{N}_{i^*}^{s^*})$, we encrypt a random message R^* for PKE scheme[2].

We apply the same modification to the oracle $\pi_{j^*}^{t^*}$ which shares the same transcript T_1 with oracle $\pi_{i^*}^{s^*}$. The party PID_{j^*} is uncorrupted and there is no collision among the hashed transcripts (due to the previous games). If there exists an adversary $\mathcal{A}_{\mathsf{AKE}}$ who can distinguishes this game from the previous game, we can use it to construct an algorithm $\mathcal{B}_{\mathsf{PKE}}$ to break the security of the PKE scheme as described in 2.2.4.

[1] Parties established by this query are called adversarially-controlled.

[2] Note that we still use the one-time MAC key to compute the MAC value as the previous game. However, it will be independent of the ciphertext of PKE.

Assuming that an adversary $\mathcal{B}_{\mathsf{PKE}}$ using its encryption oracle $\mathcal{O}^{\mathsf{ENC}}$ and decryption oracle $\mathcal{O}^{\mathsf{DEC}}$ interacts with the challenger $\mathcal{C}_{\mathsf{PKE}}$. $\mathcal{B}_{\mathsf{PKE}}$ acts as a challenger for $\mathcal{A}_{\mathsf{AKE}}$ in this game. We show that the simulation of $\mathcal{B}_{\mathsf{PKE}}$ perfectly simulates the challenger $\mathcal{C}_{\mathsf{AKE}}$ in this game from the adversary's point of view.

At the beginning of the simulation game, $\mathcal{B}_{\mathsf{PKE}}$ obtains the public parameters and the public key pk' of the challenger $\mathcal{C}_{\mathsf{PKE}}$. Then using the identity PID_{j^*} (as the intended partner of the test oracle $\pi_{i^*}^{s^*}$) it sets $pk_{j^*} = pk'$. Moreover, $\mathcal{B}_{\mathsf{PKE}}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, and honestly computes all other long-term public/private key pairs $(pk_{\mathsf{PID}_i}, sk_{\mathsf{PID}_i}) \xleftarrow{\$} \mathsf{PKE.KGen}$ for each honest party PID_i, $i \in [\ell]$ and $i \neq j^*$. Simultaneously, it also generates one-time MAC keys as the previous game. $\mathcal{A}_{\mathsf{AKE}}$ receives the public parameters and all long-term public keys $\{ pk_{\mathsf{PID}_1}, \ldots, pk_{\mathsf{PID}_j^*}, \ldots, pk_{\mathsf{PID}_\ell} \}$ as input. Obviously, $\mathcal{B}_{\mathsf{KE}}$ can honestly answer all oracle queries supplied by $\mathcal{A}_{\mathsf{AKE}}$. Meanwhile, $\mathcal{B}_{\mathsf{PKE}}$ generates the challenge ciphertext $C_{i^*}^{s^*}$ with the transcripts $(\mathsf{K}_{i^*}^{s^*} || \mathsf{N}_{i^*}^{s^*}, \mathsf{R}^*)$. The challenger $\mathcal{C}_{\mathsf{PKE}}$ responds with the ciphertext $C_{i^*}^{s^*}$ to $\mathcal{B}_{\mathsf{PKE}}$. For other oracles of the party PID_{j^*}, $\mathcal{B}_{\mathsf{PKE}}$ can use its decryption oracle to simulate the protocol executing. Obviously, this is a perfect simulation of $\mathcal{C}_{\mathsf{PKE}}$. $\mathcal{B}_{\mathsf{PKE}}$ can always correctly answer all queries made by $\mathcal{A}_{\mathsf{AKE}}$. If $\mathcal{A}_{\mathsf{AKE}}$ can correctly distinguish this game from the previous game, $\mathcal{B}_{\mathsf{PKE}}$ can break the security of the PKE scheme as described in 2.2.4. Due to the security of the PKE scheme, the advantage of $\mathcal{A}_{\mathsf{AKE}}$ in distinguishing between this game and the previous game is bound by ϵ_{PKE}. Thus, we have that

$$\Pr[\mathsf{break}_4^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_5^{(\mathsf{Auth})}] + \epsilon_{\mathsf{PKE}}.$$

GAME 6. This game proceeds exactly like the previous game except that the challenger aborts if the fresh (test) oracle $\pi_{i^*}^{s^*}$ accepts a confirmation MAC message $\mathsf{M}_{j^*}^{t^*}$ but it has not been sent by any oracle of its intended partner PID_{j^*}. The key $\widetilde{K}_{i^*}^{s^*}$ of the OTM in computation of $\pi_{i^*}^{s^*}$ is a random value which is independent of the ciphertext $C_{i^*}^{s^*}$ and only would be used once at most to authenticate the transcript $\mathsf{T}_2^{(i^*,s^*)}$. Applying the security of OTM we have that

$$\Pr[\mathsf{break}_5^{(\mathsf{Auth})}] \leq \Pr[\mathsf{break}_6^{(\mathsf{Auth})}] + \epsilon_{\mathsf{OTM}}.$$

Claim 6.4. $\Pr[\mathsf{break}_6^{(\mathsf{Auth})}] = 0$

Proof. Note that $\mathsf{break}_6^{(\mathsf{Auth})}$ occurs only if there exists a τ and a τ-fresh oracle $\pi_{i^*}^{s^*}$ that has internal state $\Phi_{i^*}^{s^*} = \mathtt{accept}$ and $\mathsf{PID}_{i^*}^{s^*} = j^*$, but there is no *unique* oracle $\pi_{j^*}^{t^*}$ such that $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ have matching conversations. Due to the previous games there exists (only one) oracle $\pi_{j^*}^{t^*}$ which has output the confirmation one-time MAC message $\mathsf{M}_{j^*}^{t^*}$ received by oracle $\pi_{i^*}^{s^*}$, otherwise the execution will be aborted. Thus, if oracle $\pi_{i^*}^{s^*}$ accepts, then it must have a matching conversation to oracle $\pi_{j^*}^{t^*}$. Therefore we have

$$\Pr[\mathsf{break}_6^{(\mathsf{Auth})}] = 0.$$

□

Summing up the probabilities from Game 0 to Game 6, we proved Lemma 6.3, i.e.,

$$\epsilon_{\mathsf{Auth}} \leq \frac{(d\ell)^2}{2^\lambda} + d\ell\epsilon_{\mathsf{KE}} + d\ell^2(\epsilon_{\mathsf{CRHF}} + \epsilon_{\mathsf{PKE}} + \epsilon_{\mathsf{OTM}}).$$

□

6.3.2.2 Key Indistinguishability

Lemma 6.5. *For any adversary $\mathcal{A}$ running in time $t' \approx t$, the probability that $\mathcal{A}$ correctly answers the* Test*-query is at most $1/2 + \epsilon_{\mathsf{Ind}}$ with*

$$\epsilon_{\mathsf{Ind}} \leq \frac{(d\ell)^2}{2^\lambda} + d\ell\epsilon_{\mathsf{KE}} + d\ell^2(\epsilon_{\mathsf{CRHF}} + \epsilon_{\mathsf{PKE}} + \epsilon_{\mathsf{OTM}}) + (d\ell)^2\epsilon_{\mathsf{KE}},$$

where all quantities are as the same as stated in the Theorem 6.2.

Proof. Let $\mathsf{break}_\delta^{(\mathsf{Ind})}$ denote the event that the adversary $\mathcal{A}_{\mathsf{AKE}}$ correctly guesses the bit b' sampled by the Test-query in Game δ, and $\mathsf{Test}(\pi_{i*}^{s^*})$ is the τ-th query of $\mathcal{A}_{\mathsf{AKE}}$, and $\pi_{i*}^{s^*}$ is a τ-fresh oracle that is ∞-revealed throughout the security game. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{break}_\delta^{(\mathsf{Ind})}] - 1/2$ be the advantage of $\mathcal{A}_{\mathsf{AKE}}$ in Game δ. The proof proceeds using a sequence of games including steps similar to [Sho04, BR06].

GAME 0. This is the original security game. Thus, we have that

$$\Pr[\mathsf{break}_0^{(\mathsf{Ind})}] = \epsilon_{\mathsf{Ind}} + 1/2 = \mathsf{Adv}_0 + 1/2.$$

GAME 1. The challenger in this game proceeds as before, but it aborts if the test oracle accepts without unique partner oracle. Applying the security of authentication property of this protocol, we thus have

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \epsilon_{\mathsf{auth}}.$$

GAME 2. This game is similar to the previous game. However, the challenger $\mathcal{C}_{\mathsf{AKE}}$ now guesses the partner oracle $\pi_{j*}^{t^*}$ that stays fresh and participates with $\pi_{i*}^{s^*}$ in the test session. $\mathcal{C}_{\mathsf{AKE}}$ aborts if its guess is not correct. Thus, we have

$$\mathsf{Adv}_1 \leq (d\ell)^2\mathsf{Adv}_2.$$

GAME 3. Finally, we replace the session key k^* of the test oracle $\pi_{i*}^{s^*}$ and its partner oracle $\pi_{j*}^{t^*}$ with the random value $\widetilde{k^*}$. If there exists an adversary $\mathcal{A}_{\mathsf{AKE}}$ who can distinguish this game from the previous game, then we use it to construct an algorithm $\mathcal{B}_{\mathsf{KE}}$ to break the passive security of the key exchange protocol described in Section 3.3.

Assuming that an adversary $\mathcal{B}_{\mathsf{KE}}$ using its Execute and EphemeralKeyReveal oracles interacts with $\mathcal{C}_{\mathsf{KE}}^{\mathsf{Passive}}$ that is a challenger in the security game of the passively secure KE scheme as in Section 3.3.2. Then, $\mathcal{B}_{\mathsf{KE}}$ acts as a challenger for $\mathcal{A}_{\mathsf{AKE}}$ in Game 3. We show that the simulation of $\mathcal{B}_{\mathsf{KE}}$ perfectly simulates the challenger $\mathcal{C}_{\mathsf{AKE}}$ in this game from the adversary's point of view, i.e., (i) showing how $\mathcal{B}_{\mathsf{KE}}$ simulates the protocol execution environment; (ii) showing how $\mathcal{B}_{\mathsf{KE}}$ can perfectly answer to all queries issued by $\mathcal{A}_{\mathsf{AKE}}$ and (iii) showing that if the adversary $\mathcal{A}_{\mathsf{AKE}}$ can correctly output a bit b' to answer the Test-query, $\mathcal{B}_{\mathsf{KE}}$ can use it to break the security of KE. We describe this simulation as follows:

- At the beginning of the simulation game, $\mathcal{B}_{\mathsf{KE}}$ implements a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [d]\}$, and honestly computes all long-term verification/private key pairs $(pk_{\mathsf{PID}_i}, sk_{\mathsf{PID}_i}) \xleftarrow{\$} \mathsf{PKE.KGen}$ for each honest party PID_i, $i \in [\ell]$, by calling the key generation algorithm PKE.KGen. Similarly, it also generates one-time MAC keys as before. The adversary $\mathcal{A}_{\mathsf{AKE}}$ receives all honestly generated long-term public keys $\{ pk_{\mathsf{PID}_1}, \ldots, pk_{\mathsf{PID}_\ell} \}$ as input.
- Then, $\mathcal{B}_{\mathsf{KE}}$ has to guess test oracle $\pi_{i*}^{s^*}$ and its partner oracle $\pi_{j*}^{t^*}$ supplied by $\mathcal{A}_{\mathsf{AKE}}$. $\mathcal{B}_{\mathsf{KE}}$ can answer all oracle queries honestly except for the test oracle $\pi_{i*}^{s^*}$ and its partner oracle $\pi_{j*}^{t^*}$ as follows: In response to Corrupt queries made by $\mathcal{A}$, $\mathcal{B}_{\mathsf{KE}}$ returns the contents of long-term private keys to $\mathcal{A}_{\mathsf{AKE}}$. Further, $\mathcal{B}_{\mathsf{KE}}$ can answer to Reveal and RevealState queries made by $\mathcal{A}$ by just forwarding them to the $\mathcal{C}_{\mathsf{KE}}$ challenger and using the MAC keys. Note that this strategy works only if $\mathcal{B}_{\mathsf{KE}}$ guesses the test oracle $\pi_{i*}^{s^*}$ and its partner oracle $\pi_{j^*}^{t^*}$.
- For Test query made by $\mathcal{A}_{\mathsf{AKE}}$, $\mathcal{B}_{\mathsf{KE}}$ can first query $\mathcal{C}_{\mathsf{KE}}$ for executing a fresh test instance and obtains $(\mathsf{K}_b, \mathsf{T}_{\mathsf{KE}})$ from $\mathcal{C}_{\mathsf{KE}}$. Then, $\mathcal{B}_{\mathsf{KE}}$ simulates the test oracle which accept using the transcript T_{KE} and returns K_b to $\mathcal{A}$. After that $\mathcal{A}$ may continually perform the queries.
- Finally, $\mathcal{A}_{\mathsf{AKE}}$ terminates with outputting b'. $\mathcal{B}_{\mathsf{KE}}$ returns b' supplied by $\mathcal{A}_{\mathsf{AKE}}$ to $\mathcal{C}_{\mathsf{KE}}$.

Obviously, this is a perfect simulation of $\mathcal{B}_{\mathsf{KE}}$. $\mathcal{B}_{\mathsf{KE}}$ can always correctly answer all queries made by $\mathcal{A}_{\mathsf{AKE}}$. If $\mathcal{A}_{\mathsf{AKE}}$ correctly guesses the value b used to answer the Test query, $\mathcal{B}_{\mathsf{KE}}$ can break the passive security of the KE scheme. Exploiting the security of passively-secure key exchange protocol KE, we obtain that

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{KE}}.$$

Further, in Game 3 the adversary always receives a uniformly random session key in response to Test query, i.e it receives no information about b used to answer the Test query. Thus, we have $\mathsf{Adv}_3 = 0$.

Finally, we obtained the advantage of $\mathcal{A}_{\mathsf{AKE}}$ shown in Lemma 6.5 by putting together all probabilities from Game 0 to Game 3, i.e.,

$$\epsilon_{\mathsf{Ind}} \leq \frac{(d\ell)^2}{2^{\lambda}} + d\ell\epsilon_{\mathsf{KE}} + d\ell^2(\epsilon_{\mathsf{CRHF}} + \epsilon_{\mathsf{PKE}} + \epsilon_{\mathsf{OTM}}) + (d\ell)^2\epsilon_{\mathsf{KE}}.$$

Collecting probabilities from Lemma 6.3 and Lemma 6.5, we conclude that

$$\epsilon_{\mathsf{AKE}} \leq \frac{(d\ell)^2}{2^{(\lambda-1)}} + 2d\ell\epsilon_{\mathsf{KE}} + d\ell^2(2\epsilon_{\mathsf{CRHF}} + 2\epsilon_{\mathsf{PKE}} + 2\epsilon_{\mathsf{OTM}}) + (d\ell)^2\epsilon_{\mathsf{KE}}.$$

This is the same advantage as in Theorem 6.2 and hence Theorem 6.2 follows. This completes the proof of the theorem. □

6.4 Efficiency Comparison with other popular Compilers

In this section we compare the efficiency of other popular compiler described in [BCK98, KY07, JKSS10] to our compilers and summarize the differences in Figure 6.3. The symbol "•" means that the model captures this property. Otherwise, "×".

Properties / Compilers	Standard Model	PFS	KCI	Reveal State	Implicit Key Confirmation	Explicit-Key Confirmation	Exchange of Nonces	Compute New Session-Key	Additional Communication Complexity	Additional Computation Complexity
BCK98	•	•	×	×	•	×	•	×	$\mathcal{O}(\|m\|)$	$\mathcal{O}(\|m\|)$
KY07	•	•	×	×	•	×	•	×	$\mathcal{O}(1)$	$\mathcal{O}(\|m\|)$
JKSS10	•	•	•	×	×	•	•	•	$\mathcal{O}(1)$	$\mathcal{O}(1)$
BLSSY14	•	•	×	•	•	×	×	×	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Fig. 6.3. Efficiency Comparison of other popular Compilers to our Compilers

Remark 6.6. The symbol "$|m|$" is the number of messages of key exchange protocol. In the BCK compiler, for every message from original KE protocol there exists some additional communication with the receiver $\mathsf{P_R}$ in which $\mathsf{P_R}$ first sends a random nonce to the sender $\mathsf{P_S}$, and $\mathsf{P_S}$ responds with an application of an authentication mechanism (e.g. a digital signature scheme or public key crypto-system) on the message with this nonce. It leads to an additional communication complexity $\mathcal{O}(|m|)$. The KY07 compiler also adds to every message of the original key exchange protocol a signature which is also computed over all the random values that have been generated in the exchange of nonces phase. In contrast to BCK98 compiler, each message sent does not need to be authenticated interactively. However, the compiler still modifies each message sent in the original key exchange protocol by basically adding a signature to that message. It leads to the additional signature generation and verification operations, inefficient,

additional time complexity $\mathcal{O}(|m|)$. However, JKSS10 and our compilers account only for a constant number of additional messages to be exchanged, i.e., it is independent of the original key exchange protocol. It only leads to the additional computation and communication complexity $\mathcal{O}(1)$.

Chapter 7

SessionIDs in Key Exchange Protocols and No-Match Attacks

Contents

In this chapter, we present a theoretical attack for key exchange protocols, named here *no-match attacks*, and show that proving security under the matching conversations as session IDs (MC-based sID) is a delicate issue. In particular, we provide several examples of protocols that claim to be secure under a security definition based on MC-based sID but where the security proof is actually flawed. We show that no-match attacks are often hard to avoid without costly modifications of the original protocol, moreover, give several ways to thwart no-match attacks. The content of this chapter was brought forth in a cooperation with Sven Schäge and Zheng Yang.

SUMMARY OF OUR CONTRIBUTIONS. We provide evidence that providing sound security analyzes for key exchange protocols is difficult and error-prone. Simultaneously, we present a new theoretical attack for key exchange protocols, named here *no-match* attacks, and show that proving security under the matching conversations as session IDs (MC-based sID) is a delicate issue. In particular, we show that the security proofs of several existing (and recent) security protocols are flawed. Finally, we provide several ways to thwart no-match attacks.

ORGANIZATION. This chapter is structured as follows: First, we informally describe the most widely used session identifiers (sID) definitions and give a discussion on draw-

backs of these definitions in Section 7.1. In Section 7.2 we present *no-match* attacks, and provide several examples of protocols that claim to be secure under a security definition based on MC-based sID but where the security proof is actually flawed. Finally, we discuss several ways to thwart no-match attacks in Section 7.3.

7.1 Discussion on Three Ways to Define Session IDs

It is well known that the security definition depends on the notions of partnership of oracles. The most recent definition of partnership is based on session identities, sIDs[1]. There are three most widely used ways to define session identity.

7.1.1 Pre-specified Unique Session IDs

The first one is based on (pre-specified) unique session IDs which are *externally* given to the oracles of a session at protocol start-up [CK01, CK02b] [2]. When analyzing practical protocols this approach is often problematic. Most of the important protocols do not feature a special session ID generation phase. Thus, to make them provable in any security model that is based on external session IDs one actually would have to run an additional protocol (or rely on a trusted party) to generate the session identifier before the protocol starts (and hand it over to the original protocol). This is, however, not what is done in practice and requires considerable changes to protocols.

7.1.2 Session IDs based on partial Transcripts

Secondly, the session identifier sID is based on *partial* transcripts from the exchanged messages. In some security models for key exchange the exact definition of session ID is not further specified. In fact, its form may even be protocol-dependent. Usually such a session ID is then defined when analyzing concrete protocols as a truncated or a partial transcript of the messages exchanged between the communication partners [KPW13, BDK+14]. This means that not all the messages are part of the session ID, in contrast to the definition of MC-based sID. We believe that when using partial transcripts, there should at least be an argumentation for the choice of messages that are included in the definition of session ID. Or to put it another way, authors should explain why they deviate from matching conversations and why their choice of session ID does not lead to weak security results.

[1] Note that the BR93 Model [BR93a] defines partnership using the notion of *matching conversation*. Partnership in the BR95 [BR95] model is defined using the notion of a *partner function*, but in there is no generic definition of partner function fixed for any key exchange protocol.

[2] Note that for more information on *external* session identifier, see [CK01, CK02b].

Naturally, the definition of which messages have to be part of the session ID is ***highly protocol-dependent***. In other words, an inadvisable selection of (partial-transcripts-based) sID can lead to disastrous effects for security analysis. Let us highlight this in a brief example. For example, imagine two protocols, the first of which consists of an two-move exchange of signed DH shares followed by the two-move exchange of two unprotected nonces. The second protocol is similar except that the order of exchange of DH shares and nonces is reversed, i.e. first the parties exchange nonces, then the signed DH-shares. In both cases the session key is computed as the DH key of the two DH shares if the verification of the received signature succeeds. Now assume, the ***session ID*** is defined to be the ***partial transcript*** of the ***first two moves*** of the protocol. In spite of their similarities both protocols have totally different security properties in the security model. In the first protocol an adversary can simply alter one of the nonces exchanged, say the one sent from Alice's oracle to Bob's. Now, according to the definition both oracles do not have the same session ID, i.e. they are unrelated. This allows the adversary to query Bob's oracle for the session key. If the adversary now chooses Alice's oracle to be the Test-oracle it can simply use Bob's key to trivially break key indistinguishability. This attack however, cannot be launched against the second protocol. Intuitively, this is because whenever the adversary modifies the DH shares, it breaks the security of the underlying signature scheme or replay values that have previously been generated by any of the parties. However, in the latter case the keys computed by Alice and Bob will be derived from different DH shares that all have been drawn honestly and independently (by either Alice and Bob). Therefore revealing Bob's session key will reveal nothing about Alice's key.

In summary, this approach "***partial transcripts as*** sID" is also problematic for the analysis of key exchange protocols. First, it is highly protocol-dependent. Normally, cryptographic security definitions should be abstract and applicable to all concrete instantiations of the considered protocol class. This simplifies comparisons between different instantiations of the protocol. It allows to abstractly treat key exchange protocols when using them as a building block in more complex protocols. Otherwise, not only is it inconvenient for the analysts to adapt to new specific session IDs each time they analyzes a new key exchange protocol but it also makes schemes harder to compare. The second reason, developing a new definition of session identifier sID specifically crafted for a certain scheme is arguably more error-prone than having a general definition that is verified once and for all.

7.1.3 Session IDs based on Matching Conversations

Finally, one always may consider to use the entire message flow - the transcript - between two parties as the session identifier, namely MC-based sID. Informally speaking, two oracles are said to have *matching conversations* (MC) if every message sent by the first one has actually been received by the second one and vice versa. The great advantage of the MC-based notion is that it can be applied to any key exchange protocol as the session

identifier. Moreover, it is not generated externally but as part of the protocol itself [3]. Thus, many papers [BPR00, BCPQ01, JKL04, Kra05a, JKL06, KY07, LLM07a, ZY10, SEVB10, CF12, ABS14] define the notion of session identifiers (sIDs) using matching conversations, i.e., the concatenation of messages exchanged during the protocol run. To model the real world implementation, the MC-based sIDs seem most natural. However, the natural things being considered: "Is this an appropriate choice if we use the notion of MC-based sID to prove the security of key exchange protocols?" To answer this question, we present a theoretical attack, *no-match attack*, and show that proving security under MC-based sID is a delicate issue as follows.

7.2 New Theoretical Attacks on AKE Protocols in Security Models

In this subsection we first introduce a special attack strategy that we denote *no-match* attacks. Moreover, we will present several variants of our no-match attacks together with existing protocols in the literature that are vulnerable when proving security using MC-based sID. We give some examples of existing protocols and security analyses that are affected. Note that all our attacks apply to security models which allows the adversary to corrupt the long-term keys of the owner of the test session or its partner, e.g. in order to model perfect forward secrecy (PFS) or key compromise impersonation (KCI) attacks. Let us sketch our attacks.

7.2.1 Overview of No-Match Attack

In a no-match attack, the adversary does not actively modify the messages exchanged between two oracles except for some subtle changes. These changes are introduced in such a manner that both parties still compute the same key (or at least two related keys). However, as a result of the modifications they do not have matching conversations. By the definition of security, the adversary can now reveal the session key of one oracle and use it to distinguish the key of the other oracle (as test oracle) from a random key. We stress that since both oracles do not have matching conversations revealing the key of the first oracle formally does not violate the winning conditions of the adversary in the key indistinguishability definition. Nevertheless, it is a trivial attack since the key is delivered by a Reveal query. Our strategy emphasizes how crucial the definition of session ID (sID) is. We stress that our attack *does not* lead to practical attacks on the protocol. It is rather a theoretical attack that proposes an obstacle in security reductions. In the following, we give an informal definition of *no-match* attack.

[3] We note that the terminology here is ambiguous. In fact some papers use 'external' session IDs just as place-holders for any fitting definition of session IDs. In these context, external session IDs do not have to be pre-specified. In this way, one can refer to protocols in general and independent of their specification of session-ID. We note that, typically, these papers use matching conversations as the default instantiations of session IDs in practical settings.

Definition 7.1 (No-match Attack). *We say that an adversary $\mathcal{A}$ defined by an adversarial model has successfully launched a no-match attack if there is a pair of oracles such that they have computed the same session key but the two oracles do not have the same session identifier.*

The type of no-match attacks is applicable to protocols that are shown to provide security against key compromise impersonation (KCI) or forward secrecy (FS). In these scenarios the adversary is additionally provided with at least one long-term key, either that of the Test-oracle or that of the oracle it communicates with. Note that many widely used models in [Kra05a, LLM07a, JKSS12, BFS+13, KPW13] allow the adversary to perform this action. Intuitively we present attackers that use knowledge of this secret key to slightly modify one of the messages sent by communication parties. Usually, the modification consists of a simple re-application of some cryptographic primitive F using fresh randomness. As a consequence, both oracles from communication parties will accepts and still compute the same session key for this session. At the same time, due to the subtle modification, they will not have matching sessions. This allows the adversary to select one oracle as its Test-oracle and make a Reveal-query to another one for the session key. Then, the adversary is able to use it to win the key indistinguishability experiment of the Test-oracle. For simplicity, we suppose that a protocol message contains a cryptographic value v. The general attack proceeds in three steps:

1. $\mathcal{A}$ intercepts a protocol message which contains a cryptographic value v.
2. The adversary computes a distinct cryptographic value $\mathsf{v}' \neq \mathsf{v}$ that makes communication party A compute the same key as when using v.
3. Finally, $\mathcal{A}$ sends the message and v' to A.

Figure 7.1 shows a generic attack process of no-match attacks against AKE protocols. We present no-match attacks on existing protocols (with claimed provable security) when F is a digital signature scheme, a message authentication code, or a public key encryption system.

7.2.2 Class 1 Attacks: Integrity Protection – MACs

The first class of attacks deals with protocols where A and B first establish a common secret key. Further assume that both parties use their long-term secret keys to derive a secret MAC key that is used to guarantee integrity protection of the transcript by exchanging tags over all previous protocol messages in the final two protocol messages. Recently, Dodis, Kiltz, Pietrzak and Wichs introduced efficient algebraic probabilistic MACs [DKPW12]. We informally show that employing such MACs for integrity protection is problematic. Assume that A sends a tag over all previous messages as a protocol message of the protocol. Now if the security definition also allows to reveal the long-term secret key of A the adversary could first create the MAC key, intercept this

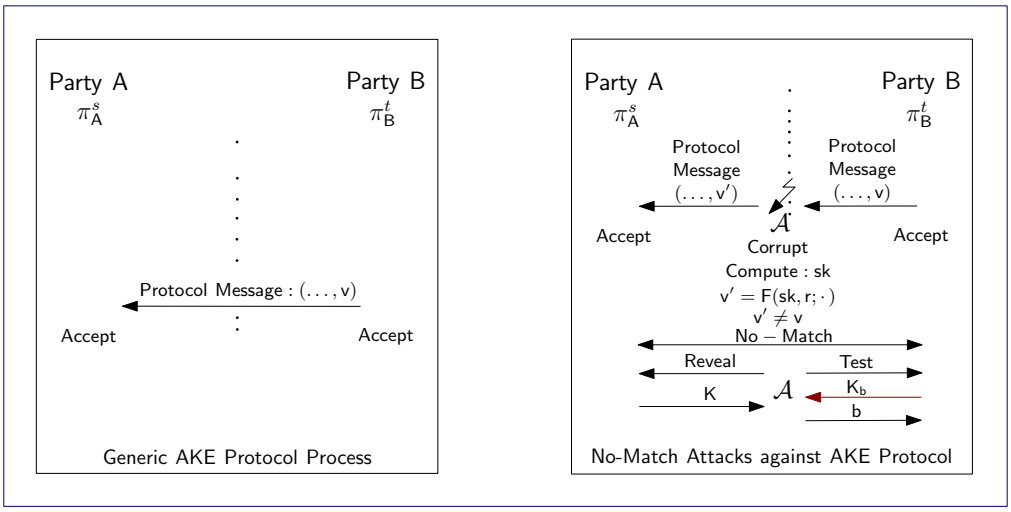


Fig. 7.1. No-Match Attack against AKE prptocols

protocol message, and compute a new tag on the same message using fresh random with the revealed key. This new tag is now sent to B instead of the original one. Both oracles (π_A^s, π_B^t) of parties A and B accept although they do not have matching sessions (due to the subtle modification). Note that the MC-based sID is used in the security proof. This allows the adversary to select one oracle (π_A^s) as its Test-oracle and make a Reveal-query to another one (π_B^t) for the session key K. Then, the adversary is able to use K to answer the Test-oracle.

7.2.2.1 Jeong-Katz-Lee One-Round AKE Protocol $\mathcal{TS}3$ (ACNS'04)

In 2004, Jeong, Katz and Lee presented a provably-secure one-round authenticated key exchange protocol named $\mathcal{TS}3$ [JKL04] which claims to provide forward secrecy without random oracle. However, the $\mathcal{TS}3$ protocol is vulnerable to a MAC-based no-match attack. Essentially in the protocol message a party sends a tag over the messages exchanged so far. This tagging algorithm is probabilistic. The key for the MAC is computed as the Diffie-Hellman value of the long-term public keys of the communication parties. In particular, it can be computed by the adversary if a single secret key is corrupted. An adversary intercepts the protocol message. Then, it may corrupt a party and obtain the secret key of the party. With this secret key the adversary can compute the MAC key and re-apply the MAC to the entire transcript with fresh randomness. This value is sent to its partner. Now both parties compute the same key although they have distinct session identifiers, i.e. no match sessions. Details on the protocol, the security model, and an illustration of our attack can be found in Appendix A.1.

7.2.2.2 Joeng-Kwon-Lee KAM Protocol (CANS'06)

In 2006, Jeong, Kwon and Lee [JKL06] introduced a new protocol, named KAM, that also relies on probabilistic and strongly secure MACs. Here a similar no-match attack is possible. Suppose Alice wishes to communicate with Bob. In the protocol message of the KAM protocol Alice sends to Bob a tag computed over her data. The key for the tagging algorithm is derived from Alice's long-term secret and Bob's ephemeral public key. Thus when considering KCI attacks, where the adversary is granted access to the long-term secret of the holder of the Test-session, the adversary can compute the MAC key on its own before Bob's accepts. It can thus intercept Alice's protocol message and substitute it with a new tag that is computed over the same messages but using distinct randomness. We stress that the KAM protocol also admits another type of no-match attacks. In this attack variant the adversaries obtains the secret MAC key by computing the ephemeral secret key of Bob via a RevealState query. Using this key and Alice's long term public key, the adversary can easily compute the MAC key used to authenticate messages sent from Alice to Bob. The rest of the attack proceeds as before. We emphasize that the attack works although no long-term secrets are exposed. Details on the KAM protocol, the security model and an illustration of our attack can be found in Appendix A.2.

7.2.3 Class 2 Attacks: Authentication via Digital Signatures

To illustrate our second attack, assume A sends a probabilistically generated digital signature on message m to B during the protocol execution. The adversary first intercepts this value, corrupts A to obtain its secret key, and then generates a new signature on m. With overwhelming probability this signature will differ from the original one. Finally, it sends the new signature to B who checks its validity and on success sends some other values to A. The adversary will not interfere anymore. At the end of the protocol, A will accept although there is no matching conversation with B. This allows the adversary to select one oracle as its Test-oracle and make a Reveal-query to another one for the session key. Then, the adversary is able to use it to win the key indistinguishability experiment of the Test-oracle.

7.2.3.1 Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal (ESORICS'12)

In 2012, Cremers and Feltz [CF12] presented a signature-based compiler which achieve perfect forward secrecy (PFS) in two-message or one-round key exchange protocols that satisfy stronger security properties than provided by eCK model [LLM07a]. PFS is considered in the presence of adversary that is allowed to reveal ephemeral secret keys and the long-term secret keys of the actor of a session. Paper [CF12] refers to four models, M^w, eCK^w, M-PFS, eCK-PFS. In contrast with (eCK^w, eCK-PFS), M^w and M-PFS do not consider the EphemeralKey query which reveals the ephemeral secret keys

of protocol sessions. Note that in paper [CF12], Cremers and Feltz considered a special case of randomized digital signature in which an efficient adversary $\mathcal{A}$ can find out the secret signature key if $\mathcal{A}$ can obtain the corresponding public signature key, a valid signature and the random coins involved in this signature generation learned through an EphemeralKey query. Therefore, they specified the signature scheme to be deterministic. In theory, if the random coins involved in the signature are not defined as the contents of the secret state or there exists no EphemeralKey query in the security model, randomized signature schemes could be applied to the Cremers-Feltz's compiler. The authors also claimed in [CF12] (Remark 1) that the signature-based compiler could be a randomized (SUF-CMA) signature scheme when the EphemeralKey query is not included in the security models. Unfortunately, even if the security models include these restrictions, the signature-based compiler is still not secure due to our no-match attacks. For simplicity, we only show one case in which no EphemeralKey query can be considered.

In Appendix A.3 we provide more details: we first give an informal overview of the used security model when the EphemeralKey query is not included, M-PFS. (More information on the remaining proposed models, can be found in [CF12].) Then, we describe the SIG(NAXOS) protocol which is generated by applying the signature-based compiler to the well-known protocol NAXOS [LLM07a]. (For more details on NAXOS we refer to [LLM07a].) Note that Cremers and Feltz first define $\mathcal{DH}_2$ as the class of all two-message key exchange protocols. As examples of protocols that belong to this class, they list NAXOS [LLM07a], NAXOS+ [LP08b], NETS [LP08a] and CMQV [Ust08].

7.2.3.2 Signed Diffie-Hellman using NAXOS Transformation Protocol (SCN'10)

In 2010, Sarr et al. [SEVB10] constructed a signed Diffie-Hellman protocol that uses the NAXOS transformation protocol, denoted here as $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ (in the original paper it is referred to as Protocol 2), and claimed that if the ephemeral keys are defined to be the nonces r_i and r_j selected by parties P_i and P_j and the signature scheme is secure against chosen message attacks (UF-CMA), this protocol can be proved secure in the eCK model (relying on the same arguments as in the NAXOS security proof [LLM07a]). However, if randomized signature schemes are applied to the $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ protocol, it is vulnerable to our no-match attacks. As before we defer all detail to Appendix A.4.

7.2.4 Class 3 Attacks: Authentication via Public Key Encryption

A similar attack can be launched if B sends an encrypted message to A which has to be decrypted and checked using As secret key. The adversary may corrupt A, intercept the ciphertext, decrypt it and compute a new ciphertext on the same message. If the encryption system is probabilistic (what is required for even CPA security) the ciphertext will differ from the original one with overwhelming probability. However, A will accept the new ciphertext and accept without having a matching conversation with B.

7.2.4.1 Affected Protocols: PKE-Based Key Exchange Protocol π (ACISP'14)

In 2014, Alawatugoda et al. [ABS14] presented a leakage model for key agreement protocols called the continuous after-the-fact leakage model (CAFL model, and showed a generic construction of a protocol π that is claimed to be secure in the CAFL model. Informally, the CAFL model is an after-the-fact leakage model for key exchange protocols, i.e. the leakage happens after the test session is activated. The protocol π is a key agreement protocol based on a public key encryption scheme which is additionally secure under adaptively chosen ciphertext after-the-fact leakage (CCLA2). In the protocol π, each party randomly chooses its ephemeral secret key, encrypts it with the public key of its intended party using the (CCLA2) encryption function and sends the ciphertext to its intended partner. The authors provided a purported proof of the protocol π secure in the CAFL model. Unfortunately, our no-match attack can be applied to the protocol π in the CAFL model. As before, we postpone the details to Appendix A.5.

Remark 7.2. Our no-match attack can also be performed for against the signature-based π protocol which is described in Paper [ASB14]. The signature-based π protocol is based on a signature scheme with the UF-CMLA security property (*unforgeability against chosen message leakage attacks*) and uses the NAXOS trick for computing the pseudo-ephemeral secret key. It is proved secure in After-the-fact Leakage-eCK (AFL-eCK) model and guarantees Leakage-eCK security. In this model, partnership is defined using matching conversations approach. And the adversary is allowed to corrupt the owner of the test session, in addition to obtaining bounded amount of leakage from the partner of the test session. Note that according to the definition of leakage-resilient signature, the signature algorithm might be randomized. The adversary $\mathcal{A}$ performs the similar actions as described above: $\mathcal{A}$ first intercepts the signature σ_i on message m, corrupts P_i to obtain its secrete key sk_i. Then, $\mathcal{A}$ generates a new signature σ_i^* on the same message m and sends σ_i^* to party P_j who checks its validity. At the end of the protocol, P_i will accept although there is no matching session with P_j. $\mathcal{A}$ selects oracle π_i^s as its Test-oracle and make a Reveal-query to π_j^t for the session key $\mathsf{K}_{i,j}$. Finally, $\mathcal{A}$ can use it to win the key indistinguishability experiment of the Test-oracle.

7.2.5 Summary: Affected AKE Protocols

In this subsection, we give a summary of the affected AKE protocols described above. We classify the differences into three categories: authentication via message authentication code, authentication via signature and authentication via digital signature. Figure 7.2 below shows the summary of affected protocols described above.

AKE / Class	Affected Protocols	No-Match Attacks	MC-based sID	KCI Attacks	Forward Secrecy Attacks
Class 1: Authentication via PKE	Protocol: PKE-Based π (ACISP'14)	Yes	Yes	Yes	Yes
Class 2: Authentication via MAC	Protocol: $\mathcal{TS}3$ (ACNS'04)	Yes	Yes	Yes	Yes
	Protocol: KAM (CANS'06)	Yes	Yes	Yes	Yes
Class 3: Authentication via Signature	Protocol: $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ (SCN'10)	Yes	Yes	Yes	Yes
	Compiler for PFS SIG(NAXOS) (ESORICS'12)	Yes	Yes	Yes	Yes
	Protocol: SIG-Based π (AsiaCCS'14)	Yes	Yes	Yes	Yes

Fig. 7.2. Summary of affected AKE Protocols

7.3 Solutions

Let us stress that the above attacks do seemingly not harm the practical security of the above protocols in any meaningful way. However, strictly speaking their security proofs are not sound (when considering session IDs based on matching conversations). Ideally our solutions should be as less invasive as possible to make them easily applicable to existing protocol implementations as well (either via only minor modifications of the protocol or no modification at all). In the following we propose several solutions to cope with the above attacks. One can either modify the protocols or the security definitions to re-obtain provable security. Firstly, we analyze what properties primitives like signature schemes, MACs must satisfy. The rationale is that in some protocols where the used primitives can be decided upon one could simply choose primitives that make the protocol withstand no-match attacks. In this sense the overall protocol implementation could remain unaltered. Our second approach proposes a general transformation from a protocol that is vulnerable to no-match attacks to one which isn't. The transformation is formulated in form of a simple protocol compiler that can use a key exchange protocol in a black-box manner. That is, it only requires the message transcript and the session key computed in a protocol run as input. Advantageously, our compiler is very efficient.

7.3.1 Uniqueness

The first and probably most simple solution is to only use unique primitives in primitives, such as digital signature or MAC scheme. The advantage of this solution is that it does not necessarily modify the protocol. In the following, we first explain why uniqueness property must be required for thwart *no-match* attacks.

7.3.1.1 Insufficiency of Strong Security and Deterministic Computation of v

It is relatively obvious that our no-match attacks succeed if the cryptographic value v is computed using a probabilistic algorithm. For example, if v is a digital signature, an attacker that obtains the secret signing key can simply re-sign the message to compute v'. With high probability we have $\mathsf{v}' \neq \mathsf{v}$. What is more subtle is that our attacks also work if the signature scheme used to compute v provides strong security or even is deterministic. This is exactly the point where many security proofs fail. Let us go into more detail for our running example, signed Diffie-Hellman. Recall that the security definition of strongly secure signatures gives the adversary access to the public key and a signing oracle. The winning condition is that the adversary can produce a new message/signature pair. Now consider an attempt to reduce the security of signed Diffie-Hellman to the strong security of the signature scheme. The crucial point is that in no-match attacks the adversary is also given the secret key. In particular, the security definition of strongly secure signatures does not exclude that the adversary may produce a new signature on a previously queried message when the secret key is given.

Quite similarly, it is not enough to require that the signature scheme has a deterministic signing procedure. The problem is that this only guarantees that the signing algorithm *specified by the signature scheme* outputs a single signature v per message m. However, there may exist other algorithms that output, given m, a signature $\mathsf{v}' \neq \mathsf{v}$ such that both (m, v) and (m, v') pass the signature verification positively. We remark that the definitions of deterministic signatures and unique signatures refer to different algorithms of the signature scheme [GO93, Cor02, KK12]. Whereas a deterministic signature scheme refers to a signature scheme with a deterministic *signing* algorithm, unique signatures refer to signature schemes whose *verification* algorithm meets the uniqueness property. The same argument can be applied to MAC scheme.

Remark 7.3. This may be impossible for public key encryption. Note that the well-known fact that to even meet the weak notion of IND-CPA-security, any PKE scheme must be probabilistic [GM84]. We are aware that for PKE-based authentication, security protocols usually require the stronger notion of IND-CCA security (or some related notion with restricted access to a decryption oracle). Interestingly, if the adversary also holds the secret key of the encryption system in a no-match attack, there is no obvious way to protect the ciphertext against a no-match attack. It uses the secret key to decrypt the ciphertext and obtain the plaintext. Next the plaintext is re-encrypted with fresh randomness. Similarly, if the adversary obtains the plaintext message of the ciphertext, there is no way to prevent her from re-encrypting it with a fresh randomness. Thus it seems that no-match attacks on IND-CCA-secure public key encryption cannot be protected against by an appropriate choice of the scheme from some subclass since IND-CCA-secure PKE must always provide probabilistic encryption.

7.3.1.2 Unique Verification

We propose a way to prevent *no-match* attacks that requires unique primitives. In this subsection, we give the formal definition of uniqueness.

Definition 7.4 (Unique Verification). *A verification algorithm* Vfy *defined over (*$\mathcal{K}$*,* $\mathcal{M}$*,* $\mathcal{V}$*,* $\mathcal{X}^{aux}$*) is an efficient algorithm:* $\mathcal{K} \times \mathcal{M} \times \mathcal{V} \times \mathcal{X}^{aux} \rightarrow \{0,1\}$ *described as a deterministic Turing Machine. The set* $\mathcal{K}$ *is called the key space, the set* $\mathcal{M}$ *is called the message space, the set* $\mathcal{V}$ *is called verification message space and* $\mathcal{X}^{aux}$ *is called the auxiliary information space. We say that* Vfy *provides unique verification if for all* $k \in \mathcal{K}$*, all messages* $m \in \mathcal{M}$ *and all* $\mathsf{aux} \in \mathcal{X}^{aux}$ *we always have that*

$$|\{v \in \mathcal{V} | \mathsf{Vfy}(k, m, v, \mathsf{aux}) = 1\}| \leq 1.$$

We require that the space $\mathcal{X}^{aux}$ *contains a special bit-string called the empty string denoted* $\emptyset$*.*

A verification function enables one to evaluate $V(k, m, v, \mathsf{aux})$ given the inputs k, m, v and aux, where $k \in \mathcal{K}$, $m \in \mathcal{M}$, $v \in \mathcal{V}$ and $\mathsf{aux} \in \mathcal{X}^{aux}$. In the above definition we generally consider algorithms that are keyed with some key k. Depending on the primitive this key can be a symmetric, public, or secret key.

7.3.2 Session Key Derivation Compiler for No-Match Attacks

All our solution aim at being applicable to existing implementations of key exchange protocols. The above solutions attempt to not modify the protocol. In the following we will also consider solutions that change it. However, we try to keep as less invasive as possible. To us an ideal solution is a compiler that only requires the secret session key of the original key exchange protocol and the completely exchanged messages between the sender and the receiver. In this spirit, we present a general solution to protect protocols against no-match attacks in the form of a protocol compiler. Assume we have a key exchange protocol that is only susceptible to no-match attacks but secure otherwise. In other words, the session keys of the original protocols should be indistinguishable from random with respect to the security model in which no-match attacks are not considered. To formally capture this, we could for example define the class of such protocols to be all protocols which are secure when using unique signatures (or MACs) but insecure when using non-unique signatures. To thwart the protocol also against no-match attacks we propose the following efficient transformation. Let OTPRF be a one-time PRF (i.e., a PRF that is secure only if the adversary can query a single message to the PRF oracle). Let K be the key that is output by the original protocol. We now provide a modified key derivation routine. Given K, the new session key is computed as $\mathsf{K}^* = \mathsf{OTPRF}(\mathsf{K}, \mathsf{T_C})$ where $\mathsf{T_C}$ is the transcript consisting of all the messages sent and received by some

communication party $\mathsf{C} \in \{\mathsf{A}, \mathsf{B}\}$ in chronological order (possibly after applying some deterministic sorting of messages in each protocol move) [4].

Clearly, the attacker still can exchange the cryptographic value v with a new one v' on the same message. Further, the two partners will also not have a matching conversation. However, the attacker cannot use the Reveal-query to successfully answer the Test-query. This is because our transformation OTPRF involves the whole transcript of all messages sent and received by oracles. Instead of sending v, the *no-match* attacker sends v' to oracle π^s_{A}, see Figure 7.1. It is easy to see that in this case the transcript T'_{A} of π^s_{A} does not match the transcript T_{B} of π^t_{B}. As a result, the communication oracles do not share the same session key after this attack, i.e., $\mathsf{OTPRF}(\mathsf{K}, \mathsf{T}'_{\mathsf{A}}) \neq \mathsf{OTPRF}(\mathsf{K}, \mathsf{T}_{\mathsf{B}})$ where $\mathsf{T}'_{\mathsf{A}} \neq \mathsf{T}_{\mathsf{B}}$. Thus, the attacker cannot win the security experiment of the Test oracle via the Reveal-query. Figure 7.3 shows our generic compiler against no-match attacks against. We can instantiate the one-time PRF very efficiently using pairwise-independent

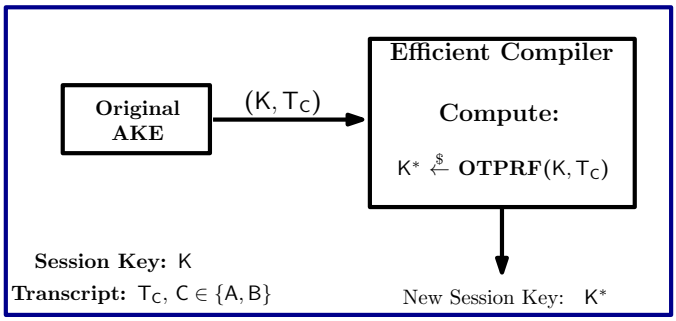


Fig. 7.3. Efficient Compiler for No-Matching Attacks

hash functions [GL89]. As an advantage, pairwise-independent hash functions provide statistical security (i.e., security even against unbounded adversaries) in contrast to general PRFs. As a result no additional complexity assumptions are required in the security proof of the modified protocol. As an advantage, the above compiler is computationally very efficient and does not add any security assumptions to the protocol. However, it does modify the key derivation function of the protocol. Typically, the key derivation function is fixed in protocol suites whereas sometimes users can choose the concrete algorithms used to produce the protocol messages like for example in TLS where the users can choose among a predefined set of algorithm-combinations, the ciphersuites. Thus for existing implementations it may be possible for the protocol participants to agree on a unique signature scheme whereas there is no way to modify the key derivation of the protocol. Another disadvantage of the protocol as presented above is that each user has to store the actual state of the transcript until the handshake phase is finished. In more complex protocols this could be exploited to launch denial of service attacks

[4] Note that only the key K from the original key exchange protocol is indistinguishable from random, we can then actually reduce security to that of the OTPRF. The proof is straight-forward.

on the protocol. A simple way to reduce the state held by the communication partners is to use a cryptographic hash function Hash to hash the transcript in a step-by-step fashion. In this way, the transcript T_i after the i-th exchanged message m_i consists of $\mathsf{T}_i = \mathsf{Hash}(\mathsf{T}_{i-1}, m_i)$ where T_0 is defined to some constant string.

7.3.3 Discussion of our Solutions

We give several efficient ways to thwart no-match attacks. However, the solutions described above have distinct advantages and disadvantages. Such as using unique primitives, the advantage of this solution is that it does not necessarily modify the original protocol. Nevertheless, requiring uniqueness of verification severely restricts the class of suitable cryptographic primitives, in particular when relying on digital signatures. It is true that probabilistic signature schemes can be made deterministic by using some general transformations, e.g., by deriving the necessary randomness via a pseudo-random function (PRF). Since PRFs can be built from one-way functions it does not increase the theoretical complexity of the constructions. However, deterministic signatures cannot be easily mapped to unique signatures via some "efficient" approaches, e.g. PRFs. Furthermore, several theoretical results [GO93, Cor02, BPR$^+$08, FS12, KK12] indicate that there is no "simple" transformation from deterministic signatures to unique signatures. For our compiler solution, we require the (original) secret session key and the complete transcripts. From another point of view, we modify the original protocols, although the transformation is very efficient. By nature, it is the best solution if we can improve the existing generic security model to avoid some trivial attacks. However, it seems not so easy to get a generic solution. As open problem, it would be interesting to develop a new formal generic definition of session identifier, or to improve the generic definition of matching conversation-based session identifier to remove some non-meaningful attacks in the security proofs.

Chapter 8

Conclusion

In this chapter we conclude the thesis by discussing our main results. First, we discuss the security of all the TLS-PSK ciphersuites which are often used for remote authentication between servers and constrained clients like smart cards or new electronic ID card, and give the first formal security proof in the standard model. Then, we construct the first secure Authenticated Key Exchange (AKE) protocol, called tAKE, in standard model whose security does not degrade with an increasing number of participating parties and sessions. Moreover, we show how to modularly construct secure, efficient and reliable AKE systems. At the same time, we also present two efficient AKE-compilers. Finally, we present a theoretical attack for an authenticated key exchange, named here *no-match attack*, and show that proving security under the matching conversations as session IDs (MC-based sID) is a delicate issue.

8.1 Discussion of our Security Analysis of TLS-PSK ciphersuites

We prove the security of Transport Layer Security Pre-Shared Key ciphersuites (TLS-PSK) in Chapter 4. The TLS-PSK is very important for remote authentication between servers and constrained clients, like smart cards or new electronic ID card, and so forth. We give the first formal security analysis for TLS-PSK ciphersuites. In order to analyze these protocols, we introduce the first definition of ACCE security for authentication protocols with pre-shared keys. We do not propose a separate model but rather an extension of the ACCE model of JKSS as described in [JKSS12] to also cover authentication via pre-shared keys. Then, we introduce a new variant of pseudo-random functions (PRFs), called *double pseudo-random function* (DPRF) for the security analysis of `TLS_RSA_PSK` and `TLS_DHE_PSK`. Finally, we show that `TLS_PSK` is ACCE secure (*without forward secrecy*), `TLS_RSA_PSK` is ACCE secure with *asymmetric perfect forward secrecy* and `TLS_DHE_PSK` is secure with (classical) *perfect forward secrecy*.

LIMITATIONS FOR OUR RESULTS.

In our work, we give a dedicated security analysis for TLS-PSK ciphersuites. We believe that it is possible to give a more modularized analysis, similar to [KPW13] who

analyzed the classical ciphersuites of TLS by abstracting the handshake phase into a Constrained-CCA-secure (CCCA) key encapsulation mechanism that is combined with a secure authenticated encryption scheme.

We stress that when showing that TLS_RSA_PSK provides asymmetric perfect forward secrecy, we do not consider TLS-RSA with RSA-PKCS encryption as it is currently used in practice. Instead we rather assume that it uses a generic IND-CCA secure encryption scheme that is secure in the standard model. It would be interesting to show that the results of [KPW13] on TLS-RSA with RSA-PKCS encryption can be transferred to show that TLS-PSK with RSA-PKCS based key transport provides asymmetric perfect forward secrecy in the random oracle model.

8.2 Discussion of our Tightly-Secure AKE Protocol

In Chapter 5, we construct the first secure Authenticated Key Exchange (AKE) protocol, called tAKE, in standard model whose security does not degrade with an increasing number of participating parties and sessions. We describe a generic three-pass AKE protocol and prove its security in an enhanced Bellare-Rogaway security model under the standard assumption. Our construction is modular and enjoys a tight security reduction. Moreover, it can be instantiated efficiently from standard assumptions. For instance, we give an SXDH-based protocol whose communication complexity is only 14 group elements and 4 exponents.

LIMITATIONS AND OPEN PROBLEM FOR OUR RESULTS.

Our enhanced Bellare-Rogaway security model provides perfect forward secrecy (PFS) and key compromise impersonation (KCI) attacks. Moreover, we model also practical PKI-related attacks. We allow the adversary to issue more than one Test-query. On the other hand, we do not allow the adversary to reveal the internal states or intermediate results of the computation of the session key, i.e., no RevealState-query as described in the Canetti-Krawczyk model. It would be interesting to show the existence of a tightly secure construction in such a security model with RevealState-query.

8.3 Discussion of our efficient AKE-Compilers

In Chapter 6, we describe our efficient AKE-compilers that generically turn passively secure key exchange protocols (KE) into authenticated key exchange protocols (AKE) where security also holds in the presence of active adversaries. Our first compiler is very efficient. It relies on signature schemes and only requires two additional moves in which signatures are exchanged. The second compiler relies on public key encryption systems. Although the first compiler is more efficient, the second compiler accounts for scenarios where the parties do not have (certified) signature keys but only encryption keys. This can often occur in practice.

From a modular processing perspective, our compilers only require the public transcript T of the key exchange protocol, and do not require any modifications in the underlying KE protocols. Thus, our compilers are easily applicable to existing systems, what makes them very useful in practice. Considering efficiency evaluation, our efficiency improvements rely on the following techniques as follows. We first use a form of *implicit key confirmation* instead of *explicit key confirmation*. As our second efficiency improvement, we formally show that for security we do not have to exchange uniformly random nonces after the key exchange protocol. Finally, our compilers do not need to computer a new session key for AKE session. In summary, it only leads to the additional computation and communication and complexity $\mathcal{O}(1)$.

8.4 Discussion of our No-Match Attacks

In Chapter 7, we present a new theoretical attack for key exchange protocols, named here *no-match attack*, and show that proving security under the matching conversations as session IDs (MC-based sID) is a delicate issue. Moreover, we provide also several examples of protocols that claim to be secure under a security definition based on MC-based sID but where the security proof is actually flawed. We show that no-match attacks are often hard to avoid without costly modifications of the original protocol. Finally, we discuss several ways to thwart no-match attacks.

LIMITATIONS OF OUR RESULTS ON *No-Match* ATTACK.

In our work, we give several efficient ways to thwart no-match attacks. However, these solutions have distinct advantages and disadvantages. Such as using unique primitives in the key exchange protocols, the advantage is that it does not necessarily modify the original protocols. However, requiring uniqueness of verification severely restricts the class of suitable cryptographic primitives, in particular when relying on digital signatures. As our second solution, we need to modify the original protocols. As open problem, it would be interesting to develop a new *formal generic* definition of session identifier, or to improve the generic definition of matching conversation-based session identifier to remove some non-meaningful attacks in the security proofs.

References

ABP+13. Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the security of RC4 in TLS. In *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, pages 305–320, 2013. 15

ABS14. Janaka Alawatugoda, Colin Boyd, and Douglas Stebila. Continuous after-the-fact leakage-resilient key exchange. pages 2258–273, 2014. 32, 126, 131, 164

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. pages 209–236, 2010. 96

AP13. Nadhem J. AlFardan and Kenneth G. Paterson. Lucky thirteen: Breaking the TLS and DTLS record protocols. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 526–540. IEEE Computer Society, 2013. 15

ASB14. Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Modelling after-the-fact leakage for key exchange. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 207–216, 2014. 131

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. pages 259–274, 2000. 16

BCF+13. Colin Boyd, Cas Cremers, Michele Feltz, Kenneth G. Paterson, Bertram Poettering, and Douglas Stebila. ASICS: authenticated key exchange security incorporating certification systems. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, pages 381–399, 2013. 15

BCK98. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). pages 419–428, 1998. 17, 38, 110, 121

BCPQ01. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. pages 255–264, 2001. 126

BDK+14. Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. Multi-ciphersuite security of the secure shell (ssh) protocol. In *2014 ACM SIGSAC Conference on Computer and Communications Security, CCS'14, Scottsdale, Arizona, USA, November 3-7, 2014*, 2014. 52, 124

Bel06. Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. pages 602–619, 2006. 74

Ber08. Daniel J. Bernstein. Proving tight security for Rabin-Williams signatures. pages 70–87, 2008. 16

BFS+13. Christina Brzuska, Mark Fischlin, Nigel P. Smart, Bogdan Warinschi, and Stephen C. Williams. Less is more: Relaxed yet composable security notions for key exchange. *International Journal of Information Security*, 12(4):267–297, August 2013. 127

BG07. U. Blumenthal and P. Goel. Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS). RFC 4785 (Proposed Standard), January 2007. 65

BHJ+14. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. Cryptology ePrint Archive, Report 2014/797, 2014. `http://eprint.iacr.org/`. 18, 93

BHJ+15. Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In *Theory of Cryptography, 12th Theory of Cryptography Conference, TCC 2015, in Warsaw, Poland on March 23 to 25, 2015. Proceedings*, 2015. 18, 27, 29, 93, 94, 97

BKN06. M. Bellare, T. Kohno, and C. Namprempre. The Secure Shell (SSH) Transport Layer Encryption Modes. RFC 4344 (Proposed Standard), January 2006. 15

BKP14. Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (hierarchical) identity-based encryption from affine message authentication. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 408–425, 2014. 16

Ble98. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. pages 1–12, 1998. 64

Bot13. Botan Software Developers. Botan, 2013. `http://botan.randombit.net/`. 16, 64

Bou13. BouncyCastle Software Developers. Bouncy Castle Crypto APIs, 2013. `http://www.bouncycastle.org/`. 16, 64

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. pages 139–155, 2000. 17, 38, 45, 126

BPR+08. Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. pages 283–292, 2008. 136

BR93a. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. pages 232–249, 1993. 37, 38, 39, 45, 47, 52, 54, 124

BR93b. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. pages 62–73, 1993. 16

BR93c. 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, 1993. 65

BR95. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. pages 57–66, 1995. 17, 38, 39, 124

BR96. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. pages 399–416, 1996. 16

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. pages 409–426, 2006. 36, 72, 82, 103, 116, 119

BU04. Mohamad Badra and Pascal Urien. Toward SSL integration in SIM smartcards. In *WCNC*, pages 889–893, 2004. 15, 16

BWJM97. Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. pages 30–45, 1997. 17, 37, 38, 45, 50, 52

BWM99a. Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols (invited talk). pages 339–361, 1999. 50

BWM99b. Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (STS) protocol. pages 154–170, 1999. 37, 38

BWM99c. Simon Blake-Wilson and Alfred Menezes. Unknown key-share attacks on the station-to-station (sts) protocol. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999. 45, 47, 50, 51

BWNH+03. S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright. Transport Layer Security (TLS) Extensions. RFC 3546 (Proposed Standard), June 2003. Obsoleted by RFC 4366. 15

Can00. Ran Canetti. Security and composition of multiparty cryptographic protocols. 13(1):143–202, 2000. 36

CBH05a. Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. Errors in computational complexity proofs for protocols. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer, 2005. 15

CBH05b. Kim-Kwang Raymond Choo, Colin Boyd, and Yvonne Hitchcock. On session key construction in provably-secure key establishment protocols: Revisiting chen & kudla (2003) and mccullagh & barreto (2005) id-based protocols. *IACR Cryptology ePrint Archive*, 2005:206, 2005. 15

CBHM04. Kim-Kwang Raymond Choo, Colin Boyd, Yvonne Hitchcock, and Greg Maitland. On session identifiers in provably secure protocols: The bellare-rogaway three-party key distribution protocol revisited. In *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, pages 351–366, 2004. 15

CF12. Cas J. F. Cremers and Michele Feltz. Beyond eCK: Perfect forward secrecy under actor compromise and ephemeral-key reveal. pages 734–751, 2012. 126, 129, 130, 157, 159

CK01. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. pages 453–474, 2001. 17, 38, 39, 45, 52, 96, 124

CK02a. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. pages 143–161, 2002. http://eprint.iacr.org/2002/120/. 37

CK02b. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. pages 337–351, 2002. 17, 36, 38, 124

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. pages 494–503, 2002. 36

Cor00. Jean-Sébastien Coron. On the exact security of full domain hash. pages 229–235, 2000. 16

Cor02. Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. pages 272–287, 2002. 133, 136

Cre09. Cas J. F. Cremers. Formally and practically relating the ck, ck-hmqv, and eck security models for authenticated key exchange. *IACR Cryptology ePrint Archive*, 2009:253, 2009. 41

CTM12. Chunhua Chen, Shaohua Tang, and Chris J. Mitchell. Building general-purpose security services on EMV payment cards. In *SecureComm*, pages 29–44, 2012. 66

CW13. Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure ibe and dual system groups. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 435–460. Springer, 2013. 16

DA99. T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746. 15, 65

DAT12. Italo Dacosta, Mustaque Ahamad, and Patrick Traynor. Trust no one else: Detecting MITM attacks against SSL/TLS without third-parties. In Foresti et al. [FYM12], pages 199–216. 64

DF11. Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor. pages 702–721, 2011. 32

DGH$^+$04. Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. pages 494–510, 2004. 74

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 43

DKPW12. Yevgeniy Dodis, Eike Kiltz, Krzysztof Pietrzak, and Daniel Wichs. Message authentication, revisited. pages 355–374, 2012. 127

DP07. Jean Paul Degabriele and Kenneth G. Paterson. Attacking the ipsec standards in encryption-only configurations. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 335–349, 2007. 15

DR06. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. RFC 4346 (Proposed Standard), April 2006. Obsoleted by RFC 5246, updated by RFCs 4366, 4680, 4681, 5746. 65

DR08. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878. 65

ECR12. Ecrypt2: Yearly report on algorithms and keysizes (2011-2012). Technical report, 2012. 16

ET05. P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). RFC 4279 (Proposed Standard), December 2005. 65, 69

FHH14. Eduarda S. V. Freire, Julia Hesse, and Dennis Hofheinz. Universally composable non-interactive key exchange. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 1–20, 2014. 36

fISB05. German Federal Office for Information Security (BSI). TR-03112, Das eCard-API-Framework, 2005. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html. 65

FLW10. Marc Fischlin, Anja Lehmann, and Daniel Wagner. Hash function combiners in TLS and SSL. pages 268–283, 2010. 74

FPS06. M. Friedl, N. Provos, and W. Simpson. Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4419 (Proposed Standard), March 2006. 15

FPZ08. Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. HMAC is a randomness extractor and applications to TLS. pages 21–32, 2008. 74

FS12. Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. pages 636–653, 2012. 136

FYM12. Sara Foresti, Moti Yung, and Fabio Martinelli, editors. *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*. Springer, 2012. 143

GBN09. M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Modeling key compromise impersonation attacks on group key exchange protocols. pages 105–123, 2009. 50

GJKW07. Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. 20(4):493–514, October 2007. 94

GKS13. Florian Giesen, Florian Kohlar, and Douglas Stebila. On the security of TLS renegotiation. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 387–398, 2013. 52

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. pages 25–32, 1989. 135

GM84. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 133

GMP+08. Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally Composable Security Analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec*, volume 5324 of *LNCS*, pages 313–327. Springer, 2008. 64

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. 17(2):281–308, April 1988. 28, 29

GO93. Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). pages 228–245, 1993. 133, 136

Gol98. Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer, 1998. 21

Gol01. Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001. 21, 22

Gol04. Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. 21

Gol10. Oded Goldreich. *P, NP, and NP-Completeness: The Basics of Complexity Theory*. Cambridge University Press, 2010. 21, 22

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. pages 97–111, 2006. 96

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. pages 444–459, 2006. 96

Gro10. Jens Groth. Short non-interactive zero-knowledge proofs. pages 341–358, 2010. 96

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. pages 415–432, 2008. 96

GT06. J. Galbraith and R. Thayer. The Secure Shell (SSH) Public Key File Format. RFC 4716 (Informational), November 2006. 15

Har06. B. Harris. RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4432 (Proposed Standard), March 2006. 15

HC98. Dan Harkins and Dave Carrel. The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard), 1998. 15

HJ12. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. pages 590–607, 2012. 16, 96

HSGW06. J. Hutzelman, J. Salowey, J. Galbraith, and V. Welch. Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol. RFC 4462 (Proposed Standard), May 2006. 15

JK02. Jakob Jonsson and Burton S. Kaliski Jr. On the security of RSA encryption in TLS. pages 127–142, 2002. 64

JKL04. Ik Rae Jeong, Jonathan Katz, and Dong Hoon Lee. One-round protocols for two-party authenticated key exchange. pages 220–232, 2004. 64, 126, 128, 151

JKL06. Ik Rae Jeong, Jeong Ok Kwon, and Dong Hoon Lee. A Diffie-Hellman key exchange protocol without random oracles. pages 37–54, 2006. 126, 129, 154, 155

JKSS10. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. Generic compilers for authenticated key exchange. pages 232–249, 2010. 17, 110, 111, 112, 121

JKSS12. Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. pages 273–293, 2012. 31, 36, 39, 47, 50, 52, 53, 55, 62, 64, 65, 127, 137

JV96. Mike Just and Serge Vaudenay. Authenticated multi-party key agreement. pages 36–49, 1996. 50

Kau05. C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306 (Proposed Standard), December 2005. Obsoleted by RFC 5996, updated by RFC 5282. 15

KBC97. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. Updated by RFC 6151. 74

KK12. Saqib A. Kakvi and Eike Kiltz. Optimal security proofs for full domain hash, revisited. pages 537–553, 2012. 16, 133, 136

KL05. Jonathan Katz and Yehuda Lindell. Handling expected polynomial-time strategies in simulation-based security proofs. pages 128–149, 2005. 36

KM11. Neal Koblitz and Alfred Menezes. Another look at security definitions. *IACR Cryptology ePrint Archive*, 2011:343, 2011. 50, 51

KP05. Caroline Kudla and Kenneth G. Paterson. Modular security proofs for key agreement protocols. pages 549–565, 2005. 17, 38

KPW13. Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the tls protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2013. 52, 54, 56, 63, 64, 65, 124, 127, 137, 138

Kra86. Evangelos Kranakis. *Primality and Cryptography*. Wiley-Teubner Series in Computer Science, 1986. 21

Kra05a. Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005. 15, 17, 37, 38, 40, 45, 52, 126, 127

Kra05b. Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. pages 546–566, 2005. 50

KSJG10. Florian Kohlar, Jörg Schwenk, Meiko Jensen, and Sebastian Gajek. Secure bindings of SAML assertions to TLS sessions. In *ARES*, pages 62–69. IEEE Computer Society, 2010. 64

KSS13. Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DH and TLS-RSA in the standard model. *IACR Cryptology ePrint Archive*, 2013:367, 2013. 63, 64, 65

Küs06. Ralf Küsters. Simulation-based security with inexhaustible interactive turing machines. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy*, pages 309–320, 2006. 36

KY07. Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. 20(1):85–113, January 2007. 17, 110, 111, 121, 126

LJYP14. Benoit Libert, Marc Joye, Moti Yung, and Thomas Peters. Concise multi-challenge cca-secure encryption and signatures with almost tight security. In *ASIACRYPT 2014*, 2014. `https://eprint.iacr.org/2014/743.pdf`. 16

LLM07a. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger Security of Authenticated Key Exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec*, volume 4784 of *LNCS*, pages 1–16. Springer, 2007. 17, 37, 38, 40, 45, 126, 127, 129, 130

LLM07b. Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. pages 1–16, 2007. 52

LMQ+03. Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An efficient protocol for authenticated key agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003. 64

LP08a. J. Lee and C.S. Park. An efficient authenticated key exchange protocol with a tight security reduction. Cryptology ePrint Archive, Report 2008/345, 2008. `http://eprint.iacr.org/`. 130

LP08b. J. Lee and J.H. Park. Authenticated key exchange secure under the computational diffie-hellman assumption. Cryptology ePrint Archive, Report 2008/344, 2008. `http://eprint.iacr.org/`. 130

LSY+14a. Yong Li, Sven Schäge, Zheng Yang, Christoph Bader, and Jörg Schwenk. New modular compilers for authenticated key exchange. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 1–18, 2014. 18, 109

LSY+14b. Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. On the security of the pre-shared key ciphersuites of TLS. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 669–684, 2014. 18, 61

LSY+14c. Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. On the security of the pre-shared key ciphersuites of tls. Cryptology ePrint Archive, Report 2014/037, 2014. `http://eprint.iacr.org/`. 18, 61

MJ. Nikos Mavrogiannopoulos and Simon Josefsson. The GnuTLS Transport Layer Security library. Last updated 2013-03-22, http://gnutls.org. 16, 64

MQV95. Alfred Menezes, Minghua Qu, and Scott A. Vanstone. Some new key agreement protocols providing mutual implicit authentication. *SecondWorkshop on Selected Areas in Cryptography (SAC 95)*, 1995. 64

MS04. Alfred Menezes and Nigel P. Smart. Security of signature schemes in a multi-user setting. *Des. Codes Cryptography*, 33(3):261–274, 2004. 50, 51

MS13. Christopher Meyer and Jörg Schwenk. Lessons learned from previous SSL/TLS attacks - a brief chronology of attacks and weaknesses. *IACR Cryptology ePrint Archive*, 2013:49, 2013. 64

MSST98. D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408 (Proposed Standard), November 1998. Obsoleted by RFC 4306. 15

MSW10. Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. The TLS handshake protocol: A modular analysis. 23(2):187–223, April 2010. 64

MSW+14. Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 733–748, 2014. 15

MU09. Alfred Menezes and Berkant Ustaoglu. Comparing the pre- and post-specified peer models for key agreement. *IJACT*, 1(3):236–250, 2009. 45, 47

MvOV96. Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 21

Nat02. National Institute of Standards and Technology. FIPS 198, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standard (FIPS), Publication 198. Technical report, 2002. 74

NMO05. Waka Nagao, Yoshifumi Manabe, and Tatsuaki Okamoto. A universally composable secure channel based on the KEM-DEM framework. pages 426–444, 2005. 36

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. 1990. 96

Oka07. Tatsuaki Okamoto. Authenticated key exchange and key encapsulation in the standard model. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 474–484. Springer, 2007. 45, 47

Ope13. OpenSSL. The OpenSSL project, 2013. http://www.openssl.org. 16, 64

PA12. Kenneth G. Paterson and Nadhem J. AlFardan. Plaintext-recovery attacks against datagram TLS. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012. 15

Pau13. Paul Bakker. PolarSSL, 2013. https://polarssl.org/. 16, 64

Pee13. PeerSec Networks. MatrixSSL, 2013. http://www.matrixssl.org/. 16, 64

Pet13. Peter Gutmann. cryptlib, 2013. https://www.cs.auckland.ac.nz/~pgut001/cryptlib/. 16, 64

Pip98. D. Piper. The Internet IP Security Domain of Interpretation for ISAKMP. RFC 2407 (Proposed Standard), November 1998. Obsoleted by RFC 4306. 15

PRS11a. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 372–389, 2011. 15

PRS11b. Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. pages 372–389, 2011. 31

PS14. Kenneth G. Paterson and Mario Strefler. A practical attack against the HIVE hidden volume encryption system. *IACR Cryptology ePrint Archive*, 2014:901, 2014. 15

PUM11. L.Cogneau P. Urien and P. Martin. EMV support for TLS-PSK. draft-urien-tls-psk-emv-02, February 2011. 15, 16, 66

Res00. E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFC 5785. 15

Sch96. Bruce Schneier. *Applied cryptography - protocols, algorithms, and source code in C (2. ed.)*. Wiley, 1996. 21

Sch11. Sven Schäge. Tight proofs for signature schemes without random oracles. pages 189–206, 2011. 16

SEVB10. Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A new security model for authenticated key agreement. pages 219–234, 2010. 126, 130, 161

Sho99a. Victor Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. 17, 38, 45

Sho99b. Victor Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. 64

Sho04. Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`. 36, 72, 82, 103, 116, 119

Tod13. Todd Ouska. CyaSSL, 2013. `http://www.wolfssl.com/yaSSL/Home.html`. 16, 64

Uri10. Pascal Urien. Introducing TLS-PSK authentication for EMV devices. In *CTS*, pages 371–377, 2010. 15, 16

Ust08. Berkant Ustaoglu. Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. *Des. Codes Cryptography*, 46(3):329–342, 2008. 64, 130

Ust09. Berkant Ustaoglu. Comparing sessionstatereveal and ephemeralkeyreveal for Diffie-Hellman protocols. pages 183–197, 2009. 41

WG06. M. Wasserman and T. Goddard. Using the NETCONF Configuration Protocol over Secure SHell (SSH). RFC 4742 (Proposed Standard), December 2006. Obsoleted by RFC 6242. 15

Yan13. Zheng Yang. Strongly secure authenticated key exchange in standard model. 2013. 109

ZY10. Jianying Zhou and Moti Yung, editors. *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, volume 6123 of *Lecture Notes in Computer Science*, 2010. 126

Curriculum Vitae

Personal Data

Name: Yong li

Email: yong.li@rub.de

Education

2012-2015 Ruhr-University Bochum, Germany
Ph.D. in computer science
at Network and Data Security Group
lead by Prof. Jörg Schwenk

2002-2008 Ruhr-University Bochum, Germany
Diploma degree in engineering
IT-Security

Academic Publications

Conference: CANS'13
Strongly Secure One-round Group Authenticated Key Exchange in the Standard Model
Yong Li, Zheng Yang

Conference: PKC'14
On the Security of the Pre-Shared Key Ciphersuites of TLS
Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, Jörg Schwenk

Conference ACNS'14
New Modular Compilers for Authenticated Key Exchange
Yong Li, Sven Schäge, Zheng Yang, Christoph Bader, Jörg Schwenk

Conference TCC'15
Tightly-Secure Authenticated Key Exchange
Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, Yong Li

ePrint 2015/374
On the Impossibility of Tight Cryptographic Reductions
Christoph Bader, Tibor Jager, Yong Li, Sven Schäge

Chapter A

Appendix: No-Match Attacks on AKE Protocols

In this section, we show several examples of existing AKE protocols that claim to be secure under a security definition based on MC-based session identifier (sID) but where the security proof is actually flawed.

A.1 The Jeong-Katz-Lee Protocol $\mathcal{TS}3$

This $\mathcal{TS}3$ protocol is not immune to our no-match attack. We first give an overview of their security model. Then, we describe the $\mathcal{TS}3$ protocol. Finally, we show our no-match attacks against this protocol. More information on $\mathcal{TS}3$ protocol, see [JKL04]

- **Jeong-Katz-Lee Model.**

  In Jeong-Katz-Lee model (named JKL04 Model), they consider forward secrecy (FS) for two-party key exchange protocol. We will not describe the JKL04 model in detail here. Instead we only point out some of its main features since these notions are necessary to explain our no-match attacks. For more information on JKL04 model, see [JKL04].

  ADVERSARY MODEL. Without loss of generality, each party P_i is defined by a set of oracles, π_i^s which represents the s-th protocol instance of player P_i, where $s \in [d]$ is an index for a range such that $d \in \mathbb{N}$. An active adversary $\mathcal{A}$ is able to issue the following queries:

  - $\mathsf{Initiate}(i, j)$: P_i to initiate execution of the protocol with P_j;
  - $\mathsf{Execute}(\mathsf{P}_i, \mathsf{P}_j)$: Parties P_i and P_j execute the protocol;
  - $\mathsf{Send}(\pi_i^s, m)$, $\mathsf{Reveal}(\pi_i^s)$, $\mathsf{Corrupt}(\mathsf{P}_i)$ and $\mathsf{Test}(\pi_i^s)$ queries are similar to those as described in 3.4.2.

  A session identifier of an protocol instance sID_i^s is a string different from those of all other sessions in the system. In paper [JKL04], session identifier (sID) is defined as the transcript of the conversations between comminication parties P_i and P_j in protocol. JKL04 partnership definition is based on the notion of session identifiers

(sIDs). The following definition describes the definition of partnership in the JKL04 model.

Definition A.1 (JKL04 Model Partnership). *Consider oracle π_i^s of party P_i. We say that π_i^s and π_j^t are partnered if $\mathsf{sID}_i^s = \mathsf{sID}_j^t$, P_i is the partner of π_j^t and P_j is the partner of π_i^s.*

Definition A.2 describes freshness, which depends on the respective notion of partnership described above.

Definition A.2 (JKL04 Model Freshness). *An oracle π_i^s is fresh if the following conditions are true:*

- *The adversary $\mathcal{A}$ has not queried $\mathsf{Reveal}(\pi_i^s)$,*
- *The adversary $\mathcal{A}$ has not queried $\mathsf{Reveal}(\pi_j^t)$, if π_i^s is partnered with π_j^t.*
- *If the adversary $\mathcal{A}$ has queried $\mathsf{Corrupt}(\mathsf{P}_i)$ or $\mathsf{Corrupt}(\mathsf{P}_j)$, then it has never queried $\mathsf{Send}(\pi_i^s, m)$.*

SECURITY GAME. The security in the model is defined using the security experiment, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The security experiment is similar to that described in 3.4.3.

Definition A.3 (JKL04 Model Security Definition). *We say that a two-party key exchange protocol Σ is (t, ϵ)-secure in JKL06 model, if for all adversaries $\mathcal{A}$ running the above security game within time t, it holds that: When $\mathcal{A}$ returns b' such that*

- *$\mathcal{A}$ has issued a Test-query to oracle π_i^s, and*
- *the oracle π_i^s is fresh throughout the security game,*

then the probability that b' equals the bit b sampled by the Test-query is bounded by

$$|\Pr[b = b'] - 1/2| \leq \epsilon.$$

The Joeng-Katz-Lee Key Agreement Protocol $\mathcal{TS}3$ is shown in Figure A.1. They presented a formal proof of security in the JKL04 model described above.

- **Protocol Description.**
  - Setup: Assume P_i wants to establish a session key with P_j. Party P_c has a public-/private key pair: $(pk_c = g^{x_c}, sk_c = x_c)$, $c \in \{i, j\}$. Let MAC be an strongly unforgeable message authentication code (MAC) scheme.
  - Round 1: Party P_i computes a MAC key $k_{i,j} = pk_j^{x_i} = g^{x_i x_j}$. Then, P_i selects a random number $\alpha_i \xleftarrow{\$} \mathbb{Z}_q$, computes g^{α_i} and calculates an authentication tag τ_i by evaluating some probabilistic $\mathsf{MAC.Tag}$ function under the established MAC key $k_{i,j}$ with internal random coins r_i over the publicly known input $(i||j||g^{\alpha_i})$, $\tau_i := \mathsf{MAC.Tag}(k_{i,j}, r_i; i||j||g^{\alpha_i})$. P_i sends g^{α_i} and τ_i to P_j. Party P_j behaves as similarly as P_i.

– Computation of session key: Party P_i verifies the tag of the received message. If the check is passed, P_i computes the session key $\mathsf{K}_{i,j} := g^{\alpha_i\alpha_j}$. The session identifier $\mathsf{sID}_i = a^{\alpha_i} \| \tau_i \| a^{\alpha_j} \| \tau_j$. Party P_j behaves as similarly as P_i.

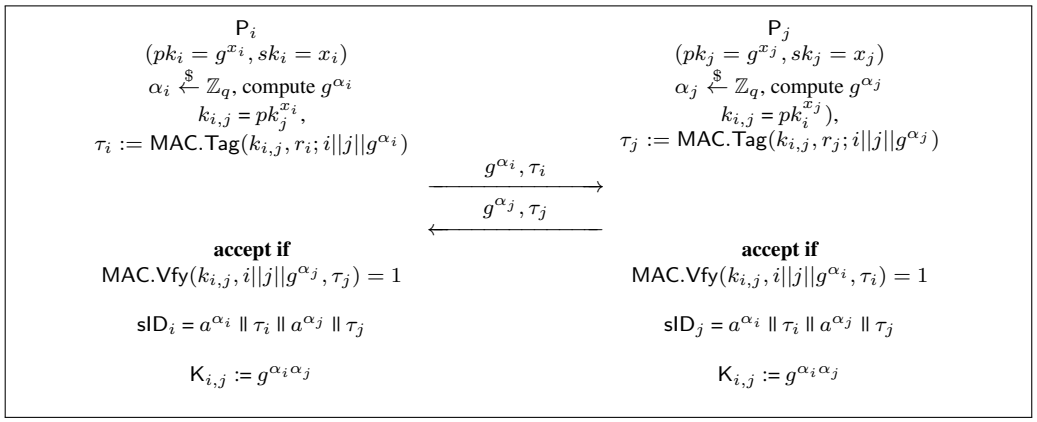


Fig. A.1. The description of $\mathcal{TS}3$ Protocol

- **No-Match Attack against Joeng-Katz-Lee ($\mathcal{TS}3$) Protocol.**

  Figure A.2 shows our no-match attack against the $\mathcal{TS}3$ Protocol. The adversary $\mathcal{A}$ performs the actions as follows:

  – According to the protocol specification, party P_i computes g^{α_i} and an authentication tag τ_i and sends (g^{α_i}, τ_i) to P_j. Party P_j behaves as similarly as P_i.

  – The adversary $\mathcal{A}$ makes a $\mathsf{Corrupt}(\mathsf{P}_i)$ query and obtains the long-term private key of party P_i, $sk_i = x_i$.

  – Assume that the adversary $\mathcal{A}$ controls all communications between parties and deletes the τ_i generated by P_i. Then $\mathcal{A}$ computes $k_{i,j} = pk_j^{x_i}$ using the private key x_i and generates a fresh authentication tag τ^* with internal random coins r^* ($r^* \neq r_i$), $\tau^* := \mathsf{MAC.Tag}(k_{i,j}, r^*; i||j||g^{\alpha_i})$. Finally, $\mathcal{A}$ sends (g^{α_i}, τ^*) to P_j. Note that the adversary $\mathcal{A}$ has queried $\mathsf{Corrupt}(\mathsf{P}_i)$, then it has never queried $\mathsf{Send}(\pi_i^s, m)$. According to the *fresh* definition oracle π_i^s is fresh.

  – Computation of session key: Parties P_i and P_j accept and compute the same session key $\mathsf{K}_{i,j} := g^{\alpha_i\alpha_j}$.

  – The adversary $\mathcal{A}$ queries $\mathsf{Test}(\pi_i^s)$ and gets K_b from oracle π_i^s. Then, $\mathcal{A}$ makes $\mathsf{Reveal}(\pi_j^t)$ and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership π_i^s and π_j^t are not partnered, i.e., $\mathsf{sID}_i^s \neq \mathsf{sID}_j^t$. Therefore, $\mathcal{A}$ is allowed to make $\mathsf{Reveal}(\pi_j^t)$ to oracle π_j^t. Finally, If $\mathsf{K}_{i,j} = \mathsf{K}_b$, then $\mathcal{A}$ returns $b = 1$; Otherwise, returns b = 0.

Remark A.4. Since the adversary $\mathcal{A}$ forces the same session key $\mathsf{K}_{i,j}$ to be computed in two non-matching sessions, then it can choose the former as its test session and win the distinguishing game.

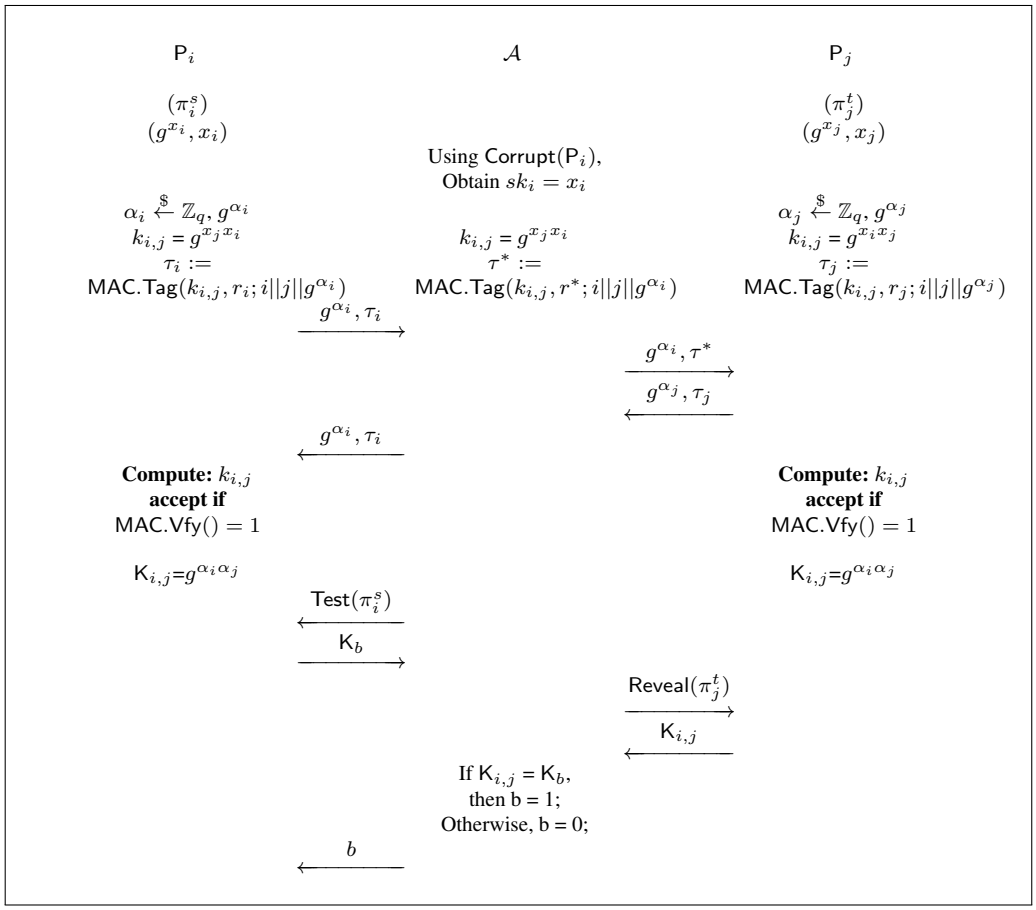


Fig. A.2. No-Match Attack against $\mathcal{TS}3$ Protocol

A.2 The Jeong-Kwon-Lee KAM Protocol

The key agreement protocol KAM introduced by Jeong, Kwon and Lee [JKL06] shown in Figure A.3 carries a formal proof of security in their model. In this subsection, we first give an informal overview of their security model. Then, we provide a description of the KAM protocol. Finally, we present a no-match attack that can be launched against this protocol.

- **Jeong-Kwon-Lee Model.**

In the security model used by the Jeong-Kwon-Lee paper (hereafter named JKL06 model), the adversary $\mathcal{A}$ is in control of all communications between parties and allowed to intercept, delay, delete, and modify any messages.The JKL06 model also considers perfect forward secrecy (PFS) and key compromise impersonation (KCI) attacks. We only give an informal overview of the Jeong-Kwon-Lee model. More details can be found in [JKL06].

ADVERSARY MODEL. Assume that each party holds a pair of private and public keys. As described above, party P_i is also defined by a set of oracles, $\{\pi_i^s\}$. An active adversary $\mathcal{A}$ is able to issue Initiate, Send, Reveal, Corrupt, RevealState and Test queries. The mentioned queries are similar to those as described in A.1.

The security definition depends on the notions of partnership of oracles. The JKL06 partnership definition is based on the notion of session identifiers (sIDs). Before we begin to describe the definition of partnership, we first give the original definition of session identifier sID as described in [JKL06].

Definition A.5 (JKL06 Model Session Identifier). *A session identifier* sID_i^s *is suggested to be the concatenation of all messages sent and received by a particular protocol instance* π_i^s*, where the order of these message is determined by the lexicographic ordering.*

The following definition A.6 describes the definition of partnership in the JKL06 model.

Definition A.6 (JKL06 Model Partnership). *Consider oracle* π_i^s *of party* P_i*. The partner of this oracle is party* P_j*. We say that two oracles* π_i^s *and* π_j^t *are partnered, if the following conditions are true:* $\mathsf{sID}_i^s = \mathsf{sID}_j^t$*,* P_i *is the partner of* π_j^t *and* P_j *is the partner of* π_i^s*.*

Any key exchange protocol should satisfy the correctness condition, i.e. two partnered oracles have accepted the same session key.

Definition A.7 describes freshness, which depends on the respective notion of partnership described above. The JKL06 model considers perfect forward secrecy (PFS), key compromise impersonation (KCI) and the reveal of intermediate random values used when computing the session key. However, the following definition of freshness we concentrate on the notion of key compromise impersonation (KCI) attacks since this notion is necessary to launch and explain our no-match attack. Informally, the adversary is allowed to ask a Corrupt query to A (the owner of the Test oracle). More information can be found in [JKL06].

Definition A.7 (JKL06 Model Freshness). *An oracle* π_i^s *is fresh if the following conditions are true:*

– *The adversary* $\mathcal{A}$ *has not queried* $\mathsf{Reveal}(\pi_i^s)$*,*

– *The adversary $\mathcal{A}$ has not queried* $\mathsf{Reveal}(\pi_j^t)$*, if* π_i^s *is partnered with* π_j^t*.*
– *The adversary $\mathcal{A}$ does not control* P_j*, if* P_j *is the partner of* π_i^s*. In other words,* P_j *is not generated by the adversary $\mathcal{A}$.*
– *KCI (Key Compromise Impersonation): the adversary $\mathcal{A}$ has queried all* $\mathsf{Send}(\pi_i^s, m)$ *after* $\mathsf{Corrupt}(\mathsf{P}_i)$ *and its partner* P_j *has not corrupted.*

SECURITY GAME. The security is defined using a security experiment, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The security experiment is similar to that described in 3.4.3. The security definition of the JKL06 model is essentially that of Appendix A.3.

The Jeong-Kwon-Lee Key Agreement Protocol KAM is shown in Figure A.3. They presented a formal proof of security in the JKL06 model described above.

- **Protocol Description.**
  – Setup: Each party P_c has a public-/private key pair: $(pk_c = g^{x_c}, sk_i = x_c)$, $c \in \{i, j\}$. Let MAC be a strongly unforgeable message authentication code (MAC) scheme. Let Hash be a hash function such that Hash: $\{0,1\}^* \rightarrow \{0,1\}^\kappa$, where κ is a security parameter.
  – Round 1: Party P_i selects a random number $\alpha_i \xleftarrow{\$} \mathbb{Z}_q$, computes $\Gamma_i = g^{\alpha_i}$ and sends Γ_i to P_j. Party P_j behaves as similarly as P_i.
  – Round 2: Party P_i performs a group membership test of Γ_j. If the test is successful, P_i computes the MAC key $k_{i,j} = \mathsf{Hash}(\Gamma_j^{x_i}) = \mathsf{Hash}(g^{\alpha_j x_i})$. Then, P_i calculates an authentication tag τ_i by evaluating some probabilistic MAC.Tag function under the established MAC key $k_{i,j}$ with internal random coins r_i over the publicly known input $(i||j||g^{\alpha_i})$, $\tau_i := \mathsf{MAC.Tag}(k_{i,j}, r_i; i||j||g^{\alpha_i})$, where r_i is defined as internal randomness. P_i sends τ_i to P_j. Party P_j behaves as similarly as P_i.
  – Computation of session key: Party P_i computes the MAC key $k_{j,i} = \mathsf{Hash}(pk_j^{\alpha_i}) = \mathsf{Hash}(g^{x_j \alpha_i})$ and verifies $\mathsf{MAC.Vfy}(k_{j,i}, i||j||g^{\alpha_j}, \tau_j) = 1$. If the check is passed, P_i computes the session key $\mathsf{K}_{i,j} := \mathsf{Hash}(pk_j^{x_i}) \oplus \mathsf{Hash}(\Gamma_j^{\alpha_i})$. Party P_j behaves as similarly as P_i.
- **No-Match Attack against the Jeong-Kwon-Lee (KAM) Protocol.**

Figure A.7 shows our no-match attack against the KAM protocol. The adversary $\mathcal{A}$ performs the following actions:

– The adversary $\mathcal{A}$ makes a $\mathsf{Corrupt}(\mathsf{P}_i)$ query and obtains the long-term private key of party P_i, $sk_i = x_i$.
– According to the protocol specification, party P_i computes Γ_i and send Γ_i to P_j. Party P_j behaves as similarly as P_i.
– Then, party P_i computes an authentication tag τ_i and sends τ_i to P_j. Since the adversary $\mathcal{A}$ controls the entire communication flows between the parties, it can

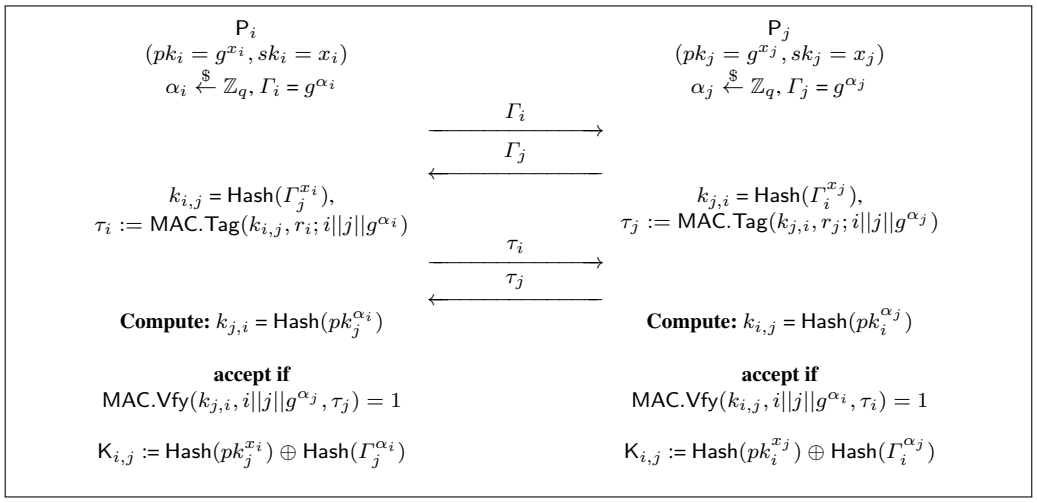


Fig. A.3. Description of the KAM Protocol

easily intercept and drop the last message τ_i generated by P_i. Then $\mathcal{A}$ computes $k_{i,j} = \mathsf{Hash}(\Gamma_j^{x_i})$ using the private key x_i and generates a fresh authentication tag τ^* with internal random coins r^* ($r^* \neq r_i$), $\tau^* := \mathsf{MAC.Tag}(k_{i,j}, r^*; i||j||g^{\alpha_i})$. Finally, $\mathcal{A}$ sends τ^* to P_j.

– Party P_j behaves similar to P_i. P_j computes an authentication tag τ_j and sends τ_j to P_i.

– Computation of session key: parties P_i and P_j accept and compute compute the same session key $\mathsf{K}_{i,j} := \mathsf{Hash}(pk_j^{x_i}) \oplus \mathsf{Hash}(\Gamma_j^{\alpha_i})$.

– The adversary $\mathcal{A}$ queries $\mathsf{Test}(\pi_i^s)$ and gets K_b from oracle π_i^s. Then, $\mathcal{A}$ asks $\mathsf{Reveal}(\pi_j^t)$ and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership π_i^s and π_j^t are not partnered, i.e. $\mathsf{sID}_i^s \neq \mathsf{sID}_j^t$. Therefore, $\mathcal{A}$ is allowed to make $\mathsf{Reveal}(\pi_j^t)$ to oracle π_j^t. Oracle π_i^s is fresh. Finally, if $\mathsf{K}_{i,j} = \mathsf{K}_b$, then $\mathcal{A}$ returns $b = 1$; Otherwise, returns b = 0.

A.3 Beyond eCK: Perfect Forward Secrecy under Actor Compromise and Ephemeral-Key Reveal

- **Cremers-Feltz Model: M-PFS.** In original paper [CF12] there are four new game-based security models for key exchange protocols, M^w, eCK^w, M-PFS and eCK-PFS. We will only describe the M-PFS model (no EphemeralKey query) here and point out some of its main features relevant to perfect forward secrecy (PFS) since these notions are necessary to explain our no-match attacks. For more information on all these models, see [CF12].

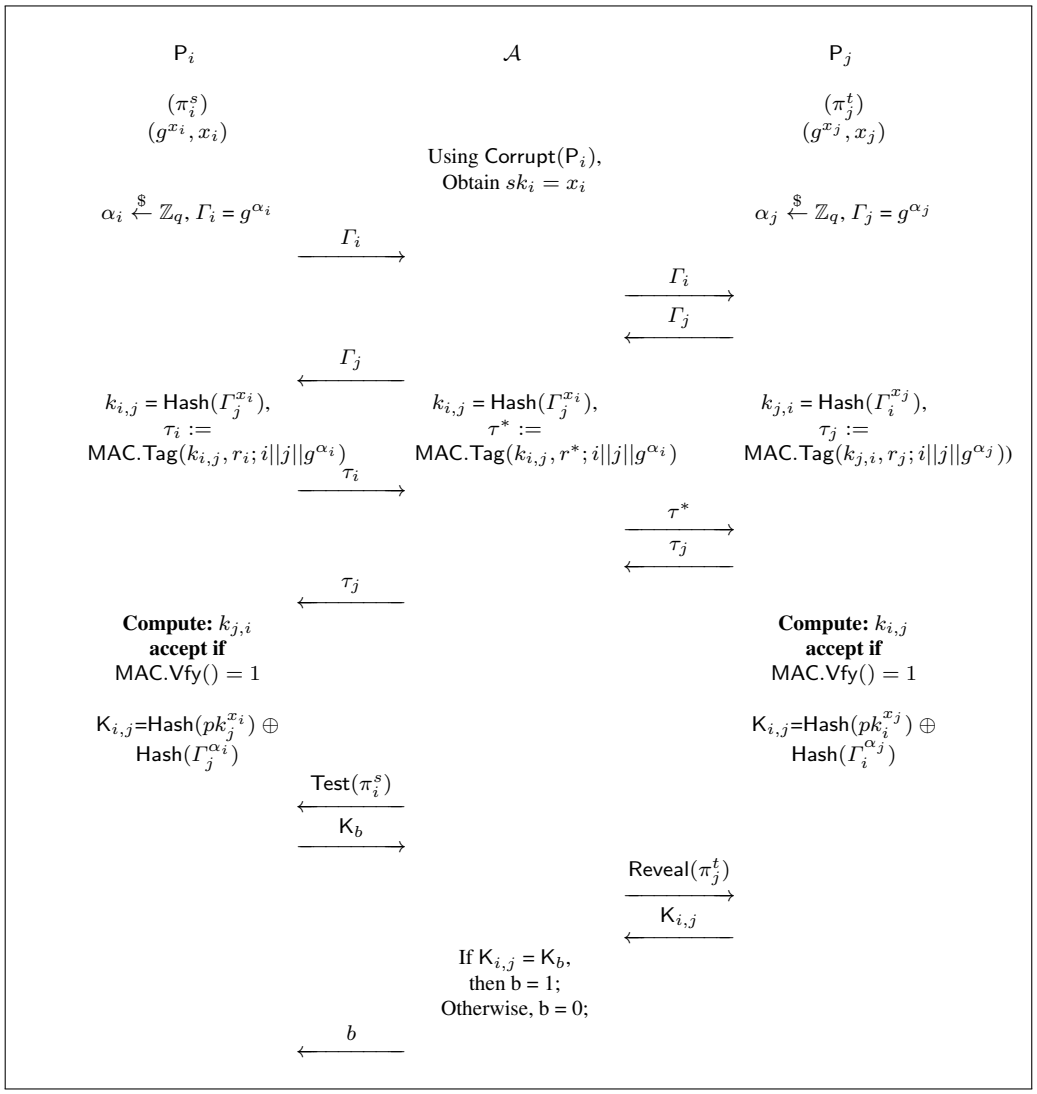


Fig. A.4. No-Match Attack against KAM Protocol by Corrupt-query

ADVERSARY MODEL. For simplicity, we use the same notions as described in the original paper. Party P_i is defined by a set of sessions. Let session s at party P_i denote as the tuple $(\mathsf{P}_i, s) \in \mathcal{P} \times \mathbb{N}$. An active adversary $\mathcal{A}$ is able to issue Send, SessionKey, Corrupt and Test queries. The mentioned queries are similar to those as described in 3.4.2.

Let the variables s_{actor}, s_{peer} denote the identities of the actor and intended peer of the session s. Let s_{role} be the role of the session execution (i.e., initiator or responder) and s_{sent}, s_{recv} be the concatenation of timely ordered messages as sent and received by s_{actor} during the protocol instance s. Each session s is associated with a quintuple of variables (s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv}).

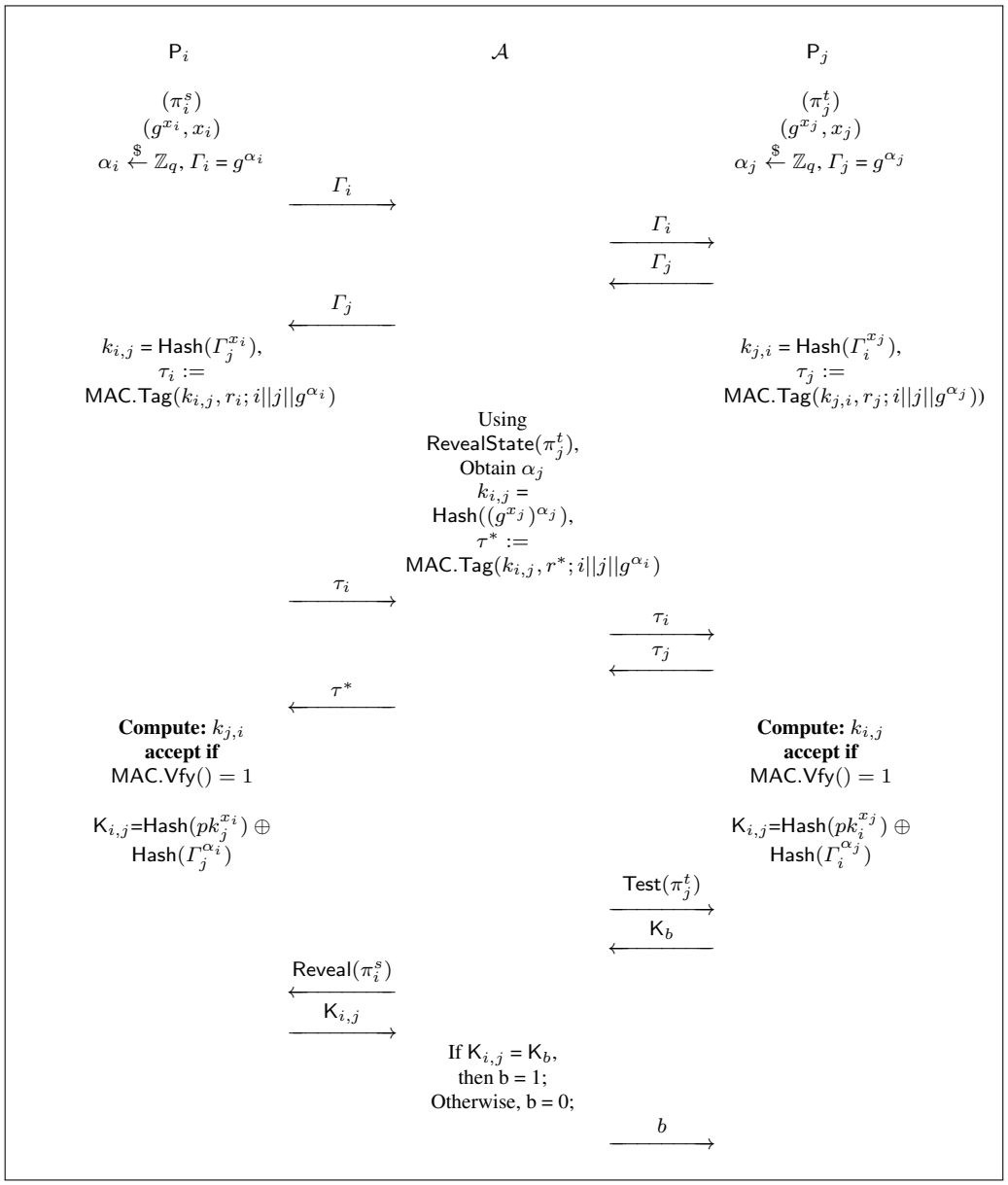


Fig. A.5. No-Match Attack against KAM Protocol by RevealState-query

Definition A.8 (Cremers-Feltz12 Model Matching Sessions). *Two completed session s and s' are said to be matching if $s_{actor} = s'_{peer}$ and $s_{peer} = s'_{actor}$ and $s_{sent} = s'_{recv}$ and $s'_{sent} = s_{recv}$ and $s_{role} \neq s'_{role}$.*

Cremers and Feltz in paper [CF12] presented a new concept of *origin session*. It allows the adversary to corrupt the long-term secret key of the intended communication partner of the test session s if an origin-session s' for test session s exists.

Definition A.9 (Cremers-Feltz12 Model Origin-Session). *A (possibly incomplete) session s' is an origin-session for a completed session s when $s'_{sent} = s_{recv}$.*

Definition A.10 describes freshness, which depends on the respective notions of *matching sessions* and *origin-session* described above.

Definition A.10 (Cremers-Feltz12 Model Freshness). *A completed session s is said to be fresh if the following conditions are true:*

- *The adversary $\mathcal{A}$ has not queried* $\mathsf{SessionKey}(s)$,
- *The adversary $\mathcal{A}$ has not queried* $\mathsf{SessionKey}(s^*)$, *if s^* matches s,*
- *If there exists no origin-session for session s, then $\mathcal{A}$ does not query a* $\mathsf{Corrupt}(s_{peer})$ *before the completion of session s.*

SECURITY GAME. The security in the model is defined using the security experiment, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The security experiment is similar to that as described in 3.4.3.

Definition A.11 (Model Security Definition). *The security definition is similar to that described in A.3.*

For similarly, we only describe the SIG(NAXOS) protocol as an example in Figure A.10 and show our no-match attack against this protocol.

- **Protocol Description.**
  - Setup: Each party P_c has two independent valid long-term secret/public key pairs, one key pair from protocol NAXOS: $(pk_c = g^{x_c}, sk_i = x_c)$ and one pair $(pk_c^{\mathsf{sig}}, sk_c^{\mathsf{sig}})$ for a randomized digital signature scheme, where $c \in \{i, j\}$. Let H_1:$\{0,1\}^* \rightarrow \mathbb{Z}_p$ and H_2:$\{0,1\}^* \rightarrow \{0,1\}^\kappa$ be two hash functions, where κ is a security parameter. Let SIG be a signature scheme (SUF-CMA).
  - Party P_i selects a random number $\alpha_i \xleftarrow{\$} \mathbb{Z}_q$, computes $\Gamma_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. Then, P_i computes a signature σ_i by evaluating some probabilistic $\mathsf{SIG.Sign}$ function under the private key sk_i^{sig} with internal random coins r_i over the publicly known input $(\Gamma_i||\mathsf{P}_j)$, $\sigma_i := \mathsf{SIG.Sign}(sk_i^{\mathsf{sig}}, r_i; \Gamma_i||\mathsf{P}_j)$. Finally, P_i sends (Γ_i, σ_i) to P_j.
  - Party P_j behaves as similarly as P_i. Party P_j selects a random number $\alpha_j \xleftarrow{\$} \mathbb{Z}_q$, computes Γ_j and σ_j, and sends (Γ_j, σ_j) to P_i.
  - Computation of session key: Party P_i verifies $\mathsf{SIG.Vfy}(pk_i^{\mathsf{sig}}, \Gamma_j||\Gamma_i||\mathsf{P}_i, \sigma_j) = 1$. If the check is passed, P_i computes the session key as follows: $\mathsf{K}_{i,j} := \mathsf{H}_2(\Gamma_j^{sk_i}, pk_j^{\mathsf{H}_1(\alpha_i, sk_i)}, \Gamma_j^{\mathsf{Hash}_1(\alpha_i, sk_i)}, i, j)$. Party P_j behaves as similarly as P_i,
- **No-Match Attack against the SIG(NAXOS) Protocol.**

  Figure A.7 shows our no-match attack against the KAM Protocol. The adversary $\mathcal{A}$ performs the actions as follows:

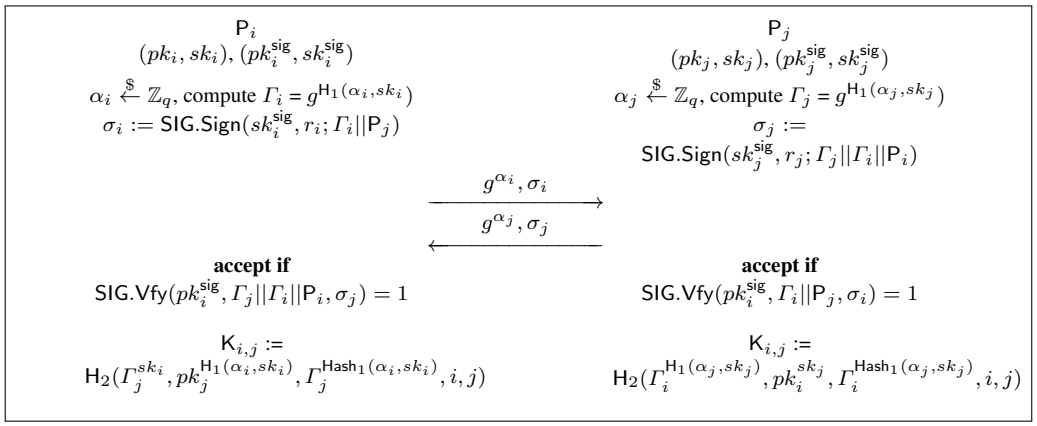


Fig. A.6. The description of the SIG(NAXOS) Protocol

- The adversary $\mathcal{A}$ makes a $\mathsf{Corrupt}(\mathsf{P}_i)$ query and obtains the long-term private key of party P_i, sk_i^{sig}.
- According to the protocol specification, Party P_i computes $\Gamma_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. Then, P_i computes a signature σ_i with internal random coins r_i over $(\Gamma_i || \mathsf{P}_j)$, $\sigma_i := \mathsf{SIG.Sign}(sk_i^{\mathsf{sig}}, r_i; \Gamma_i || \mathsf{P}_j)$. Finally, P_i sends (Γ_i, σ_i) to P_j.
- Assume that the adversary $\mathcal{A}$ controls all communications between parties and deletes the σ_i generated by P_i. Then $\mathcal{A}$ computes a fresh signature σ^* with internal random coins r^* ($r^* \neq r_i$), $\sigma_i := \mathsf{SIG.Sign}(sk_i^{\mathsf{sig}}, r^*; \Gamma_i || \mathsf{P}_j)$. Finally, $\mathcal{A}$ sends (Γ_i, σ^*) to P_j.
- Party P_j behaves as similarly as P_i. P_j computes (Γ_j, σ_j) and sends them to P_i.
- Computation of session key: Parties P_i and P_j accept and compute the same session key $\mathsf{K}_{i,j}$.
- The adversary $\mathcal{A}$ queries $\mathsf{Test}(s)$ and gets K_b from s. Then, $\mathcal{A}$ makes $\mathsf{SessionKey}(s')$ and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership s and s' are not partnered, i.e., $s_{sent} \neq s'_{recv}$. Therefore, $\mathcal{A}$ is allowed to make the query $\mathsf{SessionKey}(s')$. Session s is also fresh. Finally, if $\mathsf{K}_{i,j} = \mathsf{K}_b$, then $\mathcal{A}$ returns $b = 1$; Otherwise, returns b = 0.

A.4 Signed Diffie-Hellman under the NAXOS Transformation

- **Extended Canetti-Krawczyk Model (eCK).** In Paper [SEVB10], Sarr et al. presented a signed Diffie-Hellman protocol using NAXOS transformation, and claimed this protocol is secure in the eCK model. We use the same description of eCK as described in 3.2.3.
- **Protocol Description.**

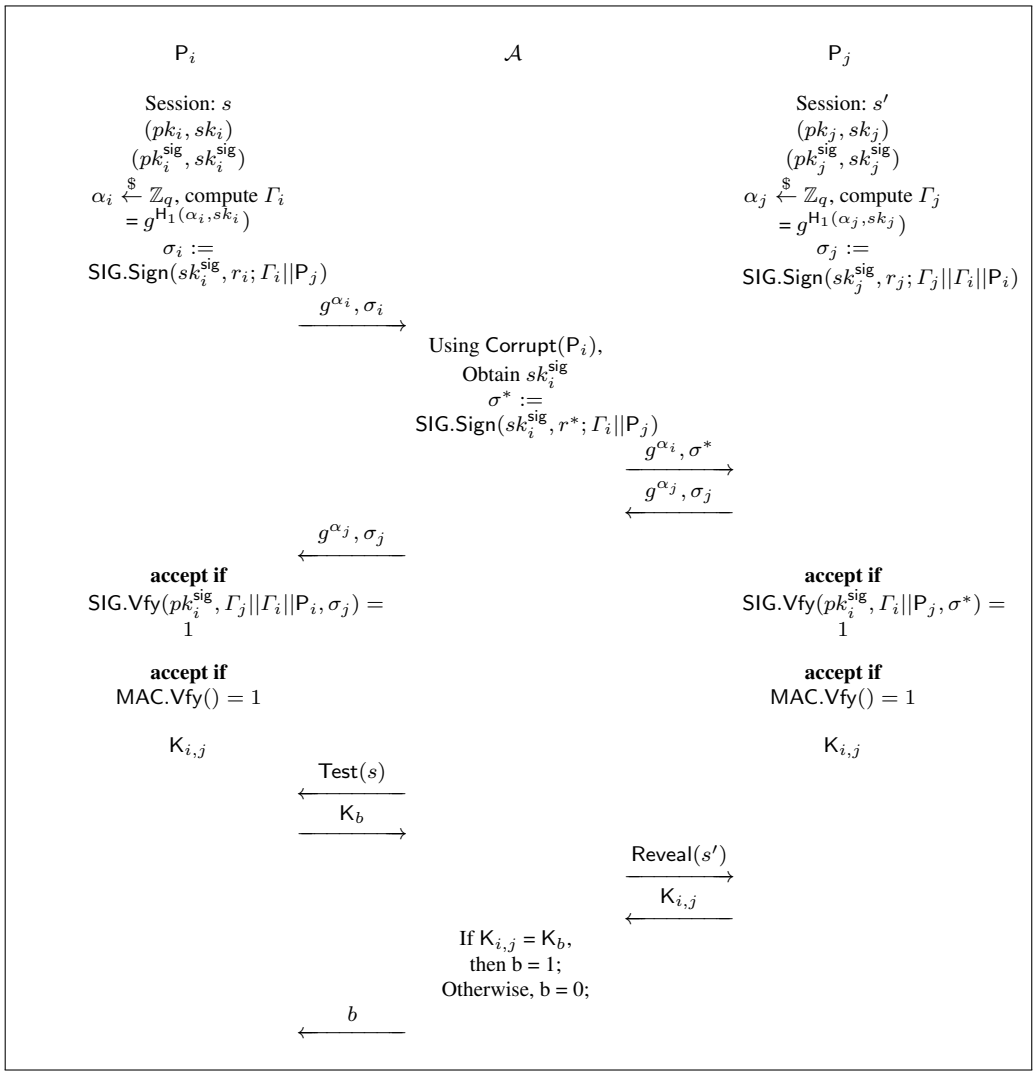


Fig. A.7. No-Match Attack against SIG(NAXOS) Protocol

- Setup: Each party P_c has two independent valid long-term secret/public key pairs, one pair: (pk_c, sk_i) and one pair (pk_c^{sig}, sk_c^{sig})) for a randomized digital signature scheme, where $c \in \{i, j\}$. Let H_1:$\{0,1\}^* \rightarrow \mathbb{Z}_p$ and H_2:$\{0,1\}^* \rightarrow \{0,1\}^\kappa$ be two hash functions, where κ is a security parameter. Let SIG be a signature scheme.
- Party P_i selects a random number $\alpha_i \xleftarrow{\$} [1, q-1]$, computes $\mathsf{X}_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. Then, P_i computes a signature σ_i by evaluating some probabilistic $\mathsf{SIG.Sign}$ function under the private key sk_i^{sig} with internal random coins r_i over ($\mathsf{P}_j||\mathsf{X}_i$), $\sigma_i := \mathsf{SIG.Sign}(sk_i^{\mathsf{sig}}, r_i; \mathsf{P}_j||\mathsf{X}_i)$. Finally, P_i sends (P_j, X_i, σ_i) to party P_j.
- Party P_j verifies that $\mathsf{X}_i \in \mathcal{G}^*$ and σ_i is valid. If the check is not passed, then P_j rejects. Otherwise, party P_j behaves as similarly as P_i. Party P_j selects a random number $\alpha_j \xleftarrow{\$} [1, q-1]$, computes $\mathsf{Y}_j = g^{\mathsf{H}_1(\alpha_j, sk_j)}$. Then, P_j computes a signature

σ_j, $\sigma_j := \mathsf{SIG.Sign}(sk_j^{\mathsf{sig}}, r_j; \mathsf{Y}_j||\mathsf{P}_j||\mathsf{X}_i)$. Finally, P_j sends (Y_j, P_j, X_i, σ_j) to P_i. P_j computes the session key $\mathsf{K}_{i,j} := \mathsf{H}_2(\mathsf{X}_i^{\mathsf{H}_1(\alpha_j, sk_j)})$.

– Party P_i verifies that $\mathsf{Y}_j \in \mathcal{G}^*$ and σ_j is valid. If the check is passed, P_i computes the session key $\mathsf{K}_{i,j} := \mathsf{H}_2(\mathsf{Y}_j^{\mathsf{H}_1(\alpha_i, sk_i)})$.

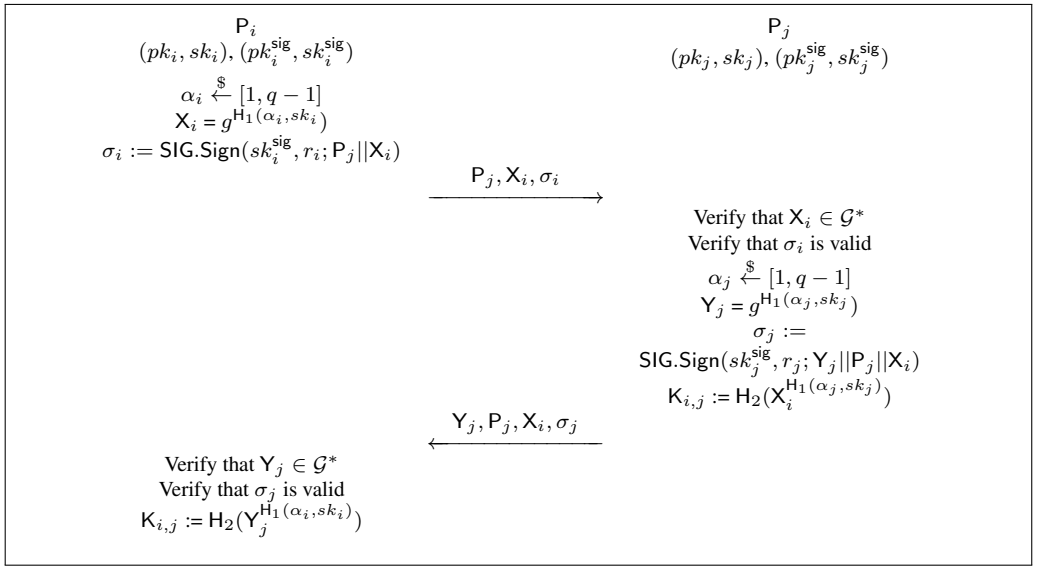


Fig. A.8. The description of the $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ Protocol

- **No-Match Attack against the $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ Protocol.**

  Figure A.9 shows our no-match attack against the $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ Protocol. The adversary $\mathcal{A}$ performs the actions as follows:

  – The adversary $\mathcal{A}$ makes a $\mathsf{Corrupt}(\mathsf{P}_i)$ query and obtains the long-term private key of party P_i, sk_i^{sig}.

  – According to the protocol specification, Party P_i computes $\mathsf{X}_i = g^{\mathsf{H}_1(\alpha_i, sk_i)}$. and a signature σ_i with internal random coins r_i over ($\mathsf{P}_j||\mathsf{X}_i$). Finally, P_i sends (P_j, X_i, σ_i) to P_j.

  – Assume that the adversary $\mathcal{A}$ controls all communications between parties and deletes the σ_i generated by P_i. Then $\mathcal{A}$ computes a fresh signature σ^* with internal random coins r^* ($r^* \neq r_i$), $\sigma^* := \mathsf{SIG.Sign}(sk_i^{\mathsf{sig}}, r^*; \mathsf{P}_j||\mathsf{X}_i)$. Finally, $\mathcal{A}$ sends (P_j, X_i, σ^*) to P_j.

  – Party P_j verifies that $\mathsf{X}_i \in \mathcal{G}^*$ and σ_i is valid. If the check is passed, party P_j behaves as similarly as P_i. Party P_j computes Y_j and σ_j and sends (Y_j, P_j, X_i, σ_j) to P_i. P_j computes the session key $\mathsf{K}_{i,j} := \mathsf{H}_2(\mathsf{X}_i^{\mathsf{H}_1(\alpha_j, sk_j)})$.

– Party P_i verifies that $\mathsf{Y}_j \in \mathcal{G}^*$ and σ_j is valid. If the check is passed, P_i computes the session key $\mathsf{K}_{i,j} := \mathsf{H}_2(\mathsf{Y}_j^{\mathsf{H}_1(\alpha_i, sk_i)})$. Note that P_i and P_j have the same session key $\mathsf{K}_{i,j}$.
– The adversary $\mathcal{A}$ queries $\mathsf{Test}(\pi_i^s)$ and gets K_b from oracle π_i^s. Then, $\mathcal{A}$ makes $\mathsf{SessionKey}(\pi_j^t)$ and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership of eCK model π_i^s and π_j^t are not partnered, i.e., $\mathsf{sID}_i^s \neq \mathsf{sID}_j^t$. Therefore, $\mathcal{A}$ is allowed to make $\mathsf{SessionKey}(\pi_j^t)$ to oracle π_j^t. Oracle π_i^s is fresh. Finally, if $\mathsf{K}_{i,j} = \mathsf{K}_b$, then $\mathcal{A}$ returns $b = 1$; Otherwise, returns b = 0.

A.5 The PKE-based Key Exchange Protocol π

- **Continuous After-the-Fact Leakage Model (CAFL model).** In paper [ABS14], Alawatugoda et al. introduced a new leakage model for key exchange protocols. We informally give the description of the CAFL model. For simplicity, we use the same notions as described in the original paper.

  ADVERSARY MODEL. Party P_i is defined by a set of sessions. Let $\pi_{i,j}^s$ represent the $s\text{-}th$ session at party P_i, with intended partner P_j. An active adversary $\mathcal{A}$ is able to issue Send, SessionKey, Corrupt, EphemeralKey and Test queries. The mentioned queries are similar to those as described in 3.4.2. Note that $\mathcal{A}$ can also use Send query to activate a new protocol session as an initiator with m and f. More information on the leakage function f, see [ABS14]. By issuing leakage function f embedded with the Send query $\mathcal{A}$ can get λ-bounded amount of leakage information about the long-term secret key of party.

  Definition A.12 (CAFL Model Partnership). *Two oracles $\pi_{i,j}^s$ and $\pi_{i',j'}^t$ are said to be partners if*

  – *$\pi_{i,j}^s$ and $\pi_{i',j'}^t$ have computed session keys and*
  – *$(\pi_{i,j}^s)_{sent} = (\pi_{i',j'}^t)_{received}$ and*
  – *$(\pi_{i',j'}^t)_{sent} = (\pi_{i,j}^s)_{received}$ and*
  – *$i = j'$ and $j = i'$ and*
  – *If $(\pi_{i,j}^s)_{role}$ = initiator, then $(\pi_{i',j'}^{s'})_{role}$ = responder, or vise versa.*

  Definition A.13 describes freshness, which depends on the respective notions of *matching sessions* and *origin-session* described above.

  Definition A.13 (CAFL Model Freshness). *A completed session s is said to be λ-CAFL-fresh if the following conditions are true:*

  – *The adversary $\mathcal{A}$ has not queried $\mathsf{SessionKey}(\pi_{i,j}^s)$,*
  – *The adversary $\mathcal{A}$ has not queried $\mathsf{SessionKey}(\pi_{j,i}^t)$, if $pi_{j,i}^t$) is its partner oracle,*

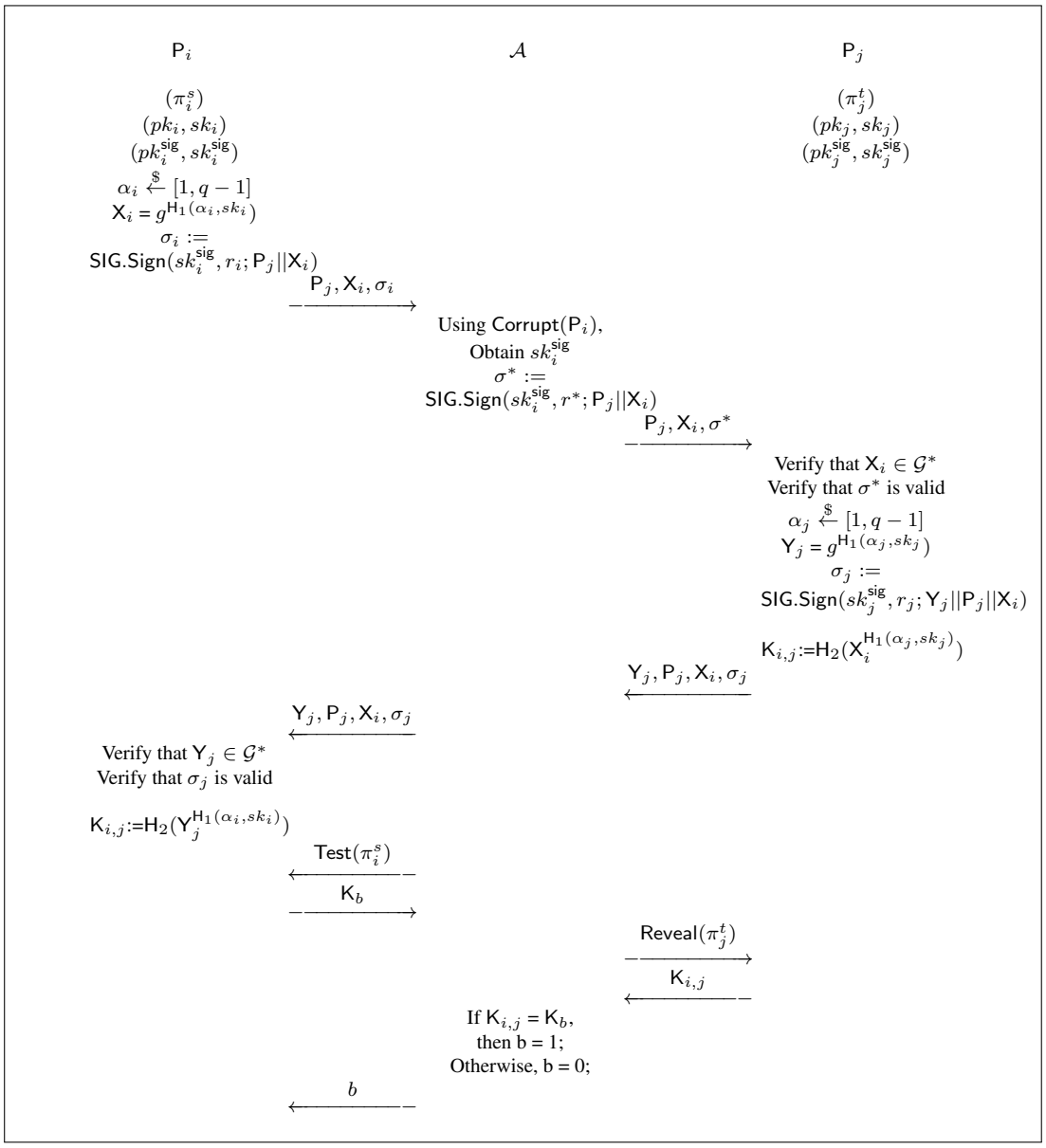


Fig. A.9. No-Match Attack against the $\mathsf{SIGDH}_{\mathsf{NAXOS}}$ Protocol

– *If the partner oracle $\pi_{j,i}^t$ exists, the adversary $\mathcal{A}$ does not query none of the following combinations:*

· $\mathsf{Corrupt}(\mathsf{P}_i)$ *and* $\mathsf{Corrupt}(\mathsf{P}_j)$,

· $\mathsf{Corrupt}(\mathsf{P}_i)$ *and* $\mathsf{EphemeralKey}(\pi_{i,j}^s)$,

· $\mathsf{Corrupt}(\mathsf{P}_j)$ *and* $\mathsf{EphemeralKey}(\pi_{j,i}^t)$,

· $\mathsf{EphemeralKey}(\pi_{i,j}^s)$ *and* $\mathsf{EphemeralKey}(\pi_{j,i}^t)$.

– *If the partner oracle $\pi^t_{j,i}$ does not exist, the adversary $\mathcal{A}$ does not query none of the following combinations:*
  · $\mathsf{Corrupt}(\mathsf{P}_j)$,
  · $\mathsf{Corrupt}(\mathsf{P}_i)$ *and* $\mathsf{EphemeralKey}(\pi^s_{i,j})$.
– *For each* $\mathsf{Send}(\pi^s_{i,j})$ *or* $\mathsf{Send}(\pi^t_{j,i})$ *query, the ouput of the leakage function is at most λ bits.*

SECURITY GAME. The security in the model is defined using the security experiment, played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The security experiment is similar to that as described in 3.4.3.

Definition A.14 (CAFL Model Security Definition). *The security definition is similar to that described in A.3.*

- **Protocol Description.**
  – Setup: Each party P_c has a valid long-term secret/public key pairs (pk_c, sk_c) for a (CCLA2) public-key scheme, where $c \in \{i, j\}$. Let KDF be a secure key derivation function.
  – Party P_i selects a random number $\alpha_i \xleftarrow{\$} \{0,1\}^\kappa$, computes a ciphertext Ciph_i by evaluating some probabilistic Enc function under the public key pk_j of its extended partner with internal random coins r_i over α_i, $\mathsf{Ciph}_i = \mathsf{Enc}(pk_j, r_i; \alpha_i)$. Finally, P_i sends (P_i, Ciph_i) to party P_j.
  – Party P_j decrypts Ciph_i and updates the secret state sk_j to sk'_j, $(sk'_j, \alpha_j) = \underline{\mathsf{Dec}}(sk_j, \mathsf{Ciph}_i)$ and $sk_j \leftarrow sk'_j$. Then, P_j selects a random number $\alpha_j \xleftarrow{\$} \{0,1\}^\kappa$, computes a ciphertext Ciph_j under the public key pk_i with internal random coins r_j over α_j, $\mathsf{Ciph}_j = \mathsf{Enc}(pk_i, r_j; \alpha_j)$. Finally, P_j sends (P_j, Ciph_j) to party P_i.
  – Party P_i behaves as similarly as P_j. It decrypts Ciph_j and updates the secret state sk_i to sk'_i, $(sk'_i, \alpha_j) = \underline{\mathsf{Dec}}(sk_i, \mathsf{Ciph}_j)$ and $sk_i \leftarrow sk'_i$.
  – P_i and P_j compute the same session key $\mathsf{K}_{i,j} := \mathsf{KDF}(\mathsf{P}_i, \mathsf{P}_j, \alpha_i, \alpha_j)$.
- **No-Match Attack against the Protocol π.**

  Figure A.11 shows our no-match attack against the protocol π. The adversary $\mathcal{A}$ performs the actions as follows:
  – The adversary $\mathcal{A}$ makes a $\mathsf{Corrupt}(\mathsf{P}_i)$ query and obtains the long-term private key of party P_i, sk_i.
  – According to the protocol specification, P_i selects a random number $\alpha_i \xleftarrow{\$} \{0,1\}^\kappa$, computes a ciphertext Ciph_i and sends (P_i, Ciph_i) to P_j.
  – Party P_j decrypts Ciph_i and updates the secret state sk_j to sk'_j, Then, it selects a random number $\alpha_j \xleftarrow{\$} \{0,1\}^\kappa$, computes a ciphertext Ciph_j and sends (P_j, Ciph_j) to party P_i.

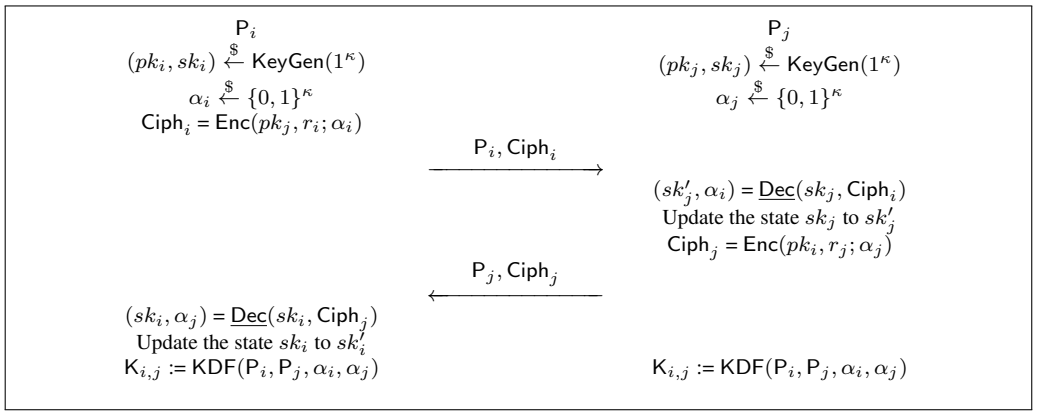


Fig. A.10. The description of the Protocol π

– Assume that the adversary $\mathcal{A}$ controls all communications between parties and deletes the Ciph_j generated by P_j. $\mathcal{A}$ decrypts Ciph_j and obtains the random number α_j. Then, it computes a fresh ciphertext Ciph^* with internal random coins r^* ($r^* \neq r_j$), $\mathsf{Ciph}^* := \mathsf{Enc}(pk_i, r^*; \alpha_j)$. Finally, $\mathcal{A}$ sends (P_j, Ciph^*) to P_i.
– Party P_i decrypts Ciph^* and updates the secret state sk_i to sk'_i. Finally, P_i and P_j computes the session key $\mathsf{K}_{i,j} := \mathsf{KDF}(\mathsf{P}_i, \mathsf{P}_j, \alpha_i, \alpha_j)$.
– The adversary $\mathcal{A}$ queries $\mathsf{Test}(\pi^s_{i,j})$ and gets K_b from oracle $\pi^s_{i,j}$. Then, $\mathcal{A}$ makes $\mathsf{SessionKey}(\pi^t_{j,i})$ and obtains the session key $\mathsf{K}_{i,j}$. Note that according to the definition of partnership of CAFL model $\pi^s_{i,j}$ and $\pi^t_{j,i}$ are not partnered, i.e., $(\pi^t_{j,i})_{sent} \neq (\pi^s_{i,j})_{received}$. Therefore, $\mathcal{A}$ is allowed to make $\mathsf{SessionKey}(\pi^t_{j,i})$ to oracle $\pi^t_{j,i}$. Oracle $\pi^s_{i,j}$ is fresh. Finally, if $\mathsf{K}_{i,j} = \mathsf{K}_b$, then $\mathcal{A}$ returns $b = 1$; Otherwise, returns b = 0.

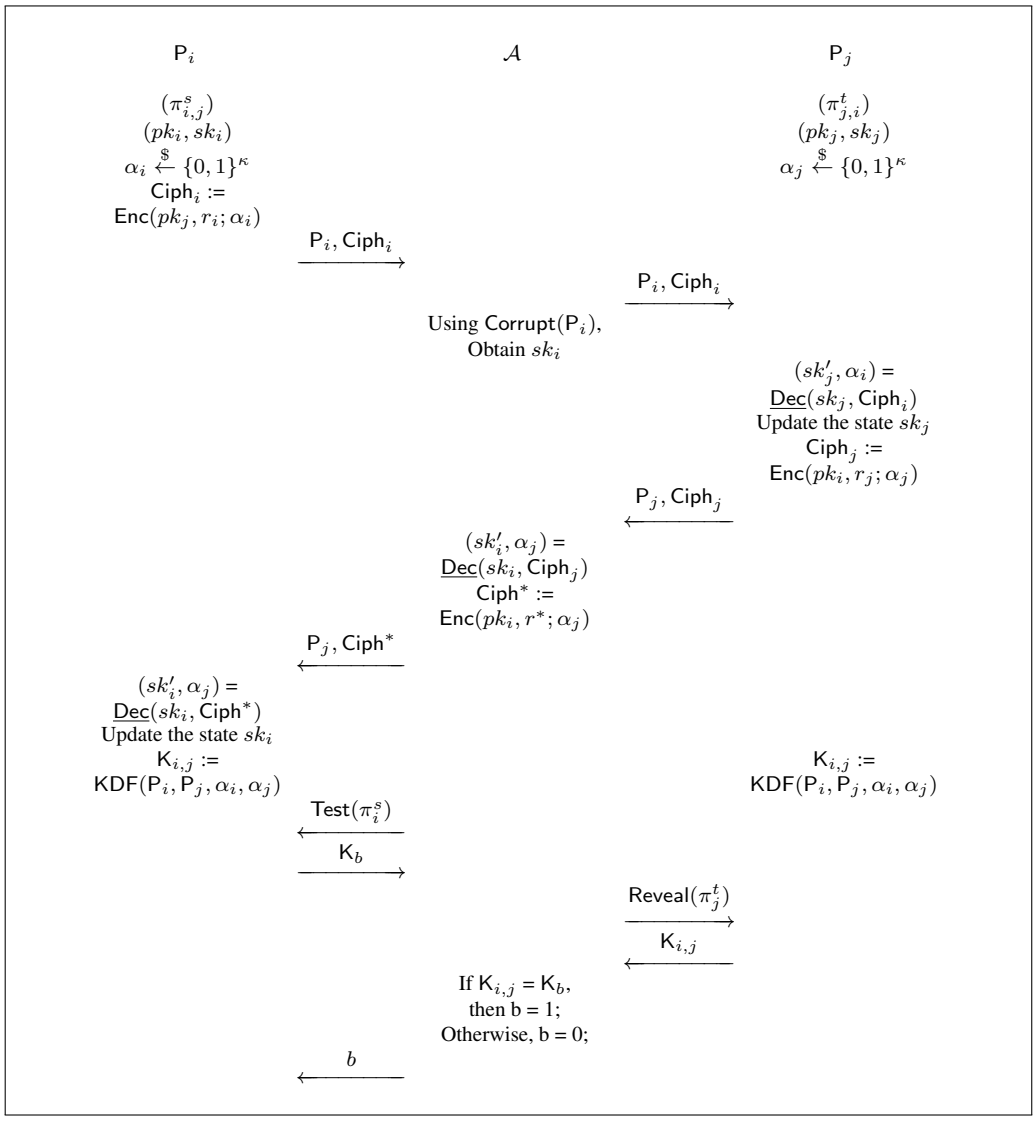


Fig. A.11. No-Match Attack against the Protocol π