Key Establishment Protocols and Timed-Release Encryption Schemes

Qiang Tang

Technical Report
RHUL–MA–2007–9
30 October 2007

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Key Establishment Protocols and Timed-Release Encryption Schemes

Qiang Tang

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2007

Declaration

These doctoral studies were conducted under the supervision of Prof. Chris J. Mitchell.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Qiang Tang
October, 2007

Acknowledgements

I would like to express my deepest gratitude to Royal Holloway, University of London, whose generous financial support has enabled me to pursue my studies. I also would like to express my deepest gratitude to École Normale Supérieure (Paris) for providing me an interesting research job at the end of my Ph.D study.

The past four years have been thus far the most exciting, challenging, and rewarding part of my life. I am very thankful that I have met many wonderful people who have helped me in many ways in my pursuit of a PhD at Royal Holloway.

My sincere thanks go to my supervisor, Chris J. Mitchell, whose dedication has been outstanding, and who has given generously of his time to guide and encourage me in my work over the last four years. His invaluable comments and unwavering support have been crucial in helping me to achieve my goals and develop as a researcher. I would also like to thank my advisor, Peter Wild, for his consistent encouragement.

I have also enjoyed working with and learning a lot from many people. A very partial list includes Liqun Chen, Zhaohui Cheng, Kim-Kwang Raymond Choo, Alexander W. Dent, Kenneth G. Paterson, Keith Martin, and Marko Schuba. Especially, I also would like to thank Alexander for his guidance in working on TRE-PC schemes, thank Liqun for her guidance and support, and thank Marko for his guidance during my interenship at Ericsson Research Labs (Germany). It has been great working with and learning from you.

I would also like to thank my friends and family for all their encouragement over the last four years. Especially, I want to thank my dear wife, Shenglan. She has been fantastic in every way, and I couldn't have done this without her love and support.

Abstract

This thesis is divided into two distinct parts. The first part of the thesis explores security issues in key establishment protocols, including both key distribution protocols and key agreement protocols, and in both the general and the password-based setting. The second part of the thesis explores security issues of Timed-Release encryption schemes, especially those with a Pre-Open capability.

In the first part, we initially present a formal description of key establishment protocols, and summarise the security properties that may be required of such a protocol. Secondly, we examine existing security models for key establishment protocols. We show that none of these security models fully capture the desired security properties. Thirdly, we examine some existing protocols and demonstrate certain vulnerabilities. Some of these vulnerabilities have not previously been detected because of the lack of a formal security analysis, while others have been missed because the adopted security models fail to address such security vulnerabilities. Fourthly, we describe a novel security model for general key establishment protocols, and we further adapt it for the password setting. Finally, we propose key establishment protocols which are proved secure in our novel security model.

In the second part we start by examining an existing security model for Timed-Release Encryption schemes with a Pre-open Capability (TRE-PC), and we demonstrate several limitations of this model. We then propose a new security model for such public-key encryption schemes, and establish relationships between the proposed security notions. We also propose a general construction for TRE-PC schemes and an instantiation of certain primitives.

Contents

List of Figures

List of Tables

Abbreviations

AES:	Advanced Encryption Standard
BDH:	Bilinear Diffie-Hellman
CAPTCHA:	Completely Automated Public Turing test to tell Computers and Humans Apart
CDH:	Computational Diffie-Hellman
DBDH:	Decisional Bilinear Diffie-Hellman
DDH:	Decisional Diffie-Hellman
DEM:	Data Encryption Mechanism
DH:	Diffie-Hellman
DPUKS:	Dishonest Partner Unknown key Share
GDH:	Gap Diffie-Hellman
HMAC:	Hash-based Message Authentication Code
IND-CCA:	Indistinguishability under Chosen Ciphertext Attack
IND-CPA:	Indistinguishability under Chosen Plaintext Attack
KE:	Knowledge of Exponent
KEM:	Key Encapsulation Mechanism
MAC:	Message Authentication Code
PKG:	Private Key Generator
RFC:	Request for Comments
RSA	Rivest-Shamir-Adleman
TLS:	Transport Layer Security
TRE:	Timed-Release Encryption
TRE-PC:	Timed-Release Encryption with Pre-open Capability
UC:	Universally Composable
WLAN:	Wireless Local Area Network

Chapter 1

Introduction

Contents

In this chapter, we provide the motivation for our research and describe the contributions of this thesis. We also present the overall structure of the thesis.

1.1 Motivation

This thesis explores security issues in key establishment protocols and timed-release public key encryption schemes. In this introductory section we briefly outline the main motivations behind the research.

Key establishment plays a fundamental role in enabling other security services, such as symmetric encryption and message authentication. The study of key establishment is well established, although the modern study of key establishment protocols can be traced back to the seminal work of Needham and Schroeder [197] in 1978. Since then, research into key establishment has grown rapidly, especially in the two-party setting, and a considerable number of protocols and security models have been proposed. Despite this rich research literature, many relevant issues have not been satisfactorily addressed. The first issue is that the security properties of key establishment protocols are still not well understood; as a result, many protocols and security models fail to cover the properties required by key establishment in practice. Our attempts at tackling this issue leads to the results presented in Chapter 3, where we provide a detailed analysis of the security properties of key establishment protocols. The second issue is that many security models proposed in the literature

fail to address certain practical security threats. Our contribution towards the resolution of this issue is given in Chapter 4, where we examine several representative security models, and analyse their capabilities for modelling security properties. In Chapter 6, we also propose a security model for security evaluation. The third issue is that the authors of many protocols claim that they are secure, although they are, in fact, insecure. Our work on this topic is described in Chapter 5, where we examine several protocols and point out their vulnerabilities. The last issue considered is how to build efficient and secure key establishment protocols. Our contributions on this issue are contained in Chapter 7, where we examine a protocol compiler and also propose a more efficient such scheme. In addition, we also propose two new compilers and several novel protocols.

Public key encryption is another fundamental part of cryptography, and plays a fundamental role in providing many security services. Timed-release public key encryption schemes, which are a special type of public key encryption scheme, have the capability to release the plaintext after a pre-defined time. We are particularly interested in those timed-release public key encryption schemes with pre-open capabilities, which enable the message sender to disclose the content in advance of the scheduled time if the sender wishes to. It is a challenging task to build a security model for this type of scheme, and we give some general constructions. Our contributions on this issue are given in Chapters 8 and 9, where we examine an existing security model, propose a new model and associated security notions, and establish the relationships between these notions. We also present a hybrid construction method for pre-release encryption schemes based on a TRE-PC KEM and a DEM, and we further propose an instantiation of this construction method.

1.2 Thesis Structure and Contributions

In Chapter 2, we give the preliminary material needed in the subsequent chapters of the thesis. We first review the relevant notions from complexity theory and mathematics, and then introduce the relevant cryptographic primitives and their security definitions. We also briefly review the security analysis methods contained in the literature, and, in particular, emphasise complexity-theoretic methods.

1.2 Thesis Structure and Contributions

The remainder of this thesis is divided into two distinct parts. In **Part I**, we present a study of key establishment protocols. This part of the thesis consists of Chapters 3–7.

In Chapter 3, we review the history of key establishment, describe some representative protocols, and then present a formal description for key establishment. Finally we describe the security properties which may be required of a key establishment protocol. In Chapter 4, we first review some representative security models for key establishment schemes, including the Bellare-Rogaway model [24, 29], which was the first proposed complexity-theoretical (indistinguishability-based) security model for key establishment, the Shoup model [217], which is simulatability-based, and the Canetti-Krawcyzk model [68], which combines the indistinguishability and simulatability techniques.

In Chapter 5, we present our security analyses of the enhanced Burmester-Desmedt protocol [61], a protocol proposed by Choi and Hwang and modified versions proposed in [80, 97, 98, 253], a scheme generated by the compiler of Katz and Yung [145], and the key agreement protocol contained in the Chinese WLAN implementation plan [1, 3, 14]. We then present some example protocols [116, 119, 217], which suffer from dishonest partner unknown key-share attacks. Finally, we present our analyses of the password-only authenticated key agreement protocol proposed by Jablon [130], the password-based authenticated key establishment protocol (referred to as the Laih-Ding-Huang protocol) proposed by Laih, Ding and Huang [162], the EKE-U and EKE-M protocols proposed by Byun and Lee [66], and the leakage-resilient authenticated key transport protocol (referred to as the RSA-AKE protocol) proposed by Shin, Kobara and Imai [216].

In Chapter 6, we propose a unified security model along the lines of the Bellare-Rogaway model, and also present some associated security definitions. In Chapter 7, we first present two compilers designed to transform any key establishment protocol secure against passive attackers into another scheme secure against any active attacker. We then present a variant of the Burmester-Desmedt protocol which is secure under the DBDH assumption, and two password-based key establishment protocols which are secure based on DDH and KE assumptions in the random oracle model.

In **Part II**, we present a comprehensive study of Timed-Release Encryption schemes. This part of the thesis consists of Chapters 8 and 9.

In Chapter 8, we investigate a security model for TRE-PC schemes proposed by Hwang, Yum, and Lee [124]. Firstly, we show that this model possesses a number of shortcomings and fails to model some potentially practical security vulnerabilities faced by TRE-PC schemes. Secondly, we propose a new security model for TRE-PC schemes which models security against four kinds of attacker and avoids the shortcomings of the Hwang-Yum-Lee model. We also establish the relationships amongst the security notions defined in the new model. In Chapter 9, we introduce the notion of a TRE-PC KEM, which is a special type of KEM (key encapsulation mechanism), and propose an implementation of this notion. We then show a way to construct a TRE-PC scheme using a TRE-PC KEM and a DEM (data encryption mechanism).

In Section 10, we conclude this thesis, and outline some directions for further research.

1.3 Publications

This thesis contains material that was previously published with Chen [228], with Choo [229], with Dent [90, 91], and with Mitchell [195, 231, 232, 233, 234, 235, 236, 237].

The contents of [76, 195, 227, 230, 232, 233, 234, 235, 236, 237] form the basis for Chapter 5, the content of [231] forms the basis for Chapter 7, and the content of [90, 91] forms the basis for Chapter 8 and 9.

Chapter 2

Preliminary Topics

Contents

In this chapter we first give general background material, and then review those cryptographic primitives relevant to this thesis. We also briefly introduce the security proof techniques used in this thesis, and, in particular, the complexity-theoretic methods.

2.1 General Background

2.1.1 Notation

We use "$\oplus$" to denote the exclusive-or operator and "$||$" to denote the string concatenation operator. If x is chosen uniformly at random from the set Y, then we

write $x \in_R Y$. The symbol "$\perp$" denotes an error message. In addition, we use ℓ to denote the security parameter.

2.1.2 Complexity Theory

Complexity theory provides mechanisms that can be used to classify computational problems in terms of the resources required to solve them, where the resources involved are usually time and, occasionally, storage space.

Definition 1 *An algorithm is a computational procedure which takes a variable input and terminates with some output. If an algorithm follows the same execution path each time it is executed with the same input, the algorithm is said to be deterministic. Otherwise, if an algorithm's execution path differs each time it is executed on the same input, the algorithm is said to be randomised*[1]*.*

The running time of an algorithm on a particular input is the number of steps or primitive operations executed before the algorithm terminates. The worst-case running time of an algorithm is an upper bound on the running time of an algorithm for any input. The running time for any input is usually expressed as a function of the input size, which in this thesis is usually the security parameter. The expected running time of an algorithm is the average running time of the algorithm over all inputs of a specific size. In security analyses, the notion of negligible probability is widely used, and is defined as follows.

Definition 2 *The function $P(\ell) : \mathbb{Z} \rightarrow \mathbb{R}$ is said to be negligible if, for every polynomial $f(\ell)$, there exists an integer N_f such that $P(\ell) \leq \frac{1}{f(\ell)}$ for all $\ell \geq N_f$.*

If $P(\ell)$ is negligible, then the probability $1 - P(\ell)$ is said to be overwhelming.

Because the exact running time of an algorithm is often difficult to compute, we usually rely on approximations of the running time. In particular, we often refer to

[1] As the randomness is always derived from a finite domain, the execution paths of a randomised algorithm will actually only differ with an overwhelming probability.

the order of the asymptotic upper bound of the running time of an algorithm using the big-O notation.

Definition 3 *If $f, g : \mathbb{R} \to \mathbb{R}$, then we write $g = O(f)$ if and only if there exists a constant c such that $g(x) \leq cf(x)$ for all $x > K$, for some constant K.*

Definition 4 *A polynomial-time algorithm is an algorithm whose worst-case running time is of the form $O(\ell^c)$, where ℓ is the input size and c is a constant.*

In general, we regard polynomial time algorithms as efficient, and other algorithms as inefficient. In a security analysis, an attacker (or adversary) is instantiated as an algorithm, and a two-stage attacker refers to an algorithm that is composed of two subalgorithms.

Definition 5 *If a problem cannot be solved by any polynomial-time algorithm (by existing techniques), then it is defined to be intractable or infeasible.*

A cryptographic scheme is defined as a set of algorithms used to provide some cryptographic service, a cryptographic protocol is defined as a distributed algorithm defined by a sequence of steps specifying the actions required by two or more entities to achieve a specific security objective, and a cryptographic primitive is defined as a basic tool used to provide certain security functionality. Note that the term protocol is sometimes applied to algorithms which demand interaction (or communication) between entities. Nonetheless, the terms *scheme* and *protocol* have been used interchangeably in many places (including this thesis).

2.1.3 Mathematical Background

We use the notation $\mathbb{G}$ to denote a group, which is a set with an associated binary operation which satisfies the group axioms [21]. By default, we write the group operation multiplicatively, and therefore a group is usually denoted as $(\mathbb{G}, *)$ or simply $\mathbb{G}$. The number of elements in $\mathbb{G}$, denoted $|\mathbb{G}|$, is called the order of $\mathbb{G}$. A

group $\mathbb{G}$ is cyclic if there is an element $g \in \mathbb{G}$ such that, for each $x \in \mathbb{G}$, there is an integer i satisfying $g^i = x$. In this case, g is called a generator of $\mathbb{G}$. A field, denoted by $(\mathbb{F}, +, *)$, is a set with two binary operations, namely addition $+$ and multiplication $*$, and the elements in $\mathbb{F}$ satisfy the field axioms [21].

The following assumptions have been widely used in constructing a variety of cryptographic primitives:

- The Discrete Logarithm (DL) assumption is as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs a cyclic group $\mathbb{G}$ of prime order q. On the input of $(\mathbb{G}, q, g)$ and a DL challenge g^a, where g is a generator of $\mathbb{G}$ and a is randomly chosen from $\mathbb{Z}_q$, a polynomial-time attacker can only compute a with a negligible probability.
- The Computational Diffie-Hellman (CDH) assumption is as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs a cyclic group $\mathbb{G}$ of prime order q. On the input of $(\mathbb{G}, q, g)$ and a CDH challenge (g^a, g^b), where g is a generator of $\mathbb{G}$ and a, b are randomly chosen from $\mathbb{Z}_q$, a polynomial-time attacker can only compute g^{ab} with a negligible probability.
- The Decisional Diffie-Hellman (DDH) assumption is as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs a cyclic group $\mathbb{G}$ of prime order q. On the input of $(\mathbb{G}, q, g)$ and a DDH challenge (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), where g is a generator of $\mathbb{G}$ and a, b, c are randomly chosen from $\mathbb{Z}_q$, a polynomial-time attacker can only distinguish between the two 3-tuples with a negligible advantage.

It is straightforward to verify that the DL assumption implies the CDH assumption, and the CDH assumption implies the DDH assumption; however, whether or not generally DDH implies CDH, and/or CDH implies DL, remains unclear. Other computational assumptions, such as the Generalised Diffie Hellman assumption [222], are also used in the literature, but we omit the descriptions here.

Besides these computational/decisional assumptions, the Knowledge of Exponent (KE) assumption is also used in a number of papers (e.g. [27, 87]). The KE assump-

tion is defined as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs a cyclic group $\mathbb{G}$ of prime order q. For any adversary $\mathcal{A}$ that takes $(\mathbb{G}, q, g)$ and a KE challenge g^a ($a \in_R \mathbb{Z}_q$) as input, and returns (C, Y) where $Y = C^a$; there exists an extractor $\mathcal{A}'$, which given the same input as $\mathcal{A}$ returns c such that $g^c = C$. This assumption is denoted as KEA1 by Bellare and Palacio [27].

We write $\mathbb{Z}_n$ for the group of integers $\{0, 1, \cdots, n-1\}$ under addition modulo n; if n is prime, we also write $\mathbb{Z}_n^*$ for the group of integers $\{1, 2, \cdots, n-1\}$ under multiplication modulo n. In the latter case, if $n - 1 = 2p$ where p is also prime (in which case n is referred to as a safe prime), then there is a unique cyclic subgroup $\mathbb{G}$ of $\mathbb{Z}_n^*$ of order p. Such a group $\mathbb{G}$ is widely used in the cryptographic literature.

Since the seminal work of Koblitz and Miller [150, 194], elliptic curve cryptography has become an important part of public key cryptography. An elliptic curve is the set of solutions (x, y) to an equation of the form $y^2 = x^3 + Ax + B$ over a field $\mathbb{F}$, together with an extra point O, known as the point at infinity. The set of points on an elliptic curve forms a group under a certain addition rule. In practice, we usually select $\mathbb{F}$ to be the integers modulo a prime q.

As a special case of the DL assumption, the Elliptic Curve Discrete Logarithm (ECDL) assumption is defined as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs a description of $(E, \mathbb{F})$, where E is an elliptic curve over the finite field $\mathbb{F}$. On the input of $(E, \mathbb{F})$ and a ECDL challenge (P, Q), where P is a point of $E(\mathbb{F})$ and Q is a random element of the additive group generated by P, a polynomial-time attacker can only compute a satisfying $Q = aP$, with a negligible probability.

Bilinear pairings have become an important tool in Cryptography. Let $\mathbb{G}_1, \mathbb{G}_2$ be additive groups of prime order q, and $\mathbb{G}_T$ be a multiplicative group of order q. Let P and Q be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear pairing is a polynomially computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with the following properties:

1. Bilinearity: for any integers a and b, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degeneracy: $\hat{e}(P, Q) \neq 1$.

In this thesis, we always assume that $\mathbb{G}_1 = \mathbb{G}_2$ and $P = Q$. The Bilinear Diffie Hellman (BDH) assumption is as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs an additive group $\mathbb{G}_1$ of prime order q, a multiplicative group $\mathbb{G}_T$ of the same order as $\mathbb{G}_1$, and a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. On the input of $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e}, P)$ and a BDH challenge (aP, bP, cP), where P is a generator of $\mathbb{G}_1$ and a, b, c are randomly chosen from $\mathbb{Z}_q$, a polynomial-time attacker can only compute $\hat{e}(P, P)^{abc}$ with a negligible probability.

The Decisional Bilinear Diffie-Hellman (DBDH) assumption is as follows: Given a security parameter ℓ, there exists a polynomial-time algorithm which takes ℓ as input and outputs an additive group $\mathbb{G}_1$ of prime order q, a multiplicative group $\mathbb{G}_T$ of the same order as $\mathbb{G}_1$, and a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. On the input of $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e})$ and a DBDH challenge $(aP, bP, cP, \hat{e}(P, P)^{abc})$ and $(aP, bP, cP, \hat{e}(P, P)^{r})$, where P is a generator of $\mathbb{G}$ and a, b, c, r are randomly chosen from $\mathbb{Z}_q$, a polynomial-time attacker can only distinguish between the two 4-tuples with a negligible advantage.

Abdalla *et al.* [7] showed that the DDH assumption is equivalent to the following parallel DDH assumption. The equivalence is also implicitly shown by Katz and Yung [145]. On the input of $(\mathbb{G}, q, g)$, a polynomial-time attacker can only distinguish between the following $2n$-tuples with a negligible advantage.

$$(g^{s_1}, g^{s_2}, \cdots, g^{s_n}, g^{s_1 s_2}, g^{s_2 s_3}, \cdots, g^{s_n s_1})$$

and

$$(g^{s_1}, g^{s_2}, \cdots, g^{s_n}, g^{r_1}, g^{r_2}, \cdots, g^{r_n}),$$

where $s, s_1, s_2, \cdots, s_n, r_1, r_2, \cdots, r_n \in_R \mathbb{Z}_q$.

Using the same techniques as those in [7, 145], we can prove that the DBDH assumption is equivalent to the following parallel version of the DBDH assumption. On the input of $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e})$, an attacker can only distinguish between the following $2n$-tuples with a negligible advantage.

$$(sP, s_1P, s_2P, \cdots, s_nP, \hat{e}(P, P)^{ss_1 s_2}, \hat{e}(P, P)^{ss_2 s_3}, \cdots, \hat{e}(P, P)^{ss_n s_1})$$

and

$$(sP, s_1P, s_2P, \cdots, s_nP, \hat{e}(P, P)^{r_1}, \hat{e}(P, P)^{r_2}, \cdots, \hat{e}(P, P)^{r_n}),$$

where $s, s_1, s_2, \cdots, s_n, r_1, r_2, \cdots, r_n \in_R \mathbb{Z}_q$.

2.2 Cryptographic Primitives

We now review the following cryptographic primitives and associated security notions: cryptographic hash functions and the random oracle model, block ciphers and the ideal cipher model, message authentication codes (MACs), public-key encryption, key encapsulation mechanisms (KEMs) and data encryption mechanisms (DEMs), and digital signatures.

2.2.1 Hash Functions and the Random Oracle Model

Cryptographic hash functions (hereafter simply referred to as hash functions) play a fundamental role in cryptography. In [190], hash functions are defined as follows:

Definition 6 *A hash function* H *is an algorithm which has, at minimum, the following two properties:*

1. *compression:* H *maps an input* x *of arbitrary finite bit-length to a fixed-length output* $\mathsf{H}(x)$.
2. *ease of computation: given* x, $\mathsf{H}(x)$ *is easy (or efficient) to compute.*

A hash function may also be required to possess some (or all) of the following security properties:

- preimage resistance: for any pre-specified output, it is computationally infeasible to find any input which hashes to that output, i.e., given any y for which no corresponding input is known, it is infeasible to find any preimage x such that $\mathsf{H}(x) = y$.

- 2nd preimage resistance: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, it is infeasible to find a 2nd-preimage x' ($x' \neq x$) such that $\mathsf{H}(x) = \mathsf{H}(x')$.
- collision resistance: it is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, i.e., such that $\mathsf{H}(x) = \mathsf{H}(x')$. (Note that here there is free choice of both inputs.)

Definition 7 *A one-way hash function (OWHF) is a hash function which possesses the following additional properties: preimage resistance and 2nd-preimage resistance.*

Definition 8 *A collision-resistant hash function (CRHF) is a hash function which possesses the following additional properties: preimage resistance, 2nd-preimage resistance, and collision resistance.*

Hash functions play an important role in the design of many cryptographic protocols, and therefore also have a major influence over the security of such schemes. Some of these protocols can be proved secure by assuming that the hash function is one-way or collision resistant. However, there are many protocols for which a security proof has not been successfully formulated using these assumptions. Bellare and Rogaway [30] propose the random oracle model for hash functions, which attempts to capture the concept of an ideal hash function. This model has been widely used in security analyses. In the random oracle model, a hash function $\mathsf{H} : S_1 \longrightarrow S_2$ is simulated as follows. A simulator maintains a list for all queried message and their corresponding hash values. When a message from S_1 is submitted to the hash oracle, the simulator first checks whether this message has already been queried. If yes, the simulator returns the corresponding hash value; otherwise returns a random element of S_2 and adds the new pair to the list. Proofs of security in the random oracle model are often far easier to construct than proofs in the standard model (i.e. without random oracles), and, in general, it seems that schemes proven secure in the standard model tend to be less efficient than schemes that employ hash functions and which can be proven secure in the random oracle model.

Recent work [22, 106, 199] has demonstrated that, for certain schemes, proofs of security in the random oracle model do not necessarily mean that the actual scheme

is secure when the random oracle is instantiated by a hash function. Despite the doubts that have been cast over the use of random oracles, proofs of security in the random oracle model are still widely accepted. In practice, it seems to be very difficult to design secure hash functions, and recent work has shown that many common hash functions are weaker than previously thought [243, 244, 245, 246].

2.2.2 Block Ciphers

Definition 9 *An n-bit block cipher is a function $\mathsf{Enc} : V_n \times \mathcal{K} \longrightarrow V_n$, such that for each m-bit $k \in \mathcal{K}$, $\mathsf{Enc}(\cdot, k)$ is an invertible mapping (the encryption function for k) from V_n to V_n, where $V_n = \{0,1\}^n$. The inverse mapping is the decryption function, denoted $\mathsf{Dec}(\cdot, k)$.*

It has become conventional to model block ciphers as a set of pseudo-random permutations (PRPs) [104]. By this we mean (informally) that an n-bit block cipher under a secret randomly-chosen key is computationally indistinguishable from a randomly-chosen permutation of the set of all n-bit vectors. However, in certain cases, we face a similar problem to that encountered when modelling hash functions; that is, we cannot construct a proof of security using only the PRP assumption in the standard model. To address this problem, the so called ideal cipher model was proposed and used in [42, 92, 101, 134, 148, 192, 247]. In the ideal cipher model, a block cipher $(\mathsf{Enc}, \mathsf{Dec}) : V_n \times \mathcal{K} \longrightarrow V_n$ is simulated as follows. For every $k \in \mathcal{K}$, a simulator maintains a list for all queried plaintexts and their corresponding ciphertexts. When a message from V_n is submitted to the encryption oracle Enc, the simulator first checks whether this message has already been a plaintext in the list. If yes, the simulator returns the corresponding ciphertext; otherwise returns a random element of V_n and adds the new pair to the list. When a message from V_n is submitted to the decryption oracle Dec, the simulator first checks whether this message has already been a ciphertext in the list. If yes, the simulator returns the corresponding plaintext; otherwise returns a random element of V_n and adds the new pair to the list. In practice, the ideal block cipher is normally instantiated with an existing block cipher scheme, such as AES [86].

2.2.3 Message Authentication Codes

A message authentication code (MAC) algorithm is a family of functions $\{\mathsf{h}_k\}$, parameterised by a secret key k, with the following properties:

1. Ease of computation: for a known function h_k, given a value k and an input x, $\mathsf{h}_k(x)$ is easy to compute. This result is called the MAC-value or MAC.
2. Compression: h_k maps an input x of arbitrary finite bit-length to an output $\mathsf{h}_k(x)$ of fixed bit-length.

Definition 10 *A MAC algorithm is said to be secure against existential forgery if, for any fixed key k (not known to the attacker), and given any number of MAC queries $h_k(x)$, where the values of x may be chosen by the attacker after observing the results of previous queries, it is computationally infeasible for a polynomial-time attacker to find a pair $(x^*, h_k(x^*))$ where x^* (which could be chosen by the attacker) was not in the set of MAC queries.*

Definition 11 *A MAC algorithm is said to be secure against selective forgery if, for any fixed key k (not known to the attacker), and given any number of MAC queries $h_k(x)$, where the values of x may be chosen by the attacker after observing the results of previous queries, it is computationally infeasible for a polynomial-time attacker to find a pair $(x^*, h_k(x^*))$ where x^* (previously given and not chosen by the attacker) was not in the set of MAC queries.*

HMAC, or keyed-hash message authentication code, is a type of MAC calculated using a cryptographic hash function in combination with a secret key. It was first proposed by Bellare, Canetti, and Krawczyk [23], and was later specified in an RFC [156] and ISO/IEC 9797-2 [128].

2.2.4 Public Key Encryption

Let $\mathcal{M}$ be the plaintext space and $\mathcal{C}$ be the ciphertext space. A public-key encryption scheme consists of a set of three algorithms, as follows:

1. A probabilistic polynomial-time key generation algorithm Gen, which takes as input a security parameter ℓ and outputs a public/private key pair (pk, sk).
2. A (possibly) probabilistic polynomial-time encryption algorithm Enc, which takes as input a message $m \in \mathcal{M}$ and a public key pk, and outputs a ciphertext $c = \mathsf{Enc}(m, pk)$.
3. A deterministic polynomial-time decryption algorithm Dec, which takes as input a ciphertext $c \in \mathcal{C}$ and a secret key sk, and outputs either a message $m \in \mathcal{M}$ or the error symbol $\perp$.

We next consider two security notions: indistinguishability against adaptive chosen ciphertext attack (IND-CCA2) and indistinguishability against chosen plaintext attack (IND-CPA). Other security notions and their inter-relationships can be found in [25].

Definition 12 *A public-key encryption scheme is said to be secure against an adaptive chosen ciphertext attack (IND-CCA2 secure), if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game*[2]*.*

1. The challenger generates a valid public/private key pair (pk, sk) by running $\mathsf{Gen}(\ell)$.
2. The attacker runs $\mathcal{A}_1$ on the input pk. It terminates by outputting two equal-length messages m_0 and m_1, as well as some state information *state*. During its execution, $\mathcal{A}_1$ can query the decryption oracle with any input.
3. The challenger picks a random bit $b \in \{0, 1\}$ and computes the challenge $c^* = \mathsf{Enc}(m_b, pk)$.

[2]Note that here and throughout we implicitly treat the advantage as a function of the security parameter ℓ

4. The attacker runs $\mathcal{A}_2$ on the input c^* and *state*. It terminates by outputting a guess b' for b. During its execution, $\mathcal{A}_2$ can query the decryption oracle with any input except for c^*.

The attacker wins the game if $b' = b$, and its advantage is defined to be $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 13 *A public-key encryption scheme is said to be secure against a chosen plaintext attack (IND-CPA secure) if it is IND-CCA2 secure against attackers that make no decryption query.*

2.2.5 KEMs and DEMs

A KEM consists of the following three algorithms:

- A probabilistic, polynomial-time key generation algorithm KEM.Gen that, given a security parameter ℓ as input, outputs a public/private key pair (pk, sk).
- A probabilistic, polynomial-time encapsulation algorithm KEM.Encap, which, on the input of a public key pk, outputs a pair (k, c), where k is a symmetric key and c is a ciphertext.
- A deterministic, polynomial-time decapsulation algorithm KEM.Decap, which, on the input of a ciphertext c and a private key sk, outputs either a symmetric key k or an error message $\perp$.

We assume that the possible keys k are drawn from a set of fixed length binary strings, $\{0,1\}^{\mathsf{KeyLen}(\ell)}$, where KeyLen is polynomial-time computable. The formal definitions for the security of a KEM against an adaptive chosen ciphertext attack and a passive attack are as follows.

Definition 14 *A* KEM *is defined to be secure against an adaptive chosen ciphertext attack (IND-CCA2 secure), if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{KEM.Gen}$ on the input ℓ to generate a public/private key pair (pk, sk).
2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input of pk. During its execution, $\mathcal{A}_1$ has access to a decapsulation oracle, which, on the input of c, returns $\mathsf{KEM.Decap}(c, sk)$. $\mathcal{A}_1$ terminates by outputting some state information $state$.
3. Challenge: The challenger generates a challenge encapsulated pair as follows:
   (a) The challenger generates an encapsulated pair $(k_0, c^*) = \mathsf{KEM.Encap}(pk)$.
   (b) The challenger randomly selects $k_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.
   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (k_b, c^*).
4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of $(k_b, c^*, state)$. During its execution, $\mathcal{A}_2$ has access to the decapsulation oracle. However, $\mathcal{A}_2$ may not make a query on the input c^*. $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

In this attack game, the attacker wins if $b' = b$, and the attacker's advantage is defined to be $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 15 *A* KEM *is defined to be secure against a passive attack (IND-CPA secure) if it is IND-CCA2 secure against attackers that make no decapsulation query.*

A DEM consists of the following two algorithms:

- A deterministic, polynomial-time encryption algorithm $\mathsf{DEM.Enc}$, which, on the input a message m and a key K, outputs a ciphertext C.
- A deterministic, polynomial-time decryption algorithm $\mathsf{DEM.Dec}$, which, on the input a ciphertext C and a key K, outputs a message m or an error message $\perp$.

We assume that the set of possible keys K is $\{0,1\}^{\mathsf{KeyLen}(\ell)}$. The formal definitions for the security of a DEM against an adaptive chosen ciphertext attack and a passive attack are as follows.

Definition 16 *A* DEM *is defined to be secure against an adaptive chosen ciphertext attack (IND-CCA2 secure), if a two-stage polynomial-time attacker* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *only has a negligible advantage in the following game.*

1. Phase 1: The attacker runs $\mathcal{A}_1$. At some point, $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1. In addition, $\mathcal{A}_1$ also outputs some state information *state*.
2. Challenge: The challenger randomly selects a bit $b \in \{0,1\}$ and a key $k \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, and returns $c^* = \mathsf{DEM.Enc}(m_b, k)$.
3. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of $(c^*, state)$. During its execution, $\mathcal{A}_2$ has access to the decryption oracle, which, on the input of c, returns $\mathsf{DEM.Dec}(c, k)$. However, $\mathcal{A}_2$ is not permitted to make a query on the input c^*. $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

In this attack game, the attacker wins if $b' = b$, and the attacker's advantage is defined to be $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 17 *A* DEM *is defined to be secure against a passive attack (IND-CPA secure) if it is IND-CCA2 secure against attackers that make no decryption query.*

2.2.6 Digital Signatures

Digital signature schemes provide a means by which an entity can bind its identity (or public key) to a piece of information (usually referred to as a message). A digital signature scheme is made up of the following algorithms [190]:

1. KeyGen: which takes a security parameter ℓ as input, and outputs a public (verification) key pk and a private (signing) key sk.
2. Sign: which takes as input a message m and a private key sk and produces a signature σ for the message m.

3. Verify: which takes as input a message m, a public key pk and a signature σ, and outputs either accept (denoted by 1) or reject (denoted by 0).

The existential unforgeability of a digital signature scheme is defined as follows:

Definition 18 *A digital signature scheme is existentially unforgeable under an adaptive chosen message attack if the probability of success of any polynomially bounded attacker in the following game is negligible.*

The attack game is carried out between an attacker $\mathcal{A}$ and the hypothetical challenger $\mathcal{C}$.

1. Initialisation: $\mathcal{C}$ runs $\mathsf{KeyGen}(\ell)$ to generate a public key pk and a private key sk.
2. Challenge: The attacker runs $\mathcal{A}$ on the input pk and terminates by outputting a pair m^*, σ^*. During its execution, $\mathcal{A}$ can query the Sign oracle with any input m ($m \neq m^*$).

The attacker wins the game if $\mathsf{Verify}(m^*, pk, \sigma^*) = 1$, and, the attacker's advantage is defined to be $\Pr[\mathsf{Verify}(m^*, pk, \sigma^*) = 1]$.

2.3 Security Analysis Methods

Heuristic analysis is the traditional method for security analysis [190]. This approach involves the use of a variety of informal arguments to try to establish that a successful protocol attack requires more resources (e.g., time or space) than the resources that might conceivably be possessed by an attacker. This approach may uncover protocol flaws, thereby establishing that a protocol is bad. However, claims of security may remain questionable, as subtle flaws in cryptographic protocols typically escape ad hoc analysis; in particular, unforeseen attacks remain a threat.

Information-theoretic analysis is another method adopted in the literature [190]. This approach uses mathematical proofs involving entropy relationships to prove that protocols are unconditionally secure by assuming that the attackers have unbounded computing resources. Whilst unconditional security is ultimately desirable, unfortunately, most practical schemes cannot achieve this kind of security. In fact, most existing cryptographic schemes are only computationally secure, which means that they cannot be regarded as secure if the attacker has access to unbounded resources.

As both the heuristic and information-theoretic approaches have shortcomings for practical security analysis, complexity-theoretic analysis has become the most widely used method in cryptographic research. This approach usually defines an appropriate model for the underlying target primitive, and attackers are assumed to be able to access polynomial resources. Then, if a successful protocol attack leads directly to the ability to solve a well-studied reference problem, the primitive can be regarded to be computationally secure in the underlying security model. Such analysis yields so-called provably secure protocols, although the security is conditional on the reference problem being truly (rather than presumably) difficult.

Using this approach, the computation time and probabilities are of major importance: a resource-unlimited attacker can always break provably secure schemes with probability one; or, in a shorter period of time, an attacker can guess the secret values by chance, and thus win the attack game with possibly negligible but non-zero probability. In this thesis, we consider only polynomial-time attackers. Bellare and Rogaway [32] and Shoup [218] provide some helpful methods for organising complexity-theoretic proofs.

Formal methods, such as BAN logic [64], CSP [121], and the Spi Calculus [5] have proven to be useful in finding flaws and redundancies in protocols, and some of these methods are automatable to varying degrees. For example, the SET protocol [219] has been analysed using a formal methods approach in [177, 188], and some of the web service protocols have also been formally analysed, see, e.g. [136]. However, the proofs provided are proofs within the specified formal system, and cannot be interpreted as absolute proofs of security. A one-sidedness remains: the absence of discovered flaws does not imply the absence of flaws. Some of these techniques

are also unwieldy, or applicable only to a subset of protocols or classes of attack. Many require (manually) converting a concrete protocol into a formal specification, a critical process which itself may be subject to subtle flaws. Recently, cryptographic researchers have tried to combine formal methods and complexity-theoretic analysis methods to automatically generate security proofs in computational security models (see e.g. [6, 17, 19, 48, 47, 115, 163, 164, 193]). Despite these recent advances, we are still a long way from automatic security proof generation.

In this thesis, we adopt the complexity-theoretic method in all the security proofs we give. An additional objective is to design cryptographic protocols which require the fewest cryptographic primitives, or the weakest assumptions.

Part I
Key Establishment Protocols

Chapter 3

Key Establishment Protocols

Contents

In this chapter we review the history of key establishment protocols, describe some representative protocols, and then present a formal description for key establishment protocols. Finally, we describe the security properties which may be required of a key establishment protocol.

3.1 Existing Key Establishment Protocols

In the literature, a number of terms have been used to describe a key establishment process, including key transport, key distribution, key assignment, key establishment, key exchange, key agreement, and key negotiation. Though these terms refer to slightly different (in terms of security properties) processes for sharing a key among a group of users, we use the term "key establishment" to denote a dynamic key generation process throughout this thesis.

3.1.1 Brief Review

The history of key establishment goes back a long way, although the modern study of key establishment protocols can be traced back to the seminal work of Needham and Schroeder [197]. Over the past 30 years, key establishment research, especially in the two-party setting, has been a very fruitful area in cryptography.

There are a number of key establishment protocols which are designed for particular environments. For example, those of Choi *et al.* [79], Yacobi and Beller [251], Yacobi [250], Park [205], Wong and Chan [248], Horn, Martin, and Mitchell [122], Horn and Preneel [123], Jakobsson and Pointcheval [132], and Boyd and Park [55] are designed for resource-limited devices. We next briefly summarise the literature of key establishment protocols for general purpose; detailed discussions can be found in [54, 190].

3.1.1.1 Protocols without complexity-theoretic proofs

Before the pioneering work by Bellare and Rogaway [29], who first analysed key establishment protocols using complexity-theoretic methods, the security of key establishment protocols was typically evaluated using only heuristic techniques. As a result, many protocols have been proposed which contain subtle vulnerabilities, only discovered after the schemes were published.

The summary below of existing protocols is divided into two main classes, depending on whether or not they use public key cryptography.

- The first category of protocols are those based purely on symmetric cryptography, i.e., long-term shared secret keys are used to guarantee the security of the session key. In this case, the long-term keys should possess a high entropy so that it is infeasible for an attacker to exhaustively search for them. If a password is used in this case, offline password guessing attacks are unavoidable.

  Many protocols have been proposed for use in a setting where a Trusted Third

Party (TTP) participates in the key establishment process, including by Boyd [52], Burrows, Abadi, and Needham [64], Chen, Gollmann, and Mitchell [74], Denning and Sacco [88], Gong *et al.* [108, 109, 110, 111], Bauer, Berson, and Feiertag [20], Needham and Schroeder [197], Janson and Tsudik [133], and Otway and Rees [204]. A number of protocols have also been proposed for use in a setting where no TTP is required in the key establishment process, including by Boyd [51], Burrows, Abadi, and Needham [65], Janson and Tsudik [133], and Satyanarayanan [215].

- Since the pioneering work of Diffie and Hellman [95] a range of key establishment protocols based on asymmetric techniques have been proposed, including by Agnew, Mullin, and Vanstone [11], Ateniese, Steiner, and Tsudik [16], Beller *et al.* [33, 34, 35, 36], Bird *et al.* [40], Burmester and Desmedt [61, 62], Carlsen [69], Denning and Sacco [88], Dierks and Allen [94], Harkins and Carrel [93], Hirose and Yoshida [118], Ingemarsson, Tang, and Wong [126], Johnston and Gemmell [137], Joux [138], Just and Vaudenay [141], Klein, Otten, and Beth [203], Lim and Lee [170], Mayer and Yung [187], Menezes, Qu, and Vanstone [189], Needham and Schroeder [197], Orman [202], Pereira and Quisquater [208], Pieprzyk and Li [209], Steer *et al.* [220], Steiner, Tsudik, and Waidner [222], Tatebayashi, Matsuzaki, and Newman Jr. [196], Tzeng [240, 241], Tzeng and Tzeng [242], Tseng [238], Yacobi and Beller [251], and Yacobi [250]. Some of these protocols (e.g. those given in [61, 126]) are only secure against passive attackers, because no long-term keys are employed. However, in other cases either long-term shared secret keys or other public-key techniques, such as signature schemes, are employed to give security against active attackers.

  The concept of password-based key establishment originates from the pioneering work of Lomas *et al.* [172]. Such protocols use both a secret password and public key techniques, such as Diffie-Hellman key exchange. Subsequently many password-based key establishment schemes have been proposed, including by Anderson and Lomas [15], Bakhtiari, Safavi-Naini, and Pieprzyk [18], Bellovin and Merritt [37, 38], Halevi and Krawczyk [114, 115], Jablon [130, 131], Kwon [158], Koyama and Ohta [152, 153], Koyama [151], Kwon and Song [159, 160, 161], Laih, Ding, and Huang [162], Lee *et al.* [167], Lin, Sun, and Hwang [171], Lomas *et al.* [173], Lucks [178], Nguyen and Vadhan [198], Patel [207], Steiner *et al.* [221], Tsudik and Herreweghen [239], Wu [249],

and Yen and Liu [252].

In these protocols, security is typically based on computational assumptions such as CDH, DDH, GDH, and BDH. Much less attention has been devoted to designing protocols using other intractable problems. In [149], Ko *et al.* proposed the first 2-party key agreement protocol based on a Diffie-Hellman-like conjugacy problem in a braid group. Lee [113] described the first authenticated group key agreement protocol based on the same problem. Lee's protocol has n rounds if a group of n users need to negotiate a session key. Unfortunately, Cheon and Jun [77] proposed a polynomial time algorithm for solving the Diffie-Hellman-like conjugacy problem, which means that both protocols are vulnerable to serious attacks.

Formal methods have also been used to analyse certain properties of key establishment protocols. Such work includes that of Abadi, Blanchet, and Fournet [4] and Lowe [174, 175, 176].

3.1.1.2 Protocols with complexity-theoretic proofs

The original Bellare-Rogaway model [29] was designed to enable the analysis of entity authentication and two-party key distribution in the shared secret key setting. Subsequently, a number of variants of this model have been proposed. Blake-Wilson, Johnson, and Menezes [43] extended the model to the public key setting. Bresson *et al.* [60] extended the model to the group setting. Kudla and Paterson [157] extended Bresson *et al.*'s model to model key compromise impersonation resilience. Chen, Cheng, and Smart [73] proposed another, similar, variant to model key compromise impersonation resilience. Bellare, Pointcheval, and Rogaway extended the Bellare-Rogaway model to cover password guessing attacks [28]. Bellare, Canetti, and Krawczyk [24] provide a modular approach for the construction of authenticated key establishment protocols. They also proposed the first simulatability-based security model for key establishment. Shoup [218] refined the simulatability-based security model and proposed a model which works under three different corruption assumptions. Later, Bellare, Canetti, and Krawczyk [24, 68] further extended the concept, and proposed another security model for two-party key agreement protocols in the universally composable security framework. Hitchcock, Boyd, and Nieto

[120] optimised the Bellare-Canetti-Krawczyk model. A number of papers have been devoted to discussing the validation of, and relationships between, these various security models [81, 82, 83]

Mayer and Yung [187] proposed a compiler which transforms any 2-party protocol into a centralised group protocol which, however, is not scalable. Later, Katz and Yung [145] proposed a compiler which transforms a group key exchange protocol secure (in the sense of guaranteeing key authentication and forward secrecy properties) against any passive attacker into an authenticated group key exchange protocol which is secure against both passive and active adversaries. The security of the Katz-Yung compiler is rigorously analysed in an adapted version of the security model of [60], and the protocols produced by this compiler are also more efficient and scalable than those produced by the method given in [187]. Katz and Shin [144] proposed a compiler which transforms a group key exchange protocol secure against passive attackers into an authenticated group key exchange protocol secure (in the sense of key authentication and key confirmation) against both passive and active attackers. In addition, Katz and Shin also claim that their compiler is secure in the Universally Composable framework [68].

We next summarise those key establishment protocols which have been analysed using complexity-theoretic methods.

- A number of provably secure shared secret key based key establishment schemes have been proposed, including by Bellare and Rogaway [29], Bellare, Canetti, and Krawczyk [24].
- Provably secure key establishment protocols based on public-key techniques include those due to Aiello *et al.* [13], Blake-Wilson *et al.* [44, 43, 46, 45], Bresson *et al.* [60], Bresson and Catalano [57], Bresson, Chevassut, and Pointcheval [58], Krawczyk [154, 155], Boyd [53], Chevassut *et al.* [78], Herranz and Villar [117], and Hitchcock, Boyd, and Nieto [119]. Identity-based key establishment protocols include those of Chen and Kudla [75], Girault [103], Günther [112], Jacobson, Scheidler, and Williams [139], Jeong, Katz, and Lee [135], Lauter and Mityagin [165], Okamoto [200], Okamoto and Tanaka [201], Pasini and Vaudenay [206], Mambo and Shizuya [184, 183], Saeednia [214, 213], and

Strangio [224, 225].

A large number of provably secure password-based key establishment protocols have been proposed, including those of Abdalla *et al.* [7, 8, 9, 10], Bellare *et al.* [28, 30], Boyko, MacKenzie, and Patel [56], Bresson, Chevassut, and Pointcheval [59], Byun and Lee [66], Canetti *et al.* [67], Dutta and Barua [99], Gennaro and Lindell [102], Goldreich and Lindell [105], Katz *et al.* [142], Katz, Ostrovsky, and Yung [143], Lee, Ha, and Kuo [166], Lee, Hwang, and Lee [168], MacKenzie, Patel, and Swaminathan [181], MacKenzie [179, 180], MacKenzie, Shrimpton, and Jakobsson [182], Nguyen and Vadhan [198], Raimondo and Gennaro [211], Strangio [226].

3.1.2 Example Protocols

We next briefly introduce the Diffie-Hellman key establishment protocol [95] and two extensions, namely the Burmester-Desmedt protocol [63] and the Joux protocol [138]. Note that these protocols are obviously not secure against active attackers; however, a large number of modified versions of these protocols have been proposed, with the goal of providing additional security properties (e.g. in [44, 43, 46, 45, 96]).

3.1.2.1 The Diffie-Hellman Protocol

The Diffie-Hellman key establishment protocol [95] is very simple. Let $\mathbb{G}$ be a multiplicative group with large prime order q, and g be a generator of $\mathbb{G}$. If Alice and Bob wish to establish a session key, then they exchange the Diffie-Hellman parameters g^a and g^b, where a is randomly chosen by Alice and b is randomly chosen by Bob. Then Alice and Bob can both compute the session key g^{ab}, whereas no passive interceptor can compute this key (given that the CDH assumption holds for $\mathbb{G}$).

3.1.2.2 The Burmester-Desmedt Protocol

Let $\mathbb{G}$ be a multiplicative group with large prime order q, and g be a generator of $\mathbb{G}$. Suppose that a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key. Each user U_i, for $1 \leq i \leq n$, performs the following steps. It should be noted that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $s_i \in_R \mathbb{Z}_q$, and broadcasts $Z_i = g^{s_i}$.
2. After receiving Z_{i-1} and Z_{i+1}, U_i computes and broadcasts X_i:
$$X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$$
3. After receiving every X_j $(1 \leq j \leq n,\ j \neq i)$, U_i computes the session key K_i as:
$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{ns_{i-1}s_i} \cdot (\frac{g^{s_i s_{i+1}}}{g^{s_{i-1}s_i}})^{n-1} \cdot (\frac{g^{s_{i+1}s_{i+2}}}{g^{s_i s_{i+1}}})^{n-2} \cdots \frac{g^{s_{i+n-2}s_{i+n-1}}}{g^{s_{i+n-3}s_{i+n-2}}} \\ &= g^{s_{i-1}s_i + s_i s_{i+1} + s_{i+1}s_{i+2} + \cdots + s_{i+n-2}s_{i+n-1}} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

It has been claimed that this protocol is secure against passive attackers given that the DDH assumption holds for $\mathbb{G}$ [63, 145]. A number of authenticated group key agreement schemes based on the Burmester-Desmedt protocol have been proposed, including those in [61, 80, 97, 98, 145].

We observe that the Burmester-Desmedt protocol possesses the following property. A malicious participant, U_j say, who can manipulate the communications in the network, is able to make any other participant, say U_i $(1 \leq i \leq n,\ i \neq j)$, compute the session key to be any value $K^* \in \mathbb{G}$ chosen by U_j.

To achieve this, in the second step, U_j intercepts the message X_{i-n+2} and prevents it from reaching U_i. U_j then waits until all the other messages have been received and computes the session key K in the normal way. U_j now sends $X'_{i+n-2} = X_{i+n-2} \cdot \frac{K^*}{K}$ to U_i, pretending that it comes from U_{i+n-2}.

Lemma 1 *As a result of the above attack, U_i will compute the session key as K^*.*

Proof. This is immediate, since U_i will compute the session key as $K \cdot \frac{X'_{i+n-2}}{X_{i+n-2}} = K^*$, by definition of X'_{i+n-2}. □

3.1.2.3 The Joux Protocol

Joux [138] introduced a one-round three-party key establishment protocol in which the session key can be generated after every party has broadcast a single message. The protocol makes use of bilinear pairings.

Suppose that U_i $(1 \leq i \leq 3)$ share the common values $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e})$ and P which is a generator of the group $\mathbb{G}_1$. During the protocol execution, U_i, for $1 \leq i \leq 3$, randomly chooses $x_i \in \mathbb{Z}_q$ and broadcasts $x_i P$. On receiving the messages from the other two users, U_i computes the session key as $K = \hat{e}(x_{i-1}P, x_{i+1}P)^{x_i}$. Note that the the indices of messages are taken modulo 3.

3.2 Formal Description of Key Establishment Protocols

We next consider the main components of a key establishment protocol, namely the principals exchanging the messages and the component sub-protocols. We also examine the role of session identifiers.

3.2.1 Principals in Key Establishment Protocols

In general, two types of principal may be involved in a key establishment protocol:

1. a Trusted Third Party, which may exist for many different purposes, such as certifying users' public keys and distributing long-term private keys;

2. a user, which possesses a unique identifier and any other information required by the protocol specification.

Note that any public key (for a signature scheme or a public-key encryption scheme) that is used by any of the principals must be known to be genuine by the principals concerned. This is typically achieved by arranging for the key to be be certified by a TTP, and the resulting certificate should be verified by a principal before the public key is used. In the rest of this thesis, we omit an explicit description of this certificate distribution and validation process.

We assume that an attacker is always present, i.e. a hypothetical entity trying to attack the key establishment protocol. We classify attackers into two types, depending on their ability to access the public/private information of the principals.

1. An outsider, only has access to the messages sent via public communication channels.
2. An insider, in addition to the information in the public communication channels, also has access to some privileged information, possibly including long-term private keys of users, long-term private keys of the TTP, and ephemeral secret information generated during protocol executions. The insider can be further sub-divided into the following attacker types:
   (a) An insider which has access to users' long-term private keys.
   (b) An insider which has access to users' ephemeral secret information generated during protocol executions.
   (c) An insider which has access to long-term private keys of the TTP, and possibly also ephemeral secrets of the TTP.

   It is worth noting that, in practical applications, an insider attacker may have access to more than one type of privileged information.

In practice, an inside attacker is likely to be either a group of malicious users or an entity which has compromised one or more users' secrets. In particular, if users U_i

($1 \leq i \leq n$) run the protocol to generate a session key, then some of them may be malicious, as we discuss in Section 5.4. In this case, we define the malicious users to be *dishonest partners*, and, in the security formalisation, this type of attacker is assumed to have access to these users' long-term private keys and ephemeral private keys. Without loss of generality, the term "inside attacker" henceforth excludes dishonest partners.

Depending on whether or not it can control the messages sent over the public communication channels, the attacker can be either passive or active, where a passive attacker can only eavesdrop on the messages sent over the public channels, while an active attacker can freely manipulate the messages. Thus, an attacker could be either an outsider or an insider, and either passive or active. If we assume that the protocol messages are transported over public communication channels, then, in general, we make the stronger assumption, i.e. that the attacker is active.

3.2.2 The Role of Session Identifiers

We first show that a potential security vulnerability exists in any protocol if no session identifier is used, and then examine a practical example of a key establishment protocol and show how a session identifier is generated and used in this protocol.

3.2.2.1 A Potential Security Vulnerability

Let $\mathcal{P}$ be a two-party key establishment protocol, and U_i ($1 \leq i \leq 2$) be two user devices which are both running two applications App_j ($1 \leq j \leq 2$). Suppose that both applications rely on $\mathcal{P}$ to generate a session key and protect their communications.

If $\mathcal{P}$ does not provide a unique session identifier identifying the session key in different applications, then the following generic potential security vulnerability exists, as shown in Figure 3.1.

The attacker can swap the messages sent by user U_1 between App_1 and App_2. In the end, the session key of U_1 in App_1 is identical to the session key of U_2 in App_2,

Session 1			Session 2		
App_1		App_1	App_2		App_2
m_1	$\overset{m'_1}{\longrightarrow}$		m'_1	$\overset{m_1}{\longrightarrow}$	
	$\overset{m_2}{\longleftarrow}$	m_2		$\overset{m'_2}{\longleftarrow}$	m'_2
	$\cdots$		$\cdots$		
m_{n-1}	$\overset{m'_{n-1}}{\longrightarrow}$		m'_{n-1}	$\overset{m_{n-1}}{\longrightarrow}$	
	$\overset{m_n}{\longleftarrow}$	m_n		$\overset{m'_n}{\longleftarrow}$	m'_n

Figure 3.1: Potential attack in the absence of a session identifier

while the session key of U_1 in App_2 is identical to the session key of U_2 in App_1.

This example shows that a potential protocol vulnerability (or ambiguity between concurrent protocol executions) will exist if the session key is not generated and authenticated with respect to a session identifier. In fact, if we allow concurrent executions, the session key should only be used with reference to a session identifier.

3.2.2.2 The IKE Protocol

IKE [93] is a well-known practical example of a key establishment protocol. Not surprisingly, it makes use of a session identifier in the generation and application of the session key.

In the IKE protocol [93], each session key is associated with a unique Security Parameter Index (SPI), which is an arbitrary 32-bit value. Together with the destination IP address and security protocol (AH [146] or ESP [147]), the SPI uniquely identifies the Security Association (SA) between two endpoints. Since the application protocol (AH or ESP) uses the session key by referring to the SA, the above security vulnerability is avoided.

3.2.3 Specifications of Key Establishment Protocols

Typically, a key establishment protocol consists of the following three sub-protocols: a TTP initialisation sub-protocol, a user initialisation sub-protocol, and a key establishment sub-protocol. Since the TTP initialisation sub-protocol and the user initialisation sub-protocol are usually executed once before any executions of the key establishment sub-protocol, we always assume that they are executed in the absence of any attacker.

1. TTP initialisation sub-protocol: Run by the TTP, this protocol generates the long-term public/private keys for the TTP and distributes public system parameters to all users.
2. User initialisation sub-protocol: Run by a user, this protocol generates the user's public/private system parameters, and possibly distributes parameters to other parties.
3. Key establishment sub-protocol: Run by a user, this protocol enables the user to establish a session key with a group of other users. At the end of the protocol execution, the user obtains a session identifier and the associated session key.

In the Canetti-Krawczyk model [68], a session identifier is always required for executing key establishment protocols, but there is no such requirement in the Bellare-Rogaway model and its variants (e.g. [29, 43]). Nonetheless, when evaluating security in those security models, the concept of session identifier is used to define partnerships. However, as long as the key establishment protocol is used in the same way as IKE [93], the security vulnerability described above can be avoided.

3.3 Security Properties for Key Establishment Protocols

In this section we consider the properties that may be required of a key establishment protocol. We later use this list to motivate our formal definitions of security for a key establishment protocol.

3.3 Security Properties for Key Establishment Protocols

Menezes, Oorschot, and Vanstone [190] describe a number of possible security properties for two-party key establishment protocols. Boyd and Mathuria [54] also provide a summary of security properties for key establishment protocols. In this section, building upon this previous work we enumerate the security properties which may possibly be required of a key establishment protocol.

As we have shown in the previous section, the concept of a session identifier plays a crucial role in the application and security formulation of key establishment protocols. Where relevant, in the security property descriptions below we refer to session identifiers.

1. *Key authentication* is defined to be the property that a user who has completed a successful protocol execution in a certain session, can be assured that no inside attacker can have access to the session key[1].

   Key authentication under this definition is also referred to as implicit key authentication in [190].

2. *Forward secrecy* is defined to be the property that the compromise of the long-term private keys of some (but not all) intended users in a successfully ended session does not compromise the session key established in that session. *Perfect forward secrecy* is defined to be the property whereby the compromise of the long-term private keys of all intended users in an successfully ended session does not compromise the session key in this session.

   The concept of perfect forward secrecy was originally proposed by Günther [112]. Forward secrecy under our definition is equivalent to the definition of partial forward secrecy given by Boyd and Mathuria [54], while perfect forward secrecy under our definition is equivalent to their definition of forward secrecy.

3. *Backward secrecy* is defined to be the property whereby the compromise of a user's long-term private key does not compromise this user's session key in any subsequent session.

   A protocol with this property guarantees that a user which uses a compromised long-term private key can still be assured that its future session keys

[1] It is worth noting that key authentication does not guarantee that other intended users actually possess the session key. In fact, it does not even guarantee that any intended user actually knows there is a key agreement process going on.

will remain secret to the attacker. This property is very desirable in the environments where it is hard for a user to detect that its long-term private keys have been compromised.

4. *Unknown key-share resilience* is defined to be the property that an honest user A never ends up believing it shares a key with user B, although A actually shares the session key with another honest user C which is different from B, and C thinks it shares the key with another user D, which may or may not be the same as A. If B and D are malicious users[2], then we call this property dishonest partner unknown key-share resilience.

   The notion of unknown key-share attacks were first discussed by Diffie, van Oorschot and Wiener [96]. In [46], an unknown key-share attack on an authenticated key agreement protocol is defined to be an attack whereby an honest user A ends up believing it shares a key with another honest user B and, although this is in fact the case, B mistakenly believes the key is instead shared with another user E. It is easy to see that if a protocol suffers from a UKS attack under this definition, then it will also suffer from a UKS attack under our definition.

5. *Key randomness* is defined to be the property that, in any successful session for a user, the session key is uniformly distributed amongst the set of all possible session keys.

6. *Key control* is defined to be the property whereby, in any session, no strict subset of the set of the intended users is able to force the session key to be equal to a pre-determined value.

   This definition implies that, in any session, only all the intended users acting together are able to completely control the value of the session key.

7. *Entity authentication* is defined to be the property that a user who has completed a successful protocol execution in a certain session, can be assured that the other intended users are actually involved in the same session.

8. *Known-key resilience* is defined to be the property that the compromise of the session key in one session does not impose any danger to key establishment in any other sessions.

[2]In this case, D is a user different from A.

The concept of known-key attacks was discovered by Denning and Sacco [88]; Menezes, van Oorschot and Vanstone [178] use the term known key-attack for an attack in which the compromise of past session keys allows an attacker either to compromise future session keys, or to impersonate a user in the subsequent protocol executions. In fact, the compromise of the session key in one session might affect a range of possible security properties in other sessions, not just key authentication and entity authentication, which is why we propose the above definition. In our security formulations we take this definition into account when evaluating other properties.

9. *Key confirmation* is defined to be the property that, in any session, a user who has completed a successful protocol execution in a certain session, can confirm that other intended users have participated in the same session and that they have computed the same session key.

   In [190], key confirmation is defined to be the property whereby one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key.

10. *Key-compromise impersonation resilience* is defined to be the property that, in any session, the compromise of a user's long-term private key does not enable the attacker to impersonate any other non-compromised intended user to the user whose key has been comprised.

    A key establishment protocol with this property guarantees that a user, whose private key has been compromised can still be assured that the intended partners are participating in a session (although the partners will have no such guarantee). This property is very desirable in the environments where it is hard for a user to detect that its long-term private keys have been compromised. The term "key-compromise impersonation resilience" as defined by some other authors, see, for example, in [157], also covers the backward secrecy property.

11. *Denial of Service resilience* is defined to be the property whereby an attacker is unable to effectively make a resource unavailable to the intended users.

    In computer security, a denial of Service (DoS) attack is an attempt to make computing resources, especially computational resources, unavailable to its intended users. In the design of key establishment protocols, we regard a

protocol to be prone to DoS attack if the computational load on one or more of the legitimate users could be very high. Matsuura and Imai [185] propose an enhanced version of the IKE protocol to make it robust against DoS attacks. Note that DoS attacks are of particular concern in the client-server setting, where the server is often an attack target.

Note that any key establishment protocol cannot completely avoid DoS attacks, but reducing the risk of DoS attacks is always an important design criterion.

12. *Information leakage resilience* is defined to be the property whereby protocol executions do not leak unreasonable amounts of information.

    This is mainly relevant in the case where a low-entropy password is used. In this case, we need to consider both online password guessing attacks and offline dictionary attacks, where, in an online password guessing attack, an attacker tries a guessed password by manipulating a protocol message without letting the users detect any failure, while in an offline dictionary attack an attacker tries to exhaustively search for the password by possibly manipulating the protocol messages.

    In fact, there are other situations (especially when key establishment protocols are used together with other protocols), where information leakage could occur. In Figure 3.2 we present a toy example of a key establishment protocol $\mathcal{P}'$, where r is a random string, (pk_2, sk_2) is a sign/verify key pair of U_2, and $\mathcal{P}$ is a two-party key agreement protocol.

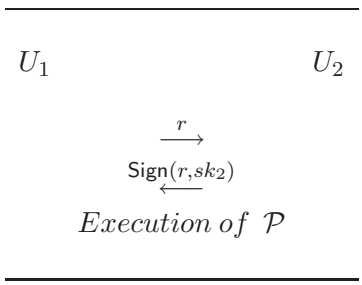

Figure 3.2: The Toy Protocol $\mathcal{P}'$

It is straightforward to verify that, if (pk_2, sk_2) is used by U_2 in other protocols, it is very likely that this key establishment protocol will lead to those protocols being insecure, because U_1 can make U_2 sign any message it selects.

3.4 Conclusions

In this chapter we have presented a brief review of the state of the art in key establishment. In particular we have classified the possible types of attacker, and emphasised the role of session identifiers. Based on this analysis we have described the properties that may be required of a key establishment protocol.

Chapter 4

Complexity-theoretic Security Models for Key Establishment

Contents

In this chapter we review three representative security models for key establishment protocols, including the Bellare-Rogaway model, the Shoup model, and the Canetti-Krawczyk model. We also compare their capabilities with respect to modelling the security properties described in section 3.3.

4.1 Introduction

Since Bellare and Rogaway proposed the first complexity-theoretic model for entity authentication and key establishment protocols, which applies to schemes using symmetric cryptography in the two-party setting [29], many papers have been published proposing extensions to this model to cover a wide range of security properties in a more general setting (see e.g. [24, 28, 31, 44, 60, 68, 75, 157, 225]). These security models use an indistinguishability approach to evaluate session key security, i.e., a key establishment protocol is said to achieve session key security if it is infeasible for any attacker to distinguish between the session key and a randomly chosen string.

In contrast to the indistinguishability approach, another approach based on simulatability has also been widely discussed in the literature (see e.g. [28, 68, 217])[1]. In this approach, ideal functionality for a key establishment protocol is first defined, where the attacker's capabilities are highly restricted (compared to the real-world). A key establishment protocol is said to achieve session key security if it is infeasible to distinguish between an ideal-world execution of the protocol and a real-world execution, where the attacker's capabilities model the threats to key establishment protocols in practice.

The rest of this chapter is organised as follows. In section 4.2 we review the Bellare-Rogaway model and certain extensions of this model. In section 4.3 we review the Shoup model. In section 4.4 we review the Canetti-Krawcyzk model. In the last section we compare the capabilities of these three models for modelling the security properties described in section 3.3, and then conclude this chapter.

4.2 The Bellare-Rogaway model

The Bellare-Rogaway model [29] was designed to model the security properties of entity authentication and key distribution protocols in the two-party setting, where the protocols use symmetric cryptography. Blake-Wilson and Menezes [44] adapted the Bellare-Rogaway model to two-party key establishment protocols which employ

[1] This approach was originally applied to public key encryption schemes [107], but has since been applied to other cryptographic primitives, including key establishment protocols.

asymmetric cryptography. The following description of the Bellare-Rogaway model combines these two approaches.

4.2.1 Preliminary Definitions

Given any key establishment protocol, without loss of generality let U_i $(1 \leq i \leq N)$ be the set of all possible users which may run the protocol, where N is a sufficiently large integer. In any protocol execution, U_i could be either an initiator or a responder, determined by whether or not U_i initiates the protocol execution.

In the Bellare-Rogaway model, each protocol instance of a user is defined to be an oracle, i.e. a Turing Machine that processes protocol messages. An oracle $\Pi_{i,j}^s$ denotes the s-th instance of user U_i involved with a partner party U_j. If the protocol instance, represented by an oracle, ends successfully, then oracle is said to accept. If the oracle $\Pi_{i,j}^s$ accepts, it outputs the following information $(pid_{\Pi_{i,j}^s}, sid_{\Pi_{i,j}^s}, sk_{\Pi_{i,j}^s})$, where $pid_{\Pi_{i,j}^s} = U_j$ is the identifier of the user with which it assumes it is communicating, $sid_{\Pi_{i,j}^s}$ is the session identifier, and $sk_{\Pi_{i,j}^s}$ is the established session key. Note that $pid_{\Pi_{i,j}^s}, sid_{\Pi_{i,j}^s}$ are regarded as public information. Once created, an oracle $\Pi_{i,j}^s$ may be in one of three states:

- active, when the oracle possesses an ephemeral state and is ready for receiving messages;
- accepted, when the oracle has ended successfully and outputs $(pid_{\Pi_{i,j}^s}, sid_{\Pi_{i,j}^s}, sk_{\Pi_{i,j}^s})$;
- aborted, when the oracle has failed to generate a session key and only outputs a failed state.

In the Bellare-Rogaway model, it is assumed that an attacker is responsible for initiating protocol executions and delivering messages sent between users. In other words, the attacker has the privilege of initiating oracles and delivering messages. Security properties of a key establishment protocol are modelled by attack games played between a hypothetical challenger and an attacker, where the challenger simulates the protocol execution, and the attacker intervenes in the protocol execution

by means of certain oracle queries (answered by the challenger). The following types of oracle queries are defined:

1. create, which on the input of $(ID_i, ID_j, role)$, where $role$ is either "initiator" or "responder", creates an active oracle $\Pi^s_{i,j}$ to start a session with U_j, where s is the index of this oracle for U_i. Usually, it is required that $i \neq j$, i.e., a user will not run a session with itself.

2. send, which on the input of an active oracle $\Pi^s_{i,j}$ and message m, delivers m to $\Pi^s_{i,j}$ and responds with either a message m' or an indication of whether or not $\Pi^s_{i,j}$ accepts or rejects the session.

3. reveal, which, on the input of an accepted oracle $\Pi^s_{i,j}$, returns the session key possessed by this oracle.

4. corrupt, which, on the input of any user identifier ID_i, returns U_i's long-term private key.

5. test, which, on the input of a fresh oracle $\Pi^s_{i,j}$ (see the definition below), returns a string which is computed as follows: choose a random bit b from the set $\{0, 1\}$, return the session key if $b = 1$, otherwise return a random string from the session key space.

If $\Pi^s_{i,j}$ accepts, its session identifier $sid_{\Pi^s_{i,j}}$ is defined to be the concatenation of the messages exchanged during the session. The concept of *matching conversations* plays an important role in the security formulation.

Definition 19 *Two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ are defined to have matching conversions if the following requirements are satisfied:*

1. *$pid_{\Pi^s_{i,j}} = U_j$ and $pid_{\Pi^t_{j,i}} = U_i$;*

2. *Each message sent by the initiator, $\Pi^s_{i,j}$ say, is received by the responder, $\Pi^t_{j,i}$ say, and each response generated by $\Pi^t_{j,i}$ is received by $\Pi^s_{i,j}$. Each message received by $\Pi^t_{j,i}$ is from $\Pi^s_{i,j}$, and each message sent by $\Pi^t_{j,i}$ is received by $\Pi^s_{i,j}$.*

Intuitively, having matching conversations implies that $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have some kind of partnership, since protocol messages between them have been faithfully delivered by the attacker. This concept of partnership is expressed in the notion of *partner oracles*.

Definition 20 *Two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ are said to be partner oracles if they have matching conversations.*

In order to facilitate the security formulation, the freshness of an oracle is defined as follows.

Definition 21 *An oracle $\Pi^s_{i,j}$ is fresh if it has accepted and satisfies the following requirements:*

1. *$\Pi^s_{i,j}$ has not been issued any* reveal *query;*
2. *If a partner oracle $\Pi^t_{j,i}$ exists, $\Pi^t_{j,i}$ has not been issued any* reveal *query;*
3. *Neither U_i nor U_j has been corrupted before $\Pi^s_{i,j}$ accepts.*

Note that the third requirement is not specified in [29] because, if only symmetric cryptography is employed, the attacker will obtain all of U_i's session keys whenever it compromises U_i's long-term secret keys. More explanations are given after the definition of the session key attack game below.

4.2.2 Security Definitions

The attack game for session key security is carried out between a two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger $\mathcal{C}$ as follows:

1. Setup: The challenger $\mathcal{C}$ generates both the public system parameters $param_1$ and private system parameters $param_2$.

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $param_1$. $\mathcal{A}_1$ can make the following types of queries: create, send, reveal, and corrupt. $\mathcal{A}_1$ terminates by issuing a test query with a fresh oracle $\Pi^s_{i,j}$, and outputting some state information *state*.

3. Challenge: The challenger $\mathcal{C}$ returns the output of $\mathsf{test}(\Pi^s_{i,j})$.

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input *state* and the output of the challenger. $\mathcal{A}_2$ can make the same types of queries as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not permitted to issue a reveal query to $\Pi^s_{i,j}$ and its partner oracle. $\mathcal{A}_2$ terminates by outputting a guess bit b'.

At the end of this game, the attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

The attack game for entity authentication, in which a polynomial-time attacker $\mathcal{A}$ tries to impersonate another user U_j to U_i, is carried out between $\mathcal{A}$ and a hypothetical challenger $\mathcal{C}$ as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and private system parameters $param_2$.

2. Challenge: The attacker runs $\mathcal{A}$ with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting an accepted oracle $\Pi^s_{i,j}$. During its execution, $\mathcal{A}$ can make any number of create and send queries.

At the end of this game, the attacker wins if $\Pi^s_{i,j}$ has no partner oracle.

Given these preliminary definitions, secure key establishment and entity authentication protocols are defined as follows.

Definition 22 *A key establishment protocol is said to be a secure authenticated key establishment protocol, if it satisfies the following requirements:*

1. *In the presence of only a passive attacker, which faithfully delivers protocol messages,* $\Pi^s_{i,j}$ *will always accept and have a partner oracle* $\Pi^t_{j,i}$*. Both oracles generate the same session key which is uniformly distributed over the session key space.*
2. *A polynomial-time attacker only has a negligible advantage in the attack game for session key security.*

A secure authenticated key establishment protocol is also said to be AK secure.

Definition 23 *A protocol is said to achieve entity authentication, if it satisfies the following requirements:*

1. *If* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$ *have matching conversations, then both of them accept.*
2. *A polynomial-time attacker only has a negligible advantage in the attack game for entity authentication.*

4.2.3 Summary of the Bellare-Rogaway model

From a corrupt query as described above, the attacker only contains a user's long-term private key. This corresponds to what is commonly referred to as the weak corruption model. In a strong corruption model, such a query returns all the ephemeral states of the unaccepted and unaborted oracles, as well as the users long-term private key.

In the Bellare-Rogaway model, the attack game for session key security covers not only the key authentication property but also possibly the following additional security properties:

- forward secrecy and perfect forward secrecy, if the attacker is allowed to compromise all the users through corrupt queries in Phase 2;
- known-key resilience and unknown-key share resilience, because the attacker is allowed to obtain other session keys through reveal queries.

However, it does not cover the following security properties:

- backward secrecy, because the attacker is not allowed access to U_i's long-term private key in Phase 1, since the tested oracle $\Pi^s_{i,j}$ should be fresh;
- key control and dishonest partner unknown key-share resilience, because the challenger will simulate the protocol execution faithfully, and no oracle queries allow the attacker to affect these behaviours;
- key confirmation and information leakage resilience.

The attack game for entity authentication covers entity authentication; however, it does not cover key compromise impersonation resilience because the attacker is not allowed access to U_i's long-term private key.

4.2.4 Variants of the Model

Since the introduction of the Bellare-Rogaway model, many variations have been proposed to cover a wide range of security properties (e.g. [24, 28, 31, 60, 68, 75, 157, 225]). We next summarise some of the more signaficant of these.

1. Bellare and Rogaway [31] modified the Bellare-Rogaway model [29] to model the security properties for three-party key establishment protocols, where a server generates and distributes a session key to two users. An important difference in this model is that partnership is defined using a special function instead of the matching conversation.
2. Bresson *et al.* [60] extended the Bellare-Rogaway model to cover group key establishment protocols. Let the participant set be $\mathcal{U} = \{U_1, U_2, \cdots, U_n\}$. The s-th ($s \geq 1$) instance of U_i is denoted by Π^s_i. The session identifier of Π^s_i is $sid^s_i = \{\mathrm{SID}_{i,j} : j \in \mathcal{U}\}$, where $\mathrm{SID}_{i,j}$ is the concatenation of all the messages that oracle Π^s_i has exchanged with an oracle of U_j during its execution. Two oracles Π^s_i and Π^t_j are defined to be partnered if they accept and one of the following two conditions is satisfied.

- $sid_i^s \cap sid_j^t \neq \phi$;
- there exist $\Pi_{u_x}^{v_x}$, where $v_x \geq 1$ and $1 \leq x \leq w$, satisfying $\Pi_{u_1}^{v_1} = \Pi_i^s$, $\Pi_{u_w}^{v_w} = \Pi_j^t$, and $sid_{u_x}^{v_x} \cap sid_{u_{x+1}}^{v_{x+1}} \neq \phi$ for any $1 \leq x \leq w-1$.

The set of possible oracle queries and the notion of freshness are defined in the Bellare-Rogaway model, as is the attack game for session key security.

Let pid_i^s be the set containing all the partnered oracles of Π_i^s. The attack game for entity authentication is defined in a similar way to the corresponding game in the Bellare-Rogaway model, except that the attacker is allowed to make reveal queries and the attacker wins if $|pid_j^t| \neq n-1$.

3. In the model proposed by Strangio [225], the attacker is required to issue its test query to a KCI-fresh oracle instead of a fresh oracle in the attack game for key authentication, where, unlike Definition 21, a KCI-fresh oracle does not require that U_i is uncorrupted when the oracle accepts. The authors of [75, 157] have proposed similar models.

4. Bellare, Pointcheval, and Rogaway [28] proposed a further variant of the Bellare-Rogaway model which has considered information leakage resilience (or resilience to password guessing attacks).

4.3 The Shoup Model

The Shoup model [217] is designed for the analysis of two-party key establishment protocols. The model works in three different corruption modes: static corruption, adaptive corruption, and strong adaptive corruption.

1. Static corruption means that the attacker can operate under a number of different aliases, but only make its decision as to whom to corrupt independently of the execution of the protocol.

2. Adaptive corruption means that the attacker can choose to corrupt an honest user, but can only obtain that user's long-term private key.

3. Strong adaptive corruption means that the attacker can choose to corrupt an honest user, obtaining the user's long-term private key and any internal data that has not been explicitly erased.

4.3.1 The Shoup model for static corruption mode

In the ideal world, the attacker can make the following oracle queries (Shoup [217] uses the term "operation" instead of "oracle query"):

1. initialize user: this operation takes the form (initialize user, i, ID_i) and assigns a unique identity ID_i to the user U_i.

2. initialize user instance: this operation takes the form (initialize user instance, $i, j, role_{i,j}, PID_{i,j}$) and specifies an instance (or an oracle) $\Pi_{i,j}$ for U_i, along with the user's role (initiator or responder) $role_{i,j}$ in the protocol execution and its partner identity PID_{ij}. After initialisation, $\Pi_{i,j}$ is defined to be active, and remains active until the execution of either an abort session or a start session operation on $\Pi_{i,j}$.

3. abort session: this operation takes the form (abort session, i, j) and aborts the oracle $\Pi_{i,j}$.

4. start session: this operation takes the form (start session, $i, j,$ *connection assignment*$[, key]$) and specifies how the session key of the oracle $\Pi_{i,j}$ is generated:

   - If *connection assignment* is *create*, then the ring master sets the session key of $\Pi_{i,j}$ to be a random string, given that $\Pi_{i,j}$ is active. After this operation, $\Pi_{i,j}$ is said to be isolated.
   - If *connection assignment* is $(connect, i', j')$, then the ring master[2] sets the session key of $\Pi_{i,j}$ to be $K_{i',j'}$ given that, $\Pi_{i,j}$ and $\Pi_{i',j'}$ are compatible and $\Pi_{i',j'}$ is isolated. After this operation, $\Pi_{i',j'}$ is no longer isolated.

     Note that $\Pi_{i,j}$ and $\Pi_{i',j'}$ are said to be compatible if $PID_{i,j} = ID_{i'}$, $PID_{i',j'} = ID_i$, and $role_{i,j} \neq role_{i',j'}$.

[2]Ring master plays the same role as the challenger in the Bellare-Rogaway model.

- If *connection assignment* is *compromise*, then the ring master sets the session key of $\Pi_{i,j}$ to be *key*, given that PID_{ij} has not been assigned to any user.

5. application: this operation takes the form (application, f) and specifies the usage of the session key by a higher-level application protocol f.

6. implementation: this operation takes the form (implementation, *comment*) and simply specifies a *comment*.

Every operation will add one record in the *transcript*; for example, an initialize user operation will add (initialize user, i, ID_i) to the *transcript*.

In the real world, the attacker can make the following oracle queries:

1. initialize user: this operation takes the form (initialize user, i, ID_i) and assigns a unique identity ID_i to the user U_i;

2. register: this operation takes the form (register, ID, *registration request*) and returns *registration receipt*.

3. initialize user instance: this operation takes the form (initialize user instance, i, j, $role_{i,j}, PID_{i,j}$) and specifies an instance (or an oracle) $\Pi_{i,j}$ for U_i, along with the user's role (initiator or responder) $role_{i,j}$ in the protocol execution and its partner identity PID_{ij};

4. deliver message: this operation takes the form (deliver message, $i, j, InMsg$), delivers the message $InMsg$ to the oracle $\Pi_{i,j}$, and returns a message $OutMsg$. In the online-TTP setting, the attacker also has access to (deliver message to TTP, $InMsg$).

5. application: this operation takes the form (application, f), and specifies the use of the session key by a higher-level application protocol f;

As in the ideal world, every operation will add one record in the *transcript*. A key establishment protocol is secure in this model in the static corruption mode, if it satisfies the following requirements:

1. Any oracle should terminate after a polynomially bounded number of message exchanges;
2. In the real world, if an attacker faithfully delivers the messages, compatible oracles end successfully and compute the same session key;
3. For any real-world attacker, there exists an ideal-world attacker such that their transcripts are computationally indistinguishable.

4.3.2 The Shoup model for other modes

In the Shoup model under the strong corruption assumption, a corruption user is enabled in the real world, and such an operation adds two records (corruption user, i) and (implementation, corruption user, LTS_i) to the *transcript*, where LTS_i is U_i's long-term state information. The corruption user operation is also enabled in the ideal world, but such an operation will only add one record (corruption user, i) to the *transcript*. In addition, the connection assignment compromise is valid if any of the following is true: (1) PID_{ij} has not been assigned to any user; (2) PID_{ij} has not been assigned to a corrupted user; (3) user U_i is corrupted.

If the strong adaptive corruption assumption is assumed, the main extension is that a (strong corruption user, i) is enabled. In the real world, through this operation, the adversary will obtain the long-term private key of U_i and all ephemeral data of U_i's active oracles. As a result, two records (strong corruption user, i) and (implementation, strong corruption user, *exposed data*) are added to the *transcript*, where *exposed data* is the information obtained by the adversary. In the ideal world, through this operation, the adversary will obtain all the session keys of $\Pi_{i',j'}$, which is isolated and $PID_{i'j'} = ID_i$. In addition, a record (strong corruption user, i) is added to the *transcript*.

4.3.3 Summary of the Shoup model

Note that in the Shoup model under all three assumptions, when a corrupted user (the attacker) establishes its session key with an uncorrupted user, it is assumed that

the attacker can set the session key to be any value through the (start session, i, j, *compromise*, *key*) query. In fact, the same assumption is adopted in [24]. It is clear that these models provide no guarantee regarding the key control property, which says that no user should fully control the session key. Therefore, in some schemes proven secure in Shoup model, it may be possible to mount attacks in which a malicious user can choose the session key.

4.4 The Canetti-Krawczyk model

4.4.1 Introduction

Canetti and Krawczyk [68] proposed a modular technique for the construction of secure two-party key establishment protocols. This technique is derived from the simulatability-based security approach proposed by Bellare, Canetti, and Krawczyk [24], while the security properties of key establishment protocols are evaluated using the indistinguishability approach proposed by Bellare and Rogaway [29]. The following notions play an important role in the Canetti-Krawczyk security definitions.

- The unauthenticated-links model (UM) is a model in which the attacker has full control of the communication links and the scheduling of all protocol events.
- The authenticated-links model (AM) is a model in which the attacker is restricted to only delivering messages generated by the genuine parties without any changes or additions to them.
- A protocol $\mathcal{P}'$ is said to *emulate* the protocol $\mathcal{P}$ (secure in the AM) in the UM if, for any attacker that interacts with $\mathcal{P}'$ in the UM, there exists an attacker that interacts with $\mathcal{P}$ in the AM such that the two interactions are computationally indistinguishable to an outside observer.
- An *authenticator* is defined to be an algorithm that takes an protocol $\mathcal{P}$ as input and outputs the description of a protocol $\mathcal{P}'$ such that $\mathcal{P}'$ emulates $\mathcal{P}$ in the UM.

Using the Canetti-Krawczyk modular approach [24, 68], a secure key agreement protocol in the UM can be constructed in two steps:

1. Construct a secure protocol $\mathcal{P}$ in the AM;
2. Apply an authenticator to $\mathcal{P}$ and then obtain a secure protocol $\mathcal{P}'$ in the UM.

4.4.2 The Canetti-Krawczyk model

In both the AM and the UM of the Canetti-Krawczyk model, session key security is defined using an attack game similar to that employed in the Bellare-Rogaway model [29]. However, an important difference between the Canetti-Krawczyk model and other security models is that it assumes that every protocol execution will take a unique session identifier as input. In other words, the session identifier is generated by some program that (directly or indirectly) invokes the key establishment protocol.

Let the users be U_i $(1 \leq i \leq N)$. In the UM of the Canetti-Krawczyk model, an attacker may have access to the following types of oracle queries.

- activate, which on the input of $(ID_i, sid_i, ID_j, role)$, where sid_i a unique session identifier and $role$ is either "initiator" or "responder", creates an oracle $\Pi_{i,j}^{sid_i}$ to starts a session with U_j.
- send, which, on the input of an oracle $\Pi_{i,j}^{sid_i}$ and a message m, delivers m to $\Pi_{i,j}^{sid_i}$.
- session-state-reveal, which, on the input of ID_i and a session identifier sid_i, returns $\Pi_{i,j}^{sid_i}$'s ephemeral internal state.
- session-key-reveal, which, on the input of an accepted[3] oracle $\Pi_{i,j}^{sid_i}$, returns the session key possessed by this oracle.
- session-expiration, which, on the input of an accepted oracle $\Pi_{i,j}^{sid_i}$, erases the session key of this oracle.

[3] An oracle $\Pi_{i,j}^{sid_i}$ is defined to be accepted if its protocol execution has successfully ended.

- corrupt, which, on the input of a user identifier ID_i, returns U_i's long-term private key and ephemeral internal states of U_i's oracles which have not ended (either accepted or aborted).
- test-session, which, on the input of a fresh oracle $\Pi_{i,j}^{sid_i}$ (see the definition below), returns a string which is computed as follows: choose a random bit b from the set $\{0, 1\}$, return the session key if $b = 1$, otherwise return a random string from the session key space.

In the Canetti-Krawczyk model, two oracles $\Pi_{i,j}^{sid_i}$ and $\Pi_{j,i}^{sid_j}$ are *partnered* if their roles are different and $sid_i = sid_j$. An oracle $\Pi_{i,j}^{sid_i}$ is said to be *fresh* if the following requirements are satisfied:

1. $\Pi_{i,j}^{sid_i}$ has accepted and it has not been issued a session-state-reveal, session-key reveal, or session-expiration query;
2. The partner oracle of $\Pi_{i,j}^{sid_i}$ has not been issued a session-state-reveal, session-key reveal;
3. U_i has not been issued a corrupt query, and U_j has not been issued a corrupt query beforing the partner oracle of $\Pi_{i,j}^{sid_i}$ is issued a session-expiration query.

The attack game for *session key security* is carried out between a two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger $\mathcal{C}$, as follows.

1. Setup: The challenger $\mathcal{C}$ generates both the public system parameters $param_1$ and private system parameters $param_2$.
2. Phase 1: The attacker runs $\mathcal{A}_1$ on the input of $param_1$. $\mathcal{A}_1$ can make the following types of queries: activate, send, session-state-reveal, session-key reveal, and corrupt. $\mathcal{A}_1$ terminates by making a test-session query on the input of a fresh oracle $\Pi_{i,j}^{sid_i}$. In addition, $\mathcal{A}_1$ also outputs some state information *state*.
3. Challenge: The challenger $\mathcal{C}$ returns the output of test-session$(\Pi_{i,j}^{sid_i})$.
4. Phase 2: The attacker runs $\mathcal{A}_2$ on the input of *state* and the output of the challenger. $\mathcal{A}_2$ can make the same types of query as $\mathcal{A}_1$ in step 1, but neither

session-state-reveal and session-key reveal query to $\Pi_{i,j}^{sid_i}$ and its partner oracle, nor corrupt query to U_i and U_j. $\mathcal{A}_2$ terminates by outputting a guess bit b'.

At the end of this game, the attacker wins if $b' = b$, and the attacker's advantage is defined to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 24 *A key establishment protocol is defined to be* SK*-secure if it achieves the following properties:*

1. *If two uncorrupted oracles have matching sessions, then they accept and compute the same session key.*
2. *Any polynomial-time attacker's advantage in the attack game for session key security is negligible.*

The session key security game does not cover perfect forward secrecy, which can be modelled by making the following modifications to the attack game for session key security.

- In Phase 1, $\mathcal{A}_1$ is allowed to issue any number of session-expiration queries.
- In Phase 2, $\mathcal{A}_2$ is allowed to issue a corrupt query to U_i (after issuing a session-expiration query to $\Pi_{i,j}^{sid_i}$).

With these modifications, a protocol secure under Definition 24 is said to be SK-secure with PFS.

The definitions in the AM are identical to these in the UM, except that send query is replaced with send* query, which means that the attacker can only faithfully deliver protocol messages.

4.4.3 Summary of the Canetti-Krawczyk model

In the Canetti-Krawczyk model, the attack game for session key security covers not only the key authentication property but also the following additional security properties: known-key resilience and unknown-key share resilience, because the attacker is allowed access to obtain other session keys through reveal queries. However, it does not cover the following security properties:

- backward secrecy, because the attacker is not allowed to U_i's long-term private key in Phase 1, since the tested oracle $\Pi^s_{i,j}$ should be fresh;
- key control and dishonest partner unknown key-share resilience, because the challenger will simulate the protocol execution faithfully, and no oracle queries allow the attacker to affect these behaviours;
- key confirmation and information leakage resilience.

In addition, entity authentication and key compromise impersonation resilience are not addressed by the Canetti-Krawczyk model.

4.5 Comparisons and Conclusions

In this chapter we have presented a brief review of the main existing security models for key establishment protocols, and shown that none of them cover all the security properties described in Section 3.3.

The capabilities of the various security models are captured in table 4.1, where the following abbreviations are used: KA: Key Authentication, FS: Forward Secrecy, PFS: Perfect Forward Secrecy, BS: Backward Secrecy, UKS: Unknown-Key Share resilience, DPUKS: Dishonest Partner Unknown-Key Share resilience, KR: Key Randomness, KCL: Key Control, EA: Entity Authentication, KK: Known-Key resilience, KCM: Key ConfirMation, KCI: Key Compromise Impersonation resilience, and IL: Information Leakage resilience.

	BR	Strangio*	Shoup	Canetti-Krawczyk	Bresson*	Bellare*
KA	yes	yes	yes	yes	yes	yes
FS	yes	yes	yes	yes	yes	yes
PFS	yes	yes	yes	yes	yes	yes
BS	no	yes	no	no	no	no
UKS	yes	yes	yes	yes	yes	yes
DPUKS	no	no	no	no	no	no
KR	yes	yes	yes	yes	yes	yes
KCL	no	no	no	no	no	no
EA	yes	no	no	no	yes	no
KK	yes	yes	yes	yes	yes	yes
KCM	no	no	no	yes	no	no
KCI	no	yes	no	no	no	no
IL	no	no	no	no	no	yes

Table 4.1: Comparison between Security Models

Note that the BR stands for the Bellare-Rogaway model, Strangio* stands for the security models in [75, 157, 225], Bresson* stands for the security model proposed by Bresson *et al.* [60], and Bellare* stands for the security model proposed by Bellare, Pointcheval, and Rogaway [28].

Not only none of these models is capable of covering the full list of security properties, their concepts of partnership have certain shortcomings. In the Bellare-Rogaway model and its variants, partnership also depends on complete matching conversations between the communicating parties, which is actually a extremely strong requirement. The Canetti-Krawczyk model assumes a pre-distributed unique session identifier; however, this is also a very strong requirement, which may cause additional communication and computation complexity when implementing the key establishment protocols.

In Chapter 6, we propose a new indistinguishability-based security model in an attempt to address these issues.

Chapter 5

Attacks against key establishment protocols

Contents

In this chapter we analyse a number of key establishment protocols and demonstrate certain security vulnerabilities in these protocols. Some of these vulnerabilities have arisen because of the lack of a rigorous analysis in a well-defined security model, while others are beyond the security models originally used to analyse the protocols.

5.1 Introduction

In this chapter, we provide security analyses of a number of key establishment protocols; some of these analyses have appeared in [195, 232, 233, 234, 235, 236, 237]. These security analyses reflect a number of issues in the design and security analysis of key establishment protocols. They help us to understand the precise security properties of existing key establishment protocols, and improve our understanding of the existing security models and their capability to model security properties. They also provide a motivation for the development of a new security model (in the next chapter), which addresses some of the issues identified in existing protocols and models.

The rest of this chapter is organised as follows. In Section 5.2, we provide a security analysis of some extensions of the Burmester-Desmedt protocol, including the authenticated version proposed by Burmester and Desmedt [61], the Choi-Hwang-Lee protocol [80] and the Du-Wang-Ge-Wang protocol [97], and the modified versions given in [98, 253], and a protocol generated by the Katz-Yung compiler [145]. In Section 5.3 we provide security analyses of the second version of the Wireless Authentication Infrastructure (WAI) protocol from the Chinese WLAN implementation plan [3]. In Section 5.4 we provide some examples of protocols, namely those given in [116, 119, 217], which suffer from dishonest partner unknown key-share attacks. In Section 5.5, we present an analysis of certain password-based key establishment protocols, include the Jablon protocol [130], the Lai-Ding-Huang protocol [162], the EKE-U and EKE-M protocols [66], and the RSA-AKE protocol [216]. In the final section we conclude this chapter.

5.2 Extensions of the Burmester-Desmedt protocol

In this section we provide security analyses of a number of authenticated versions of the Burmester-Desmedt protocol, described in Section 3.1.2.2, including those given in [61, 80, 97, 98, 145, 253].

5.2.1 Analysis of the Authenticated Burmester-Desmedt protocol

The authenticated Burmester-Desmedt protocol [61] was designed to improve the security of the unauthenticated version (described in Section 3.1.2.2) by providing partial authentication for the protocol messages using a zero knowledge proof scheme. We show that the authenticated protocol suffers from attacks against both the key authentication property and the entity authentication property. Note that these attacks appear in the work of Tang and Mitchell [235].

5.2.1.1 A proof of knowledge scheme

We first describe a scheme which is a key component of the authenticated Burmester-Desmedt protocol.

In the initialisation stage, the system selects four large primes p_1, p_2, q_1, q_2 satisfying $p_1 \leq q_2$, $q_1|(p_1 - 1)$, and $q_2|(p_2 - 1)$. Let g_1 be a generator of a multiplicative group of order q_1 in $\mathbb{Z}_{p_1}^*$, and g_2 be a generator of a multiplicative group of order q_2 in $\mathbb{Z}_{p_2}^*$. The public system parameters are $(p_1, p_2, q_1, q_2, g_1, g_2)$.

User U_i $(i \geq 1)$ chooses $a_{i1} \in_R \mathbb{Z}_{q_1}, a_{i2}, a_{i3} \in_R \mathbb{Z}_{q_2}$, publishes its public key $(\beta_{i1}, \beta_{i2}, \beta_{i3})$, where $\beta_{i1} = g_1^{a_{i1}}, \beta_{i2} = g_2^{a_{i2}}, \beta_{i3} = g_2^{a_{i3}}$, and keeps (a_{i1}, a_{i2}, a_{i3}) as its private key.

Suppose U_i wishes to prove its knowledge of z to U_j $(j \neq i)$; the scheme operates as follows.

1. U_i sends z and $\gamma_{i1} = g_1^{b_{i1}}$ to U_j, where $b_{i1} \in_R \mathbb{Z}_{q_1}$.
2. U_i proves to U_j that it knows the discrete logarithm base g_1 of $\beta_{i1}^z \gamma_{i1}$ and the discrete logarithm base g_2 of $\beta_{i2}^{\gamma_{i1}} \beta_{i3}$, using the zero-knowledge discrete logarithm proof scheme of Chaum *et al.* [72], described below.
3. U_j checks that $\gamma_{i1}^{q_1} \equiv 1 \pmod{p_1}$, $g_1^{q_1} \equiv \beta_{i1}^{q_1} \equiv 1 \pmod{p_1}$, $g_2^{q_2} \equiv \beta_{i2}^{q_2} \equiv \beta_{i3}^{q_2} \pmod{p_2}$, that p_1, p_2 are primes, and that $p_1 \leq q_2$. If any of the checks fail, U_j terminates the protocol. Otherwise U_j now believes that U_i knows z.

The zero knowledge discrete logarithm proof scheme, due to Chaum *et al.* [72], operates as follows. Suppose P is a large prime, and that $\alpha^x \equiv \beta \pmod{P}$. Suppose also that P, α, β are made public and x is a secret of Alice. If Alice wants to prove her knowledge of x to Bob, she uses the following procedure.

1. Alice selects T numbers $e_i \in_R \mathbb{Z}_{P-1}$ $(1 \leq i \leq T)$. Alice computes and sends $h_i = \alpha^{e_i} \bmod P$ $(1 \leq i \leq T)$ to Bob.
2. Bob chooses and sends T bits $b_i \in_R \{0, 1\}$ $(1 \leq i \leq T)$ to Alice.
3. For each bit b_i $(1 \leq i \leq T)$, if $b_i = 0$ Alice sets $s_i = e_i$; otherwise Alice computes $s_i = e_i - e_j \bmod (P-1)$, where j is the minimal number for which $b_j = 1$. Finally, Alice sends $(x - e_j) \bmod (P-1)$ and s_i $(1 \leq i \leq T)$ to Bob.
4. For each bit i $(1 \leq i \leq T)$, if $b_i = 0$ Bob checks that $\alpha^{s_i} = h_i$; otherwise Bob checks that $\alpha^{s_i} = h_i h_j^{-1}$. In addition, Bob also checks that $\alpha^{x-e_j} = \beta h_j^{-1}$. If all the checks succeed, Bob accepts.

5.2.1.2 Description of the Authenticated Burmester-Desmedt protocol

The TTP runs the TTP initialisation sub-protocol to generate the public parameter $(p_1, p_2, q_1, q_2, g_1, g_2)$ for the authentication scheme described in Section 5.2.1.1. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i $(i \geq 1)$ runs the user initialisation sub-protocol to generate its public/private keys $(\beta_{i1}, \beta_{i2}, \beta_{i3})$ and (a_{i1}, a_{i2}, a_{i3}), where $a_{i1} \in_R \mathbb{Z}_{q_1}, a_{i2}, a_{i3} \in_R \mathbb{Z}_{q_2}$ and $\beta_{i1} = g_1^{a_{i1}}, \beta_{i2} = g_2^{a_{i2}}, \beta_{i3} = g_2^{a_{i3}}$. U_i should get its public key certified by the TTP. In addition, this sub-protocol generates (p, q, g), where $q | p-1$ and g is a generator of the multiplicative sub-group of order q in $\mathbb{Z}_p^*$.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol is defined as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $s_i \in_R \mathbb{Z}_q$, and computes and broadcasts $Z_i = g^{s_i}$.

2. After receiving Z_{i-1} and Z_{i+1}, U_i proves its knowledge of Z_i to U_{i+1}, and verifies U_{i-1}'s knowledge of Z_{i-1}. If both the proof and the verification succeed, U_i computes and broadcasts X_i:

$$X_i = (Z_{i+1}/Z_{i-1})^{s_i}$$

3. After receiving X_j $(1 \leq j \leq n,\ j \neq i)$, U_i computes the session key K_i as:

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1s_2+s_2s_3+s_3s_4+\cdots+s_ns_1} \end{aligned}$$

Burmester and Desmedt [61] claim that this is a secure key establishment protocol in the sense that, in any session, it is computationally infeasible for any active (outside or inside) attacker to mount an attack against the key authentication property.

5.2.1.3 Security Vulnerabilities of the Authenticated Protocol

We show that the authenticated Burmester-Desmedt protocol suffers from attacks against both the key authentication property and the entity authentication property. Note that the latter property is not considered in [61].

Attack against the key authentication property: In any target session, the attacker replaces the message Z_{i+1} broadcast by U_{i+1} sent to U_i with $Z'_{i+1} = Z^2_{i-1}$, for every i $(1 \leq i \leq n)$. We now show that the protocol will end successfully and the attacker can compute the session key held by U_i $(1 \leq i \leq n)$.

Lemma 2 *Under the above attack, if all parties follow the protocol correctly (except for the attacker itself) and all messages are successfully delivered, then the protocol will end successfully, and the attacker can compute the session key of* U_i.

Proof. Under the attack, it is clear that the protocol will end successfully, because it is only required that U_i $(1 \leq i \leq n)$ proves his knowledge of Z_i to U_{i+1} (while the attacker only changes the message that U_{i+1} sends to U_i).

It is also clear that, in the second step, U_i will broadcast $X_i = (Z'_{i+1}/Z_{i-1})^{s_i} = (Z_{i-1})^{s_i}$. Then, after intercepting all the broadcast values X_i $(1 \leq i \leq n)$, the attacker can compute the session key held by U_i as

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= (X_i)^n \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \end{aligned}$$

which involves only values broadcast by the various recipients. The result follows. □

Attack against the entity authentication property: Before describing the attack, we first describe a vulnerability of the zero-knowledge discrete logarithm proof scheme. The vulnerability arises from the fact that the proof scheme does not enable the prover to specify the verifier.

Suppose Alice wishes to prove her knowledge of x to Bob; then an attacker can concurrently impersonate Alice to prove knowledge of x to any other entity, Carol say. The attack can be mounted as follows. The attacker intercepts the first and third protocol messages sent by Alice to Bob in the first instance of the protocol, and forwards them to Carol (pretending to be Alice). The second protocol message sent by Carol to Alice is intercepted by the attacker and forwarded to Alice as if it comes from Bob. It is straightforward to verify that Carol will believe that the attacker knows x.

The attack against the entity authentication property can be mounted as follows. If the attacker detects that a set S of users U_i $(1 \leq i \leq n)$ start a session to negotiate a session key, it impersonates U_i to start a second session among a set S' of users, where S' contains U_{i-1}, U_i, U_{i+1} and such that these three users are ordered in the same way as they are in S. In the second session, the attacker impersonates U_i. During the protocol executions, the attacker performs as follows.

1. In the first session, the attacker intercepts the messages sent from U_{i-1} and U_{i+1} to U_i, and prevents them from reaching U_i. In the second session, the attacker impersonates U_i to broadcast the value of Z_i which was broadcast by U_i in the first session. Suppose that, in the second session, the messages broadcast by U_{i-1} and U_{i+1} are Z'_{i-1} and Z'_{i+1}, respectively; the attacker then

impersonates U_{i-1} and U_{i+1} to send Z'_{i-1} and Z'_{i+1} to U_i in the first session.

2. In the first session, when U_i proves its knowledge of Z_i to U_{i+1}, the attacker mounts the above attack (against the zero-knowledge discrete logarithm proof scheme) by impersonating U_i to prove its knowledge of Z_i to U_{i+1} in the second session. In the second session, when U_{i-1} proves its knowledge of Z'_{i-1} to the attacker, the attacker mounts the above attack (against zero-knowledge discrete logarithm proof scheme) by impersonating U_{i-1} to prove its knowledge of Z'_{i-1} to U_i in the first session.

   As a result, in the first session U_i computes and broadcasts X_i which is computed as follows:

$$X_i = (Z'_{i+1}/Z'_{i-1})^{s_i}$$

   The attacker intercepts this message and impersonates U_i to broadcast the same message in the second session.

It is straightforward to verify that the second session ends successfully which means the attacker succeeds in impersonating U_i to other users, and the first session fails, because the proof between U_i and U_{i+1} is hijacked by the attacker so that U_{i+1} aborts. In this attack, U_i believes it is negotiating a session key with the users in S withouting noticing the key establishment activity in the group S', therefore, if the attacker can compromise the ephemeral state (the secret s_i for generating Z_i in the first session) of U_i then it can obtain the session key for the second session. As a result, it is straightforward to verify that this protocol is does not achieve session key security.

5.2.2 Analysis of the Choi-Hwang-Lee and Du-Wang-Ge-Wang Protocols

The Choi-Hwang-Lee protocol is an authenticated variant (using bilinear pairings) of the Burmester-Desmedt protocol [80]. The Du-Wang-Ge-Wang protocol [97] is a similar variant. Zhang and Chen [253] describe an attack against the entity authentication property that can be mounted against both the Choi-Hwang-Lee and Du-Wang-Ge-Wang protocols. They also proposed a modified scheme designed to prevent this impersonation attack. In addition, Du *et al.* [98] proposed a different

modification to the Du-Wang-Ge-Wang protocol to address the same attack. We show that both the modified versions still suffer from attacks against the entity authentication property by dishonest partners. Note that these attacks appear in the work of Tang and Mitchell [230].

5.2.2.1 Description of the Protocols

We first describe the Du-Wang-Ge-Wang protocol. The TTP runs the TTP initialisation sub-protocol to generate $\{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, \mathsf{H}, \mathsf{H}_1, s, P_{pub}\}$, where $\mathbb{G}_1$ is a cyclic additive group with generator P whose order is a prime q, $\mathbb{G}_2$ is a cyclic multiplicative group with the same order q, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map, two hash functions $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$, $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $s \in_R \mathbb{Z}_q$ is a global master-key, and $P_{pub} = sP$ is a public key. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to retrieve a public/private key pair (Q_i, s_i) from the TTP, where $ID_i \in \mathbb{Z}_q^*$ is U_i's unique identifier, $Q_i = \mathsf{H}_1(ID_i)$, and $S_i = sQ_i$.

Suppose a set of users U_i ($1 \leq i \leq n$) wish to establish a session key; then the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i selects $r_i \in_R \mathbb{Z}_q$, and broadcasts (Y_i, T_i), where $Y_i = r_iP$ and $T_i = \mathsf{H}(Y_i)S_i + r_iP_{pub}$.

2. After receiving (Y_j, T_j) from U_j ($1 \leq j \leq n, j \neq i$), U_i verifies that:

$$\hat{e}(\sum_{j \in \{1,\cdots,n\}, j \neq i} T_j, P) = \hat{e}(\sum_{j \in \{1,\cdots,n\}, j \neq i} (\mathsf{H}(Y_j)Q_j + Y_j), P_{pub})$$

   If all the checks succeed, U_i broadcasts $X_i = \hat{e}(P_{pub}, r_i(Y_{i+1} - Y_{i-1}))$; otherwise it aborts.

3. After receiving X_j from U_j ($1 \leq j \leq n, j \neq i$), U_i computes its session key K_i as:

$$\begin{aligned} K_i &= \hat{e}(P_{pub}, nr_iY_{i-1})X_i^{n-1}X_{i+1}^{n-2}\cdots X_{i-2} \\ &= \hat{e}(P,P)^{(r_1r_2 + r_2r_3 + \cdots + r_nr_1)s} \end{aligned}$$

The key establishment sub-protocol of the Choi-Hwang-Lee protocol works in a similar way, except that the computations in step 2 and 3 are slightly different:

- In step 2, the verification equation is:

$$\hat{e}(T_{i-1} + T_{i+1} + T_{i+2}, P) = \hat{e}(\sum_{j=\{i-1,i+1,i+2\}} (\mathsf{H}(Y_j)Q_j + Y_j), P_{pub})$$

  Then, U_i computes and broadcasts $X_i = \hat{e}(r_i(Y_{i+2} - Y_{i-1}), Y_{i+1})$.

- In step 3, the session key is computed as:

$$\begin{aligned} K &= \hat{e}(nr_iY_{i-1}, Y_{i+1})X_i^{n-1}X_{i+1}^{n-2}\cdots X_{i-2} \\ &= \hat{e}(P,P)^{r_1r_2r_3+\cdots+r_nr_1r_2} \end{aligned}$$

5.2.2.2 Existing Attacks and Improvements

Zhang and Chen [253] show that both the Choi-Hwang-Lee and Du-Wang-Ge-Wang Protocols suffer from impersonation attacks against the entity authentication property from dishonest partners. Because of the similarity of these attacks, we only describe the attack on the Du-Wang-Ge-Wang protocol.

Suppose that, in a past session, users U_i $(1 \leq i \leq n)$ have successfully established a session key. Suppose also that U_i sent (Y_i^*, T_i^*) in step 1, where $Y_i^* = r_i^*P$, and $T_i^* = \mathsf{H}(Y_i^*)S_i + r_i^*P_{pub}$.

With (Y_i^*, T_i^*) and X_i^*, U_{i-1} and U_{i+1} can collude to impersonate U_i in a new session among any group of users, as long as U_{i-1}, U_i, and U_{i+1} are involved. Let the group of users be U_i $(1 \leq i \leq n)$. To mount the attack, U_{i+1} and U_{i-1} impersonate U_i to broadcast (Y_i^*, T_i^*) in step 1, and impersonate U_i to broadcast X_i in step 2, where

$$\begin{aligned} X_i &= \hat{e}(P_{pub}, r_i^*(Y_{i+1} - Y_{i-1})) \\ &= \hat{e}(r_i^*P, s(Y_{i+1} - Y_{i-1})) \\ &= \hat{e}(Y_i^*, (r_{i+1} - r_{i-1})P_{pub}) \end{aligned}$$

It is straightforward to verify that the new session will end successfully, and all users will compute the same session key.

In order to avoid this attack, Zhang and Chen [253] proposed the following modified key establishment sub-protocol.

1. U_i chooses $r_i \in_R \mathbb{Z}_q$, and broadcasts (Y_i, T_i), where t_i is a time stamp, $Y_i = r_i P$, and $T_i = \mathsf{H}(Y_i||t_i||ID_1||\cdots||ID_n)S_i + r_i P_{pub}$.

2. After receiving (Y_j, T_j) from U_j $(1 \leq j \leq n, j \neq i)$, U_i verifies that:

$$\hat{e}(\sum_{j\in\{1,\cdots,n\},j\neq i} T_j, P) = e(\sum_{j\in\{1,\cdots,n\},j\neq i} (H(Y_j||t_{auth})Q_j + Y_j), P_{pub}) \quad (5.1)$$

   where $t_{auth} = t_i||ID_1||\cdots||ID_n$.

   If all the checks succeed, U_i computes and broadcasts $X_i = \hat{e}(P_{pub}, r_i(Y_{i+1} - Y_{i-1}))$; otherwise it aborts.

3. After receiving X_j from U_j $(1 \leq j \leq n, j \neq i)$, U_i computes its session key K_i as:

$$\begin{aligned} K_i &= \hat{e}(P_{pub}, nr_i Y_{i-1}) X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i-2} \\ &= \hat{e}(P, P)^{(r_1 r_2 + r_2 r_3 + \cdots + r_n r_1)s} \end{aligned}$$

It is straightforward to see that the protocol ends successfully only if the same timestamp is used by all parties. Hence, the protocol requires a strict synchronisation of time.

Du *et al.* [98] proposed another modified version of the Du-Wang-Ge-Wang protocol, in which synchronous counters are held by every group of users which may run the protocol to establish a session key. That is, if a user is involved in t different groups which at some time jointly establish a session key, then that user will need to maintain t distinct counters, one for each such user group. It is assumed that every counter's initial value is 1; after a successful protocol execution, the counter is increased by 1.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; then the modified key establishment sub-protocol works as follows:

1. Suppose c is the current value of the counter held by U_i for the set of users

$\{U_1, U_2, \cdots, U_n\}$. User U_i computes and broadcasts $(Y_i,\ T_i)$, where $r_i \in_R \mathbb{Z}_q$, $Y_i = r_i P$, and $T_i = \mathsf{H}(Y_i) c S_i + r_i P_{pub}$.

2. After receiving (Y_j, T_j) from U_j $(1 \leq j \leq n, j \neq i)$, U_i verifies that:

$$\hat{e}(\sum_{j \in \{1,\cdots,n\}, j \neq i} T_j, P) = \hat{e}(\sum_{j \in \{1,\cdots,n\}, j \neq i} (\mathsf{H}(Y_j) c Q_j + Y_j), P_{pub}) \tag{5.2}$$

   If all the checks succeed, U_i broadcasts $X_i = \hat{e}(P_{pub}, r_i(Y_{i+1} - Y_{i-1}))$; otherwise it aborts.

3. After receiving X_j from U_j $(1 \leq j \leq n, j \neq i)$, U_i computes its session key K_i as:

$$\begin{aligned} K_i &= \hat{e}(P_{pub}, n r_i Y_{i-1}) X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i-2} \\ &= \hat{e}(P, P)^{(r_1 r_2 + r_2 r_3 + \cdots + r_n r_1) s} \end{aligned}$$

   and updates the value of the counter it holds for the set of users $\{U_1, U_2, \cdots, U_n\}$ to $c + 1$.

5.2.2.3 Further Security Vulnerabilities

Although the authors of [98, 253] have modified the Du-Wang-Ge-Wang protocol to prevent the attacks against the entity authentication property, we show that these modified protocols are still vulnerable to attacks from a subset of malicious participants. In other words, these protocols cannot achieve entity authentication in the presence of dishonest partners.

1. Firstly, in the modified Du-Wang-Ge-Wang protocol of Zhang and Chen, three or more users can collude to impersonate another user and make all users compute a common session key. Suppose that a set of users U_i $(1 \leq i \leq n, n > 4)$ run the protocol to establish a session key. We show that U_{i-1}, U_i, and U_{i+1} are able to impersonate another user U_j.

   In any session, if all values are exchanged successfully amongst the users, then equation 5.1 will hold for all users other than U_i and U_j if the following equation holds.

$$\hat{e}(T_i + T_j, P) = \hat{e}(\mathsf{H}(Y_i || t_{auth}) Q_i + Y_i + \mathsf{H}(Y_j || t_{auth}) Q_j + Y_j, P_{pub}) \tag{5.3}$$

To mount the attack, U_{i-1}, U_i, and U_{i+1} perform as follows.

(a) In step 1, U_i impersonates U_j to broadcast $(Y_j,\ T_j)$, where $r_j \in_R \mathbb{Z}_q$, $Y_j = r_j P$, and T_j is any element in $\mathbb{G}_1$. U_i also broadcasts its own message (Y_i, T_i), where

$$Y_i = -H(Y_j||t_{auth})Q_j,\ T_i = H(Y_i||t_{auth})S_i + r_j P_{pub} - T_j.$$

It is straightforward to verify that equation 5.3 holds, so that equation 5.1 holds. As a result, in step 2, the verification by U_k $(k \neq i, k \neq j)$ will succeed.

(b) In step 2, U_i impersonates U_j to broadcast $X_j = \hat{e}(P_{pub}, r_j(Y_{j+1} - Y_{j-1}))$, and then broadcasts its own message $X_i = \hat{e}(P_{pub}, r_i^*(Y_{i+1} - Y_{i-1}))$, where r_i^* is any value from $\mathbb{Z}_q^*$. U_{i-1} broadcasts $X_{i-1} = \hat{e}(P_{pub}, r_{i-1}(r_i^* P - Y_{i-2}))$. U_{i+1} broadcasts $X_{i+1} = \hat{e}(P_{pub}, r_{i+1}(Y_{i+2} - r_i^* P))$.
As a result, U_m $(m \neq i,\ m \neq i+1)$ will compute the common session key as:

$$\begin{aligned} K_m &= \hat{e}(P_{pub}, n r_m Y_{m-1}) X_m^{n-1} X_{m+1}^{n-2} \cdots X_{m-2} \\ &= \hat{e}(P, P)^{(r_1 r_2 + r_2 r_3 + \cdots + r_{i-1} r_i^* + r_i^* r_{i+1} + \cdots + r_n r_1)s} \end{aligned}$$

The use of a timestamp does not help to prevent this attack. It is worth noting that a directly analogous attack can also be mounted on the modified Choi-Hwang-Lee protocol of Zhang and Chen [253].

2. Secondly, the modified protocol of Du *et al.* [98] suffers from attacks against entity authentication from dishonest partners. Let c and c' be the counters corresponding to the sets of users $S = \{U_1, U_2, \cdots, U_n\}$ and $S' = \{U_1, U_2, \cdots, U_{n+1}\}$, respectively. We show that an outside attacker is able to impersonate user U_j in S' (or S) when the counters associated with the two sets of users happen to coincide.

   In any session for S', if all values are exchanged successfully amongst the users, then equation 5.2 will hold, for all users other than U_j if the following equation holds.

$$\hat{e}(T_j, P) = \hat{e}(\mathsf{H}(Y_j) c' Q_j + Y_j, P_{pub}) \tag{5.4}$$

   To mount the attack, the attacker records U_i's messages (Y_j^*, T_j^*) in a protocol execution for group S, and perform as follows in a protocol execution for the group S' when $c' = c$.

(a) In step 1, U_i impersonates U_j to broadcast (Y_j^*, T_j^*). It is straightforward to verify that the equation 5.4 holds because $c' = c$, so that equation 5.2 holds if all other users broadcast their message honestly. As a result, in step 2, the verification by U_k $(k \neq j)$ will succeed.

(b) In step 2, U_i impersonates U_j to broadcast $X_j = (\prod_{k=1,k\neq j}^{n} X_k)^{-1}$. If all other users broadcast their message honestly, then U_m $(m \neq j)$ will compute the common session key as:

$$\begin{aligned} K_m &= \hat{e}(P_{pub}, nr_m Y_{m-1}) X_m^{n-1} X_{m+1}^{n-2} \cdots X_{m-2} \\ &= \hat{e}(P,P)^{(r_1r_2 + r_2r_3 + \cdots + r_{j-1}r_j^* + r_j^* r_{j+1} + \cdots + r_n r_1)s} \end{aligned}$$

In summary, our attacks and the attacks proposed in [253] all arise as a result of the lack of direct authentication of the key materials (X_i) that are used to generate the session key. So, in all the protocols, even if a key is successfully generated, impersonation attacks might have occurred during the key establishment process.

5.2.3 Analysis of the Katz-Yung Protocol with Key Confirmation

The Katz-Yung protocol is obtained by applying the compiler of Katz and Yung [145] to the Burmester-Desmedt protocol, and was originally given as an example of the use of the compiler. Katz and Yung [145] also suggested adding a message to achieve key confirmation. We show that the resulting protocol suffers from attacks against the key confirmation property by dishonest partners. Note that this attack appears in the work of Tang and Mitchell [235].

5.2.3.1 Description of the Katz-Yung Protocol with Key Confirmation

The TTP runs the TTP initialisation sub-protocol to generate $(\mathbb{G}, q, g)$ and a pseudo-random function F: $\{0,1\}^k \times \{0,1\}^\ell \rightarrow \{0,1\}^\ell$, where $\mathbb{G}$ is a multiplicative cyclic group of prime order q, g is a generator of $\mathbb{G}$, and k is the bit-length of an element of $\mathbb{G}$. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i $(i \geq 1)$ runs the user initialisation sub-protocol to

generate a key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{sign}, \mathsf{Verify})$, and get pk_i certified by the TTP.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and broadcasts $(ID_i, 0, r_i)$.
2. After receiving $(ID_j, 0, r_j)$ $(1 \leq j \leq n, j \neq i)$, U_i generates its state information as $nonce_i = ID_1||r_1||ID_2||r_2||\cdots||ID_n||r_n$. U_i then chooses $s_i \in_R \mathbb{Z}_q$ and broadcasts $(ID_i, 1, Z_i, \sigma_{i1})$, where $Z_i = g^{s_i}$ and $\sigma_{i1} = \mathsf{Sign}(1||Z_i||nonce_i, sk_i)$.
3. After receiving $(ID_j, 1, Z_j, \sigma_{j1})$ $(1 \leq j \leq n, j \neq i)$, U_i checks that: (1) U_j is an intended user, (2) 1 is the next expected sequence number for a message from U_j, and (3) $\mathsf{Verify}(1||Z_j||nonce_i, pk_j, \sigma_{j1}) = 1$. If any of these checks fail, U_i terminates the protocol. Otherwise, U_i broadcasts $(ID_i, 2, X_i, \sigma_{i2})$, where $X_i = (Z_{i+1}/Z_{i-1})^{s_i}$ and $\sigma_{i2} = \mathsf{Sign}(2||X_i||nonce_i, sk_i)$.
4. After receiving $(ID_j, 2, X_j, \sigma_{j2})$ $(1 \leq j \leq n,\ j \neq i)$, U_i checks that: (1) U_j is an intended user, (2) 2 is the next expected sequence number for a message from U_j, and (3) $\mathsf{Verify}(2||X_j||nonce_i, pk_j, \sigma_{j2}) = 1$. If any of these checks fail, U_i terminates the protocol. Then U_i computes the session key K_i as follows:

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1s_2+s_2s_3+s_3s_4+\cdots+s_ns_1} \end{aligned}$$

   and broadcasts $(ID_i, 3, M_i, \sigma_{i3})$, where $M_i = \mathsf{F}(K_i, ID_i)$ and $\sigma_{i3} = \mathsf{Sign}(M_i, sk_i)$.
5. After receiving $(ID_j, 3, M_j, \sigma_{j3})$ $(1 \leq j \leq n, j \neq i)$, U_i checks that: (1) U_j is an intended user, (2) 3 is the next expected sequence number for a message from U_j, and (3) $\mathsf{Verify}(x_i, pk_j, \sigma_{j3}) = 1$. If any of these checks fail, U_i terminates the protocol.

5.2.3.2 A Security Vulnerability

We show that the above protocol suffers from attacks against the key confirmation property if dishonest partners exist. Specifically, we show that any $n-2$ malicious

users can make the other two users compute different keys as a result of the protocol. However, it is important to note that insider attacks are not covered by the security model used in the original Katz-Yung paper [145] and hence this attack is outside the model originally used to analyse the protocol.

For simplicity we describe the attack in the three-party case. Suppose, in a finished session of the Katz-Yung protocol involving three users U_i $(1 \leq i \leq 3)$, the key confirmation message sent by U_3 is $M_3^* = \mathsf{F}(K_3^*, ID_3)$ and $\sigma_{33}^* = \mathsf{Sign}(M_3^*, sk_3)$. U_2 can now initiate a new session among the same triple of users U_i $(1 \leq i \leq 3)$ and mount an attack as follows.

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and broadcasts $(ID_i, 0, r_i)$.
2. U_i generates its state information $nonce_i = ID_1||r_1||ID_2||r_2||ID_3||r_3$ and broadcasts $(ID_i, 1, Z_i, \sigma_{i1})$, where $s_i \in_R \mathbb{Z}_q$, $Z_i = g^{s_i}$, and $\sigma_{i1} = \mathsf{Sign}(1||Z_i||nonce_i, sk_i)$.
3. After receiving $(ID_j, 1, Z_j, \sigma_{j1})$ $(1 \leq j \leq 3, j \neq i)$, U_i checks the messages as required by the protocol specification. In this step, U_1 computes and broadcasts $(ID_1, 2, X_1, \sigma_{12})$, where

$$X_1 = (Z_2/Z_3)^{s_1},\ \sigma_{12} = \mathsf{Sign}(2||X_1||nonce_1, sk_1).$$

Analogously, U_3 computes and broadcasts $(ID_3, 2, X_3, \sigma_{32})$.

$$X_3 = (Z_1/Z_2)^{s_3},\ \sigma_{32} = \mathsf{Sign}(2||X_3||nonce_3, sk_3)$$

U_2 then waits until it has received $(ID_1, 2, X_1, \sigma_{12})$ and $(ID_3, 2, X_3, \sigma_{32})$, and then sends $(ID_2, 2, X_2', \sigma_{22}')$, and $(ID_2, 2, X_2, \sigma_{22})$ to U_1 and U_3, respectively, where

$$X_2 = (Z_3/Z_1)^{s_2}, K_2 = Z_1^{3s_2} X_2^2 X_3,\ \sigma_{22} = \mathsf{Sign}(2||X_2||nonce_2, sk_2)$$

$$X_2' = X_2 \cdot \frac{K_3^*}{K_2},\ \text{and}\ \sigma_{22}' = \mathsf{Sign}(2||X_2'||nonce_2, sk_2)$$

As a result of steps 1–3, U_1 will compute the session key as K_3^*, and U_3 will compute the session key as K_2. Hence, as a result, U_2 shares the session keys K_3^* and K_2 ($K_2 \neq K_3^*$ holds with an overwhelming probability) with U_1 and U_3, respectively.

4. U_2 intercepts the confirmation messages between U_1 and U_3 and prevents them from reaching their destinations. U_2 computes and sends $(ID_2, 3, M_2', \sigma_{23}')$ and $(ID_2, 3, M_2, \sigma_{23})$ to U_1 and U_3, respectively, where $M_2' = \mathsf{F}(K_3^*, ID_2)$, $\sigma_{23}' = \mathsf{Sign}(M_2', sk_2)$, $x_2 = \mathsf{F}(K_2, ID_2)$, and $\sigma_{23} = \mathsf{Sign}(M_2, sk_2)$. In addition, U_2 impersonates U_3 to sends $(ID_3, 3, M_3^*, \sigma_{33}^*)$ to U_1

   Note that U_2 has forced U_1 to compute the session key K_3^* which is chosen by U_2, and also obtained U_1's confirmation message. Following the same procedure, U_2 initiates a new session among U_i $(1 \leq i \leq 3)$, in step 3 of the new session manipulates its message sent to U_1 and forces U_1 to compute the session key as K_2, and then obtains U_1's confirmation message $M_1' = \mathsf{F}(K_2, ID_1)$ and $\sigma_{13} = \mathsf{Sign}(M_1', sk_1)$.

   In the first session, U_2 impersonates U_1 to send $(ID_1, 3, M_1', \sigma_{13}')$ to U_3, where $M_1' = \mathsf{F}(K_2, ID_1)$ and $\sigma_{13} = \mathsf{Sign}(M_1', sk_1)$ to U_3.

It is straightforward to verify that all the key confirmation messages will be checked successfully by U_1 and U_3, and the attack will therefore succeed. The above security vulnerability can be removed if U_i $(1 \leq i \leq 3)$ is required to compute its key confirmation message as $M_i = \mathsf{F}(K_i, ID_i)$, $\sigma_{i3} = \mathsf{Sign}(3||M_i||nonce_i, sk_i)$.

5.3 Analysis of the Second Version of the WAI protocol

The Wireless Authentication and Privacy Infrastructure (WAPI) scheme is the security mechanism employed in the Chinese Wireless LAN standard [1, 2, 3, 14]. The WAPI scheme has two sub-modules: Wireless Authentication Infrastructure (WAI) and Wireless Privacy Infrastructure (WPI). The WAI protocol realises the functionality of authentication and key establishment between mobile Stations (STAs) and Access Points (APs), while the WPI works on top of WAI to provide security guarantees for data communication.

In this section we first describe the WAI protocol, and then show that it possesses certain undesirable properties. Note that these attacks appear in the work of Tang and Mitchell [227].

5.3.1 Description of the protocol

Three types of entities are involved in the WAI protocol: Authentication Service Unit (ASU), Access Point (AP), and Station (STA).

- ASU is a TTP for STA and AP, and it generates a verify/sign key pair (pk_{asu}, sk_{asu}) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.
- STA possesses a public/private key pair (pk_{sta}, sk_{sta}) for a public-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and retrieves a certificate $Cert_{sta}$ for pk_{sta} from the ASU.
- AP possesses a public/private key pair $(pk_{ap,e}, sk_{ap,e})$ for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, and a verify/sign key pair $(pk_{ap,s}, sk_{ap,s})$ for $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. AP retrieves a certificate $Cert_{ap}$ for $pk_{ap,e}$ and $pk_{ap,s}$ from the ASU.

In addition, the system uses two hash functions $\mathsf{H}_1 : \{0,1\}^* \longrightarrow \{0,1\}^{2\ell}$ and $\mathsf{H}_2 : \{0,1\}^* \times \{0,1\}^* \longrightarrow \{0,1\}^{\ell}$, where ℓ is the security parameter.

When STA and AP wish to authenticate each other and establish a shared secret session key, they perform the following procedure.

1. AP sends a authentication request to STA, where the request contains the type information "0", which indicates that the message is a request.
2. After receiving the authentication request from AP, STA sends its certificate $Cert_{sta}$ and a timestamp t_{sta} to AP.
3. After receiving $Cert_{sta}$ and t_{sta}, AP sends $(Cert_{sta}, Cert_{ap}, t_{sta}, \sigma_{ap})$ to ASU, where $\sigma_{ap} = \mathsf{Sign}(Cert_{sta}||Cert_{ap}||t_{sta}, sk_{ap,s})$.
4. After receiving $(Cert_{sta}, Cert_{ap}, t_{sta}, \sigma_{ap})$, ASU first checks that

   $$\mathsf{Verify}(Cert_{sta}||Cert_{ap}||t_{sta}, pk_{ap,s}, \sigma_{ap}) = 1$$

   If the check succeeds, ASU sends $(R_{sta}, R_{ap}, \sigma_{asu})$ to AP, where R_{sta} is the validation result on $Cert_{sta}$, R_{ap} is the validation result on $Cert_{ap}$, and $\sigma_{asu} = \mathsf{Sign}(R_{sta}||R_{ap}, sk_{asu})$.

5. After receiving $(R_{sta}, R_{ap}, \sigma_{asu})$, AP first checks the validation result. If $Cert_{sta}$ is valid, AP forwards $(R_{sta}, R_{ap}, \sigma_{asu})$ to STA, which checks the validation results R_{sta} and R_{ap}, and the signature σ_{asu}.

6. AP selects $r_1 \in_R \{0,1\}^\ell$ and sends (SPI, c_1, σ_1) to STA, where SPI is a concatenation of AP's MAC address, STA's MAC address, and a timestamp, $c_1 = \mathsf{Enc}(r_1, pk_{sta})$, and $\sigma_1 = \mathsf{Sign}(SPI||c_1, sk_{ap,s})$.

7. After receiving (SPI, c_1, σ_1), STA sends (SPI, c_2, σ_2) to AP, where $r_1 = \mathsf{Dec}(c_1, sk_{sta})$, $r_2 \in_R \{0,1\}^\ell$, $k_1||k_2 = \mathsf{H_1}(r_1 \oplus r_2)$[1], $c_2 = \mathsf{Enc}(r_2, pk_{ap,e})$, and $\sigma_2 = \mathsf{H_2}(k_1, SPI||c_2)$. In addition, STA sets k_1 as the session key.

8. After receiving (c_2, σ_2), AP computes $r_2 = \mathsf{Dec}(c_2, sk_{ap,e})$, computes k_1, k_2 in the same way as STA, and then checks σ_2. If the check succeeds, AP accepts k_1 as the session key; otherwise, AP aborts.

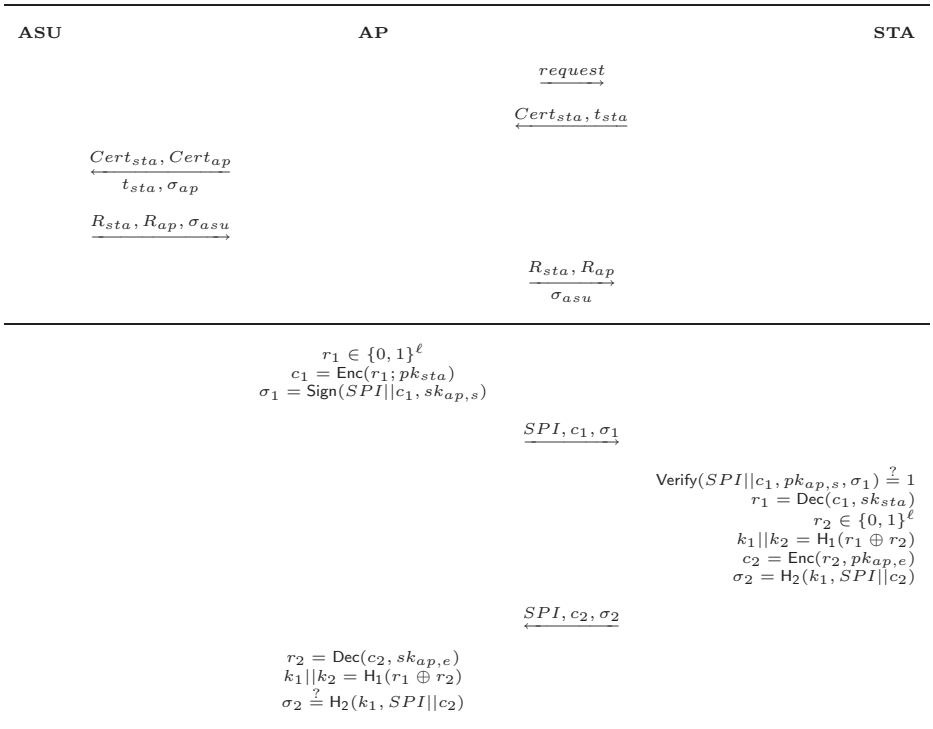

Figure 5.1: The Second Version of the WAI Protocol

The first five steps form the authentication sub-protocol, which enables STA and AP to authenticate each other. The final three steps form the key establishment sub-protocol, which enables STA and AP to establish a session key.

[1] k_1 is the first half of $\mathsf{H_1}(r_1 \oplus r_2)$, and k_2 is the second half.

5.3.2 Some security vulnerabilities

We show that, the WAI protocol is insecure in the Canetti-Krawczyk model, i.e., an inside attacker can obtain the session key in any target session by issuing session-state-reveal queries in other sessions, and corrupting one access point and one mobile station.

Suppose that, in step 6 of a session identified by SPI, AP sends $c_1 = \mathsf{Enc}(r_1, pk_{sta})$, and, $\sigma_1 = \mathsf{Sign}(SPI||c_1, sk_{ap,s})$ to STA, and in step 7, STA sends $c_2 = \mathsf{Enc}(r_2, pk_{ap,e})$, and $\sigma_2 = \mathsf{H}_2(k_1, SPI||c_2)$ to AP. The attacker mounts the attack in two steps, as follows:

1. The attacker obtains the long-term private keys of some access point AP$'$ and some station STA$'$, where AP $\neq$ AP$'$, STA $\neq$ STA$'$, AP$'$ possesses a key pair $(pk'_{ap,e}, sk'_{ap,e})$ for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a key pair $(pk'_{ap,s}, sk'_{ap,s})$ for $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and STA$'$ possesses a key pair (pk'_{sta}, sk'_{sta}) for $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.
2. In a subsequent session for AP$'$ and STA with identifier SPI' (where the attacker impersonates AP$'$), in step 6 the attacker sends $c'_1 = c_1$ and $\sigma'_1 = \mathsf{Sign}(SPI'||c'_1, sk'_{ap,s})$ to STA. It is straightward to verify that STA will succeed in verifying the attacker's message. The attacker then corrupts STA's session and obtains r_1.
3. In another subsequent session for AP and STA$'$ with identifier SPI'' (where this time the attacker impersonates STA$'$), after receiving $c''_1 = \mathsf{Enc}(r''_1; pk_{ap,e})$ and $\sigma''_1 = \mathsf{Sign}(SPI''||c''_1, sk_{ap,s})$ from AP in step 7, the attacker sends $c''_2 = c_2$ and σ''_2 to AP, where σ''_2 is randomly chosen from the appropriate domain. The attacker then corrupts AP's session and obtains r_2.
4. With r_1 and r_2, the attacker can compute the session key belonging to the session identified by SPI.

It is straightforward to verify that the attacker has played a valid game for session key security in the Canetti-Krawczyk model, which implies that the WAI protocol is not secure in this model.

In addition to the above result, we have the following comments on the WAI protocol.

- As noted by Li, Moon and Ma [169], the protocol does not achieve perfect forward secrecy. This is because, given the private keys of both AP and STA, an attacker can recompute any previous established session keys from intercepted protocol messages.

  However, the situation is actually worse than this. In certain circumstances, as described below, if an inside attacker just has access to the long-term private key of AP and the ephemeral state of STA (i.e. not to the private key of STA), then it is possible to recover past session keys.

  Suppose that, in a session identified by SPI, in step 6, AP sends $c_1 = \mathsf{Enc}(r_1, pk_{sta})$, and $\sigma_1 = \mathsf{Sign}(SPI||c_1, sk_{ap,s})$ to STA, and in step 7, STA sends $c_2 = \mathsf{Enc}(r_2, pk_{ap,e})$, and $\sigma_2 = \mathsf{H}_2(k_1, SPI||c_2)$ to AP. In a subsequent session for AP (impersonated by the attacker) and STA identified by SPI', in step 6, the attacker sends $c_1' = c_1$ and $\sigma_1' = \mathsf{Sign}(SPI'||c_1', sk_{ap,s})$ to STA. It is straightward to verify that STA will succeed in verifying the attacker's message. The attacker then corrupts STA's session and obtains r_1. Since the attacker can decrypt c_2 to obtain r_2, it can then compute the session key k_1.

- Since STA chooses r_2 after seeing r_1, and since the session keys are computed as:

  $$k_1||k_2 = \mathsf{H}_1(r_1 \oplus r_2),$$

  it follows that STA has full control over the input to the hash-function, i.e. STA can arrange for $k_1||k_2$ to be equal to $\mathsf{H}_1(r^*)$ for any value r^* of its choice. Of course, given H_1 is one-way, this does not mean that STA can control all of k_1, but STA can certainly choose some of the bits of k_1 and/or k_2, or make $k_1||k_2$ be the same as previously used values.

5.4 Examples of DPUKS Attacks

In this section we give examples of protocols which suffer from DPUKS attacks as defined in Section 3.3. Informally, in a successful DPUKS attack, dishonest partners make two honest users, who participate in different sessions, compute the same session key. We do not mean to suggest that these protocols are insecure with respect to the security model within which they were originally proposed, because

this type of attack has not been formalised. Note that some of these attacks appear in the work of Chen and Tang [76].

5.4.1 Attack against the DHKE-1 protocol

The DHKE-1 protocol was proved secure in the Shoup model [217] (in both the adaptive corruption mode and the strong adaptive corruption mode).

5.4.1.1 Description of the protocol

The TTP runs the TTP initialisation sub-protocol to generate the following parameters: a family of pair-wise independent hash functions H_k indexed by a bit string k, a group $\mathbb{G}$ of prime order q, a generator g of $\mathbb{G}$, and a pseudo-random function BitGen. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a public/private key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and retrieve a certificate $Cert_i$ for pk_i from the TTP. It is assumed that, $Cert_i$, the certificate for pk_i, contains a description of $\mathbb{G}$ and g.

Suppose U_i and U_j wish to establish a session key. The key establishment sub-protocol is defined as follows.

1. U_i selects $s_i \in_R \mathbb{Z}_q$, and sends $(g^{s_i}, \sigma_i, Cert_i)$ to U_j, where
$$\sigma_i = \mathsf{Sign}(g^{s_i}||ID_j, sk_i).$$
2. U_j selects $s_j \in_R \mathbb{Z}_q$, and sends $(g^{s_j}, k, \sigma_j, Cert_j)$ to U_i, where k is a random hash function index and
$$\sigma_j = \mathsf{Sign}(g^{s_i}||g^{s_j}||k||ID_i, sk_j).$$
3. U_i sends k_1 to U_j, where $(k_1, k_2) = \mathsf{BitGen}(\mathsf{H}_k(g^{s_i s_j}))$, and takes k_2 as the session key.

5.4.1.2 Description of the attack

Suppose that there are two sessions: one for $\{U_1, U_2\}$ and the other for $\{U_2, U_3\}$. Suppose also that U_2 is malicious. U_2 can then mount a DPUKS attack, as shown in Fig. 5.2.

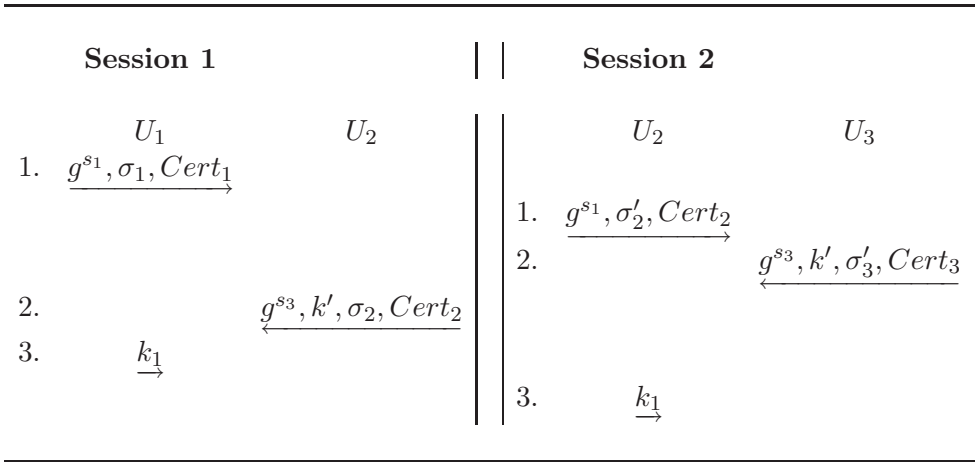

Figure 5.2: DPUKS attack against the DHKE-1 protocol

Note that U_2 sends its first message in the second session after it has received U_1's first message in the first session, it sends its first message in the first session after it has received U_3's first message in the second session, and it sends its second message in the second session after it has received U_1's second message in the first session.

It is straightforward to see that, both sessions end successfully, and U_1 and U_3 compute the same session key k_2. However, while there is no false belief in this attack, U_1 and U_3 share the same key although they are not aware of it.

5.4.2 Attack against the Harn-Lin protocol

The Harn-Lin protocol [116] was shown to suffer from unknown key share attacks in [254], and a modified version was also proposed. We show that the Harn-Lin protocol also suffers from two different DPUKS attacks. It is straightforward to verify that the modified version given in [254] is also vulnerable to these attacks.

5.4.2.1 Description of the protocol

The TTP runs the TTP initialisation sub-protocol to generate the following parameters: a group $\mathbb{G}$ of prime order q and a generator g of $\mathbb{G}$. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a public/private key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and retrieve a certificate $Cert_i$ for pk_i from the TTP.

Suppose U_i and U_j wish to establish a session key; the key establishment sub-protocol is as follows.

1. U_i selects $s_{i1}, s_{i2} \in_R \mathbb{Z}_p$ and sends $(Cert_i, r_{i1}, r_{i2}, \sigma_i)$ to U_j, where $r_{i1} = g^{s_{i1}}$, $r_{i2} = g^{s_{i2}}$, and $\sigma_i = \mathsf{Sign}(r_{i1}||r_{i2}, sk_i)$.
2. U_j selects $s_{j1}, s_{j2} \in_R \mathbb{Z}_p$ and sends $(Cert_j, r_{j1}, r_{j2}, \sigma_j)$ to U_i, where $r_{j1} = g^{s_{j1}}$, $r_{j2} = g^{s_{j2}}$, and $\sigma_j = \mathsf{Sign}(r_{j1}||r_{j2}, sk_j)$.
3. If $\mathsf{Verify}(r_{j1}||r_{j2}, pk_j, \sigma_j) = 1$, U_i computes the shared session keys K_ℓ ($1 \leq \ell \leq 4$), where

   $$K_1 = (r_{j1})^{s_{i1}}, K_2 = (r_{j1})^{s_{i2}}, K_3 = (r_{j2})^{s_{i1}}, K_4 = (r_{j2})^{s_{i2}};$$

   otherwise, U_1 aborts the protocol execution.
4. If $\mathsf{Verify}(r_{i1}||r_{i2}, pk_i, \sigma_i) = 1$, U_j computes the shared session keys K_ℓ ($1 \leq \ell \leq 4$), where

   $$K_1 = (r_{i1})^{s_{j1}}, K_2 = (r_{i1})^{s_{j2}}, K_3 = (r_{i2})^{s_{j1}}, K_4 = (r_{i2})^{s_{j2}};$$

   otherwise, U_j aborts the protocol execution.

5.4.2.2 A DPUKS attack

Suppose that there are two sessions, one for $\{U_i, U_j\}$ and the other for $\{U_j, U_k\}$. Suppose also that U_j is malicious; U_j can then mount a DPUKS attack, as shown in Fig. 5.3.

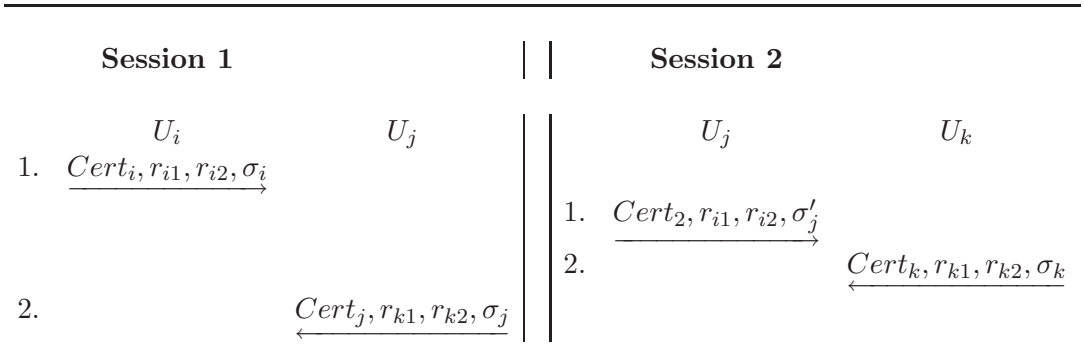

Figure 5.3: One DPUKS attack against the Harn-Lin protocol

Note that U_j sends its message in the second session after it has received U_i's message in the first session, and sends its message in the first session after it has received U_k's message in the second session.

It is straightforward to see that both sessions end successfully, and U_i and U_k compute the same session keys K_ℓ ($1 \leq \ell \leq 4$). However, while there is no false belief in this attack, U_i and U_k share the same key although they are not aware of it.

5.4.2.3 A second attack

Suppose that there are two sessions, one for $\{U_i, U_j\}$ and the other for $\{U_j[U_i], U_k\}$, where the notation $U_j[U_i]$ means that U_j is malicious and impersonating U_i. The DPUKS attack is shown in Fig. 5.4.

Note that U_j sends its message in the first session after it has received U_k's message in the second session, and U_j sends its message in the second session after it has received U_i's message in the first session.

It is straightforward to see that both sessions end successfully, and U_i and U_k compute the same session keys K_ℓ ($1 \leq \ell \leq 4$). In this attack, there is a false belief: U_k mistakenly thinks its keys are shared with U_i; however, U_i thinks its keys are shared with U_j. In fact, this is also an attack against the key authentication and the entity authentication properties.

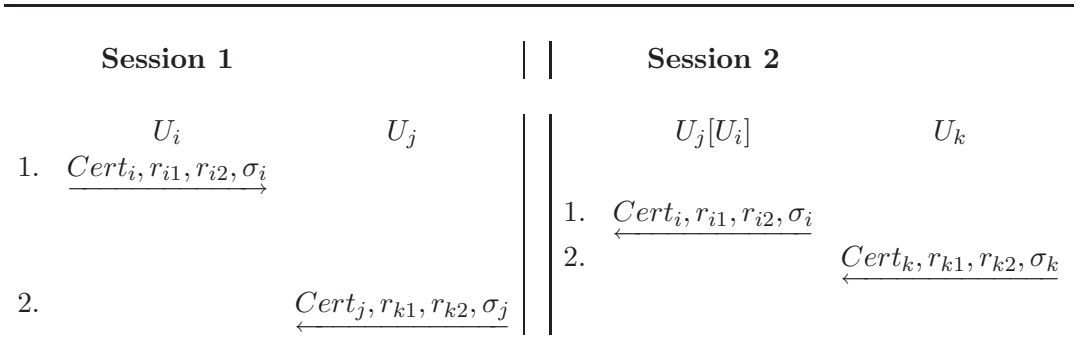

Figure 5.4: The other DPUKS attack against the Harn-Lin protocol

5.4.3 Attack against the extended Joux's protocol

The protocol (described below) of Hitchcock, Boyd, and Nieto [119] has been proved secure in Canetti-Krawcyzk model.

5.4.3.1 Description of the protocol

The TTP runs the TTP initialisation sub-protocol to generate a cyclic group $\mathbb{G}_1$ of prime order q, a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$, a generator P of $\mathbb{G}_1$, and a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a verify/sign key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{sign}, \mathsf{Verify})$, and get pk_i certified by the TTP.

Suppose U_i ($1 \leq i \leq 3$) wish to establish a session key in a session identified by sid; then the key establishment sub-protocol is defined as follows.

1. U_1 selects $s_1 \in_R \mathbb{Z}_q$, and broadcasts $(sid, s_1 P)$.
2. U_2 selects $s_2 \in_R \mathbb{Z}_q$, and broadcasts $(sid, s_2 P)$.

3. U_3 selects $s_3 \in_R \mathbb{Z}_q$, and broadcasts (sid, s_3P, σ_3), where

$$\sigma_3 = \mathsf{Sign}(sid||ID_1||ID_2||s_1P||s_2P||s_3P, sk_3).$$

4. If $\mathsf{Verify}(sid||ID_1||ID_2||s_1P||s_2P||s_3P, pk_3, \sigma_3) = 1$, U_1 broadcasts (sid, σ_1), where

$$\sigma_1 = \mathsf{Sign}(sid||ID_2||ID_3||s_1P||s_2P||s_3P, sk_1).$$

5. If the following two equations hold,

$$\mathsf{Verify}(sid||ID_1||ID_2||s_1P||s_2P||s_3P, pk_3, \sigma_3) = 1,$$

$$\mathsf{Verify}(sid||ID_2||ID_3||s_1P||s_2P||s_3P, pk_1, \sigma_1) = 1,$$

   U_2 broadcasts (sid, σ_2), where

$$\sigma_2 = \mathsf{Sign}(sid||ID_1||ID_3||s_1P||s_2P||s_3P, sk_2).$$

6. If $\mathsf{Verify}(sid||ID_1||ID_3||s_1P||s_2P||s_3P, pk_2, \sigma_2) = 1$, U_1 and U_3 accept.

At the end of the protocol execution, the session key is computed as $K = \hat{e}(P, P)^{s_1 s_2 s_3}$.

5.4.3.2 Description of the attack

Suppose that if there are two sessions, one for $\{U_1, U_2, U_3\}$ and the other for $\{U_2, U_3, U_4\}$. Then U_2 and U_3 can collude to mount an attack, as shown in Fig. 5.5.

Note that U_3 only sends its first message in the first session until it has received U_4's message in the second session. It is straightforward to see that both sessions end successfully, and that U_i $(1 \leq i \leq 4)$ computes the session key as $\hat{e}(P, P)^{s_1 s_2 s_4}$.

5.5 Attacks against Information Leakage Resilience

In this section we give some examples of password-based protocols which suffer from attacks against information leakage resilience. Since none of these protocols requires a TTP, in every case we omit references to a TTP initialisation sub-protocol.

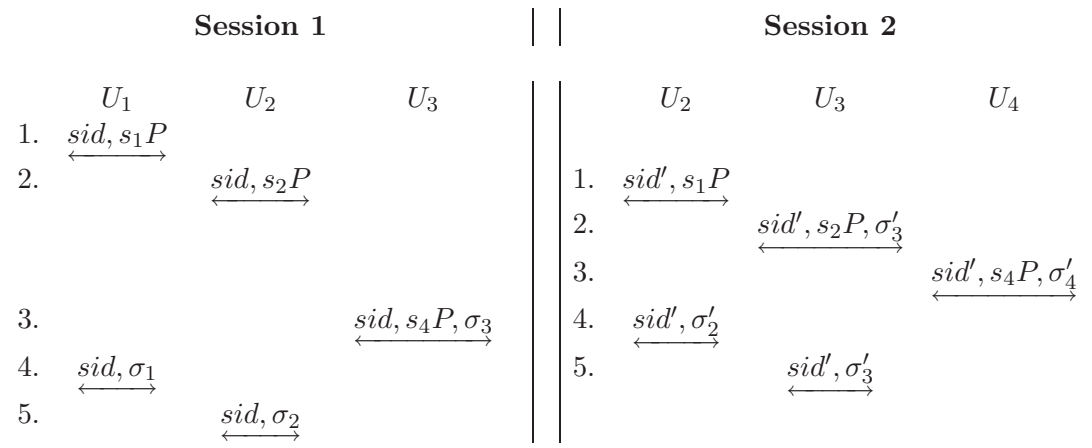

Figure 5.5: DPUKS attack against extended Joux's protocol

5.5.1 Analysis of the Jablon protocol

The Jablon protocol [130], and its variants appear in the standardised password-based key agreement mechanisms specified in [127, 129]

5.5.1.1 Description of the Jablon protocol

The user initialisation sub-protocol generates two prime numbers p and q, where $p = 2q + 1$, and a one-way hash function H. In addition, this sub-protocol also generates a shared password π, which is assumed to be an integer, between a user U with identity ID_U and a server S with identity ID_S.

The key establishment sub-protocol is defined as follows.

1. U generates $t_1 \in_R \mathbb{Z}_q^*$, and sends $m_1 = g^{t_1} \bmod p$ to S, where $g = \pi^2 \bmod p$.
2. After receiving m_1, S generates $t_2 \in_R \mathbb{Z}_q^*$, and sends $m_2 = g^{t_2} \bmod p$ to U. S computes $z = g^{t_2 t_1} \bmod p$, and checks whether $z \geq 2$. If the check succeeds, S uses z as the shared key material, and computes $K = \mathsf{H}(z)$ as the shared key.

3. After receiving m_2, U computes $z = g^{t_2 t_1} \bmod p$, and checks $z \geq 2$. If the check succeeds, U uses z as the shared key material, and computes $K = \mathsf{H}(z)$ as the shared key. Then U constructs and sends the confirmation message $C_1 = \mathsf{H}(\mathsf{H}(\mathsf{H}(z)))$ to S.

   Note that in both the ISO/IEC 11770-4 and IEEE P1363.2 versions of the mechanism, C_1 is instead computed as:

$$C_1 = \mathsf{H}(3||m_1||m_2||g^{t_1 t_2}||g).$$

4. After receiving C_1, S checks that the received message equals $\mathsf{H}(\mathsf{H}(\mathsf{H}(z)))$. If the check fails, S terminates the protocol execution. Otherwise, S computes and sends the confirmation message $C_2 = \mathsf{H}(\mathsf{H}(z))$ to U.

   Note that in both the ISO/IEC 11770-4 and IEEE P1363.2 versions of the mechanism, C_2 is instead computed as:

$$C_2 = \mathsf{H}(4||m_1||m_2||g^{t_1 t_2}||g),$$

5. After receiving C_2, U checks that it equals $\mathsf{H}(\mathsf{H}(z))$. If the check fails, U terminates the protocol execution. Otherwise, U confirms that the protocol execution has ended successfully.

Finally, note that, in the elliptic curve setting, the first password-based key agreement mechanism in [129] and the scheme BPKAS-SPEKE in [127] are essentially the same as above, except that g is a generator of the group of points on an elliptic curve.

5.5.1.2 Security Vulnerabilities of the Jablon Protocol

We describe two security vulnerabilities of the Jablon protocol, and we show that the standardised password-based key establishment mechanisms in [127, 129] also suffer from either one or both of these vulnerabilities. Observe also that the Jablon protocol is not semantically secure, because $C_2 = \mathsf{H}(K)$ is public. Note that these attacks appear in the work of Tang and Mitchell [234].

Firstly, depending on the implementation, the Jablon protocol potentially suffers from an offline dictionary attack in the way that an attacker can try several possible passwords by intervening in only one execution of the protocol.

To mount an attack, the attacker first guesses a possible password π' and replaces the server's message with $m_2' = (\pi')^{2t_2'}$ in the second step of an ongoing protocol instance, where t_2' is chosen by the attacker. The attacker then intercepts the authentication message C_1 in the third step of the same instance and mounts the attack as follows.

1. The attacker sets $i = 1$.
2. The attacker computes $\pi'' = (\pi')^i$, and checks whether π'' falls into the password set. If the check succeeds, go to the third step. Otherwise, stop.
3. The attacker checks whether $C_1 = \mathsf{H}(\mathsf{H}(\mathsf{H}((m_1)^{it_2'})))$. If the check succeeds, the attacker confirms that $\pi = \pi''$. Otherwise, set $i = i + 1$ and go to the second step.

It is straightforward to verify that this attack is valid. We now give a concrete example of how the attack could work in practice. Suppose that the password set contains all binary strings of length at most n, where the password π is made into an integer by treating the string as the binary representation of an integer. Suppose that the attacker guesses a password $\pi' = 2$; then he can try $n - 1$ passwords $(\pi')^i$ $(1 \leq i \leq n - 1)$ by intervening in only one execution of the protocol. Of course, the above version of the attack only works when the initial guessed password π' satisfies $\pi' < 2^{n/2}$.

Secondly, we show that a security vulnerability exists when one entity shares the same password with at least two other entities. This is likely to occur when a human user chooses passwords that it shares with a multiplicity of servers. Specifically we suppose that a client, say U with identity ID_U, shares a password π with two different servers, say S_1 with identity ID_{S_1} and S_2 with identity ID_{S_2}. A malicious third party can mount the attack as follows.

Suppose U initiates the protocol with an attacker which is impersonating server S_1. Meanwhile the attacker also initiates the protocol with server S_2, impersonating U.

The attacker now forwards all messages sent by U (intended for S_1) to S_2. Also, all messages sent from S_2 to U are forwarded to U as if they come from S_1. At the end of the protocol, U will believe that he/she has authenticated S_1 and has established a secret key with S_1. However S_1 has not exchanged any messages with U. In fact, the secret key will have been established with S_2.

The second attack demonstrates that, even if the server S_1 is absent, the attacker can make the client believe that the server is present and that they have computed the same session key as each other. Of course, if U shares the same password with servers S_1 and S_2, then S_1 can always impersonate U to S_2 and also S_2 to U, regardless of the protocol design. However, the problem we have described in the Jablon scheme applies even when U, S_1 and S_2 all behave honestly, and this is not a property that is inevitable.

Based on the descriptions in Section 5.5.1.1, it is straightforward to mount the second attack on the first password-based key agreement mechanism in [129]. In fact, the second attack also applies to the other key agreement mechanisms in [129]. However, if the identifier of the server is used in computing g, e.g. if it is included in the string *str*, then this attack will fail. The scheme BPKAS-SPEKE in [127] is thus immune to this attack as long as the recommendation given in [127] to include this identifier in *str* is followed.

5.5.2 Analysis of the Lai-Ding-Huang protocol

We show that the Lai-Ding-Huang protocol [162] suffers from offline password guessing attacks. This implies that the attacker can collect the messages sent during protocol execution and use them as the basis for an exhaustive search for the password, without initiating any new protocol instance. Note that these attacks appear in the work of Tang and Mitchell [232].

5.5.2.1 Description of the Lai-Ding-Huang protocol

The user initialisation sub-protocol generates a block cipher $(\mathsf{Enc}, \mathsf{Dec})$, a one-way hash function $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ where ℓ is the security parameter, a set B_n containing strings of n elements, which are drawn from some set of characters (e.g. all letters or all alphanumeric symbols), and a composite function $\varphi(r,s) = g(p(r,s))$, where g is a distortion function and p is a picture function. Specifically, given inputs r and s, where r is a random string of characters or bits and s is a random number, p generates a random picture which depicts r in some way. Given an input $p(r,s)$ (a picture) the distortion function g generates a distorted version $R' = g(p(r,s))$ such that humans have the ability to recognise r from R' while a machine typically cannot. In addition, this sub-protocol also generates a shared password π, which is assumed to be an integer, between a user U with identity ID_U and a server S with identity ID_S.

The key establishment sub-protocol is as follows.

1. U generates $t \in_R \{0,1\}^\ell$, and sends (ID_U, t) to S.
2. S first generates $s \in_R \{0,1\}^\ell$ and $r \in_R B_n$. Then S computes and sends $C_1 = \mathsf{Enc}(\varphi(r,s),\pi)$ and $C_2 = \mathsf{H}(\pi||r||t)$ to U, where, as throughout, $||$ represents the concatenation operator.
3. U first computes $\mathsf{Dec}(C_1,\pi)$, which should equal a distorted version of an image depicting r. U then recovers r' from the image, and checks whether or not $C_2 = \mathsf{H}(\pi||r'||t)$. If the check succeeds (implying that $r = r'$), U computes and sends $C_3 = \mathsf{H}(1||\pi||r'||t)$ to S. Otherwise U terminates the protocol.
4. S checks whether $C_3 = \mathsf{H}(1||\pi||r||t)$ holds. If the check succeeds, S has confirmed that U is the valid user and is involved in the current protocol execution. Otherwise, S terminates the protocol.

If the protocol ends successfully, then S and U compute their shared session key as $\mathsf{H}(2||\pi||r||t)$.

5.5.2.2 Claims of Laih, Ding and Huang

In their analysis, Laih, Ding and Huang [162] claim that the protocol is secure under the condition $n \log_2 a + \log_2(|C_\pi|) > 70$, where a is the size of the symbol set used to construct B_n, C_π is the set of passwords, and $|C_\pi|$ is the size of the password set. They also recommend that $|C_\pi|$ equals 2^{23}, which can be achieved by choosing $n = 4$ and $a = 62$, as is the case if the symbols are any lower or upper case letter or any digit from 0 – 9. Specifically they make the following two security claims.

1. Exhaustive search by a machine

   The machine first needs to compute ${C_1}' = \mathsf{Enc}(\varphi(r', s'), \pi)$ by guessing the values of r' and π', and then compares ${C_1}'$ and C_1 in order to verify this guess. There are a^n possible values for r', and $|C_\pi|$ possible values for π', i.e. the total search space is of size

   $$a^n |C_\pi| = 2^{\log_2(a^n) + \log_2(|C_\pi|)} > 2^{70}$$

   So, based on the assumption that $n \log_2 a + \log_2(|C_\pi|) > 70$, it is computationally infeasible for the machine to compute π.

2. Exhaustive search by a human being and a machine

   If a valid message $C_1 = \mathsf{Enc}(\varphi(r, s), \pi)$ is obtained, the machine first guesses a password π' and computes $A = \mathsf{Dec}(C_1, \pi')$; then the human being decides whether or not A contains a string from B_n, which indicates whether or not π' equals π. This process is repeated until the correct password is found. This would require the human to check $|C_\pi| = 2^{23}$ possible values for π'. Based on this, Laih, Ding and Huang estimate that in this case it will take about 3.2 months for a human being and a machine to successfully search for the password.

5.5.2.3 Security Vulnerabilities

In the Lai-Ding-Huang protocol, the protection of the password is based on the security of the function φ, i.e., the assumption that a machine (without a human

being involved) cannot effectively recognise r from $\varphi(r, s)$. As Laih, Ding and Huang point out [162], the string recognisation CAPTCHA schemes [12] are potentially suitable choices for the function φ.

We now exhibit a number of security vulnerabilities in the Lai-Ding-Huang protocol which exist almost regardless of the choice of φ. These vulnerabilities are based on the following observations.

1. A human being must be able to easily recognise r from $\mathsf{Dec}(\varphi(r, s), \pi)$, which implies that $\mathsf{Dec}(\varphi(r, s), \pi)$ is very different from a completely random picture.
2. If $\pi' \neq \pi$ then $\mathsf{Dec}(\varphi(r, s), \pi')$ will resemble a random image. This implies that it is possible to determine whether or not a guessed password π' is correct merely by deciding whether $\mathsf{Dec}(C_1, \pi')$ is a (distorted) image or a random pattern.
3. It is likely to be very simple to develop software to distinguish between a distorted image and a random pattern (for example, a compression algorithm should be able to compress an image whereas a random pattern will be incompressible). This is certainly a much simpler problem than automatic string recognition.
4. If humans choose passwords, then they are much more likely to choose some passwords than others; hence if users are free to choose 4-character passwords, then in practice, $|C_\pi|$ will be significantly less than 2^{23}.

Specifically, the following attacks might be mounted by a machine or a human being.

1. In some cases it might be feasible for a machine to mount an offline password guessing attack. The machine works through all possible passwords and, for each guessed password π', the machine computes $A = \mathsf{Dec}(C_1, \pi')$. By some means (see fact 3 above) the machine then checks whether or not A resembles a distorted image rather than a random bit pattern. Because of fact 2 above, the correct password can be identified from the unique case where A is a distorted image rather than a random bit pattern. This attack only requires a machine-based search of size $|C_\pi|$. If, for example, it takes a millisecond to check one

value of A, then checking through a password space of size 2^{23} will take only 2.3 hours.

2. The above attack does not take into account fact 4 above. Hence the process can be made significantly faster by checking the most likely passwords first.

3. Even if the method of distinguishing random from genuine images is not perfect, i.e. the exhaustive search yields a small number of possible candidate values π', then a human can be used to check the remaining candidate values A to eliminate all but the value corresponding to the correct password.

4. Distributed attacks are also possible. It may be possible to deploy a cooperative Internet-based attack, e.g. by distributing the pattern recognition problems to users across the Internet (see, for example, [210]).

The security issues in the Lai-Ding-Huang protocol arise from the fact that the image recognition problem (such as a CAPTCHA scheme [12]) is being used to protect the secrecy of a password. This is not something that appears to have been attempted before, and seems inherently risky. It is probably better to restrict use of such techniques to guaranteeing the presence of a human during protocol execution, rather than to protect the secrecy of passwords or keys.

5.5.3 Analysis of EKE-U and EKE-M

Byun and Lee [66] claim that the EKE-U and EKE-M protocols are both secure against dictionary attacks, and in particular against password guessing attacks mounted by dishonest partners. However, we show that the EKE-U protocol suffers from offline dictionary attacks, and the EKE-M protocol suffers from undetectable online dictionary attacks which can be mounted by any dishonest partner. For simplicity of description, we assume that $n \geq 3$ in the rest of this paper. It is straightforward to verify that our results also apply to the case where $n = 2$. Note that these attacks appear in the work of Tang and Chen [228].

5.5.3.1 Description of the EKE-U protocol

The user initialisation sub-protocol generates a full-domain hash function H [223], two one-way hash functions $\mathsf{H_1}$ and $\mathsf{H_2}$, a multiplicative cyclic group $\mathbb{G}$ of prime order q, a generator g of $\mathbb{G}$, and a symmetric-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. In addition, for every $1 \leq i \leq n$, this sub-protocol also establishes a shared password π_i between U_i with identity ID_i and the server S with identity ID_S.

Suppose that a set of users U_i $(1 \leq i \leq n)$ wish to negotiate a session key over a unicast network with the help of S; the key establishment sub-protocol is as follows.

1. U_1 selects $v_1, x_1 \in_R \mathbb{Z}_q$, and computes $T_1 = \{g^{v_1}, g^{v_1 x_1}\}$. U_1 then sends $M_1 = \mathsf{Enc}(T_1, \pi_1)$ to U_2.
2. After receiving M_1 from U_1, U_2 forwards it to S.
3. After receiving M_1 from U_2, S first decrypts it using the password π_1 to obtain $T_1 = \{g^{v_1}, g^{v_1 x_1}\}$. S then selects $v_2 \in_R \mathbb{Z}_q$, and computes $T_1' = \{g^{v_1 v_2}, g^{v_1 v_2 x_1}\}$. Finally, S sends $M_1' = \mathsf{Enc}(T_1', \pi_2)$ to U_2.
4. After receiving M_1' from S, U_2 first decrypts it using his password π_2 to obtain $T_1' = \{g^{v_1 v_2}, g^{v_1 v_2 x_1}\}$. U_2 then selects $x_2 \in_R \mathbb{Z}_q$, and computes $T_2 = \{g^{v_1 v_2 x_1}, g^{v_1 v_2 x_2}, g^{v_1 v_2 x_1 x_2}\}$. Finally, U_2 sends $M_2 = \mathsf{Enc}(T_2, \pi_2)$ to U_3.
5. Recursively, U_j $(3 \leq j \leq n-1)$ and S perform the following steps.
   (a) After receiving M_{j-1} from U_{j-1}, where
   $$M_{j-1} = \mathsf{Enc}(T_{j-1}, \pi_{j-1}),$$
   $$T_{j-1} = \{g^{V_{j-1} \cdot (X_{j-1}/x_1)}, g^{V_{j-1} \cdot (X_{j-1}/x_2)}, \cdots, g^{V_{j-1} \cdot (X_{j-1}/x_{j-1})}, g^{V_{j-1} \cdot X_{j-1}}\},$$
   $$V_{j-1} = v_1 \cdot v_2 \cdots v_{j-1}, \text{ and } X_{j-1} = x_1 \cdot x_2 \cdots x_{j-1},$$
   U_j forwards it to S.
   (b) After receiving M_{j-1} from U_j, S first decrypts it using the password π_{j-1} to obtain T_{j-1}. S then selects $v_j \in_R \mathbb{Z}_q$, and computes T_{j-1}', where
   $$T_{j-1}' = \{g^{V_j \cdot (X_{j-1}/x_1)}, g^{V_j \cdot (X_{j-1}/x_2)}, \cdots, g^{V_j \cdot (X_{j-1}/x_{j-1})}, g^{V_j \cdot X_{j-1}}\}, \text{ and}$$

$$V_j = v_1 \cdot v_2 \cdots v_j, \text{ and } X_{j-1} = x_1 \cdot x_2 \cdots x_{j-1}.$$

Finally, S sends $M'_{j-1} = \mathsf{Enc}(T'_{j-1}, \pi_j)$ to U_j.

(c) After receiving M'_{j-1} from S, U_j first decrypts it using his password π_j to obtain T'_{j-1}. U_2 then selects $x_j \in_R \mathbb{Z}_q$, and computes T_j as

$$T_j = \{g^{V_j \cdot (X_j/x_1)}, g^{V_j \cdot (X_j/x_2)}, \cdots, g^{V_j \cdot (X_j/x_j)}, g^{V_j \cdot X_j}\},$$

$$V_j = v_1 \cdot v_2 \cdots v_j, \text{ and } X_j = x_1 \cdot x_2 \cdots x_j.$$

Finally, U_j sends $M_j = \mathsf{Enc}(T_j, \pi_j)$ to U_{j+1}.

6. After receiving M_{n-1} from U_{n-1}, U_n forwards it to S.

7. After receiving M_{n-1} from U_n, S first decrypts it using the password π_{n-1} to obtain T_{n-1}. S then selects $v_n \in_R \mathbb{Z}_q$, and computes T'_{n-1}, where

$$T'_{n-1} = \{g^{V_n \cdot (X_{n-1}/x_1)}, g^{V_n \cdot (X_{n-1}/x_2)}, \cdots, g^{V_n \cdot (X_{n-1}/x_{n-1})}, g^{V_n \cdot X_{n-1}}\},$$

$$V_n = v_1 \cdot v_2 \cdots v_n, \; X_{n-1} = x_1 \cdot x_2 \cdots x_{n-1}.$$

Finally, S sends $M'_{n-1} = \mathsf{Enc}(T'_{n-1}, \pi_n)$ to U_n.

8. After receiving M'_{n-1} from S, U_n first decrypts it using its password π_n to obtain T'_{n-1}. U_n then selects $x_n \in_R \mathbb{Z}_q$, and computes T_n as

$$T_n = \{g^{V_n \cdot (X_n/x_1)}, g^{V_n \cdot (X_n/x_2)}, \cdots, g^{V_n \cdot (X_n/x_n)}\},$$

$$V_n = v_1 \cdot v_2 \cdots v_n, \text{ and } X_n = x_1 \cdot x_2 \cdots x_n.$$

Finally, U_n sends $M_n = \mathsf{Enc}(T_n, \pi_n)$ to S.

It should be noted that T_n is computed differently from T_j $(1 \leq j \leq n-1)$, in order to prevent S from computing the ultimate session key.

9. After receiving M_n from U_n, S first decrypts it using the password π_n to obtain T_n. S then selects $v_{n+1} \in_R \mathbb{Z}_q$, and computes and sends $E_i = \mathsf{Enc}(g^{V_{n+1} \cdot (X_n/x_i)}, \pi_i)$ to U_i $(1 \leq i \leq n)$, where

$$V_{n+1} = v_1 \cdot v_2 \cdots v_{n+1}, \; X_n = x_1 \cdot x_2 \cdots x_n.$$

10. After receiving E_i from S, U_i decrypts it using its password π_i to obtain $g^{V_{n+1} \cdot (X_n/x_i)}$, and then computes the key material and session key as $K^* =$

$(g^{V_{n+1}\cdot(X_n/x_i)})^{x_i}$ and $K_i = \mathsf{H}(clients||K^*)$, where *clients* is the concatenation of the identifiers of U_i $(1 \leq i \leq n)$.

If key confirmation is required, then U_i computes and broadcasts $Auth_i = \mathsf{H}(i||K_i)$.

11. After receiving every $Auth_j$ $(1 \leq j \leq n, j \neq i)$, U_i checks whether it equals $\mathsf{H}(j||K_i)$. If all the checks succeed, U_i confirms that the protocol has succeeded. Otherwise, U_i terminates the protocol as a failure.

5.5.3.2 A security vulnerability in EKE-U

In the EKE-U protocol, any malicious user U_j, for any $1 \leq j \leq n-1$, can mount offline dictionary attacks against U_{j+1}.

To mount the attack, U_j selects t_1 and t_2, and then sends M_j^* to U_{j+1} instead of M_j, where

$$M_j^* = \mathsf{Enc}(T_j^*, \pi_j),$$
$$T_j^* = \{g^{t_1}, g^{t_1 t_2}, g^{V_j\cdot(X_j/x_3)}, \cdots, g^{V_j\cdot(X_j/x_j)}, g^{V_j\cdot X_j}\},$$
$$V_j = v_1 \cdot v_2 \cdots v_j, \text{ and } X_j = x_1 \cdot x_2 \cdots x_j.$$

After receiving M_j^*, U_{j+1} will forward it to S. We now have the following result.

Lemma 3 *As a result of the above attack, U_j can mount an offline dictionary attack against U_{j+1}.*

Proof. After receiving M_j^* from U_{j+1}, S first decrypts it using the password π_j to obtain T_j^*. S then selects a random number v_{j+1} $(1 \leq v_{j+1} \leq q-1)$, and computes T_j', where

$$T_j' = \{g^{t_1 v_{j+1}}, g^{t_1 t_2 v_{j+1}}, g^{V_{j+1}\cdot(X_j/x_3)}, \cdots, g^{V_{j+1}\cdot(X_j/x_j)}, g^{V_{j+1}\cdot X_j}\},$$

$$V_{j+1} = v_1 \cdot v_2 \cdots v_{j+1}, \text{ and } X_j = x_1 \cdot x_2 \cdots x_j.$$

Finally, S sends $M_j' = \mathsf{Enc}(T_j', \pi_{j+1})$ to U_{j+1}.

U_i then intercepts M_j', and mounts an offline dictionary attack as follows.

1. U_i guesses a possible password π^*_{j+1}, and decrypts M'_j as
$$\mathsf{Dec}(M'_j, \pi^*_{j+1}) = \{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_{j+1}\}$$
2. U_i checks that $(\alpha_1)^{t_2} = \alpha_2$. If the check succeeds, then U_i confirms that $\pi^*_{j+1} = \pi_{j+1}$ because $(\mathsf{Enc}, \mathsf{Dec})$ is a block cipher. Otherwise, go to step 1.

The lemma follows. □

5.5.3.3 Description of the EKE-M protocol

The user initialisation sub-protocol is identical to that of the EKE-U protocol. Suppose that a set of users U_i $(1 \leq i \leq n)$ wish to negotiate a session key over a multicast network with the help of S, the key establishment sub-protocol is as follows.

1. For every $1 \leq i \leq n-1$, S selects $s_i \in_R \mathbb{Z}_q$ and then sends $\mathsf{Enc}(g^{s_i}, \pi_i)$ to U_i. Concurrently, U_i selects $x_i \in_R \mathbb{Z}_q$, and then broadcasts $\mathsf{Enc}(g^{x_i}, \pi_i)$.
2. After receiving every $\mathsf{Enc}(g^{x_i}, \pi_i)$ $(1 \leq i \leq n-1)$, S decrypts each of them to obtain g^{x_i}. S then computes the shared ephemeral key with U_i as $sk_i = \mathsf{H}_1(sid'||g^{x_i s_i})$, where
$$sid' = \mathsf{Enc}(g^{x_1}, \pi_1)||\mathsf{Enc}(g^{x_2}, \pi_2)||\cdots||\mathsf{Enc}(g^{x_{n-1}}, \pi_{n-1})$$
Finally, S selects a random secret N, and broadcasts $m_i = N \oplus sk_i$.
3. After receiving all the messages from S, U_i first constructs sid' in the same way as S, decrypts $\mathsf{Enc}(g^{s_i}, \pi_i)$, computes $sk_i = \mathsf{H}_1(sid'||g^{s_i x_i})$, and then computes $N = m_i \oplus sk_i$. Finally, U_i computes the session key as $K_i = \mathsf{H}_2(SIDS||N)$, where
$$SIDS = sid'||sk_1 \oplus N||sk_2 \oplus N||\cdots||sk_{n-1} \oplus N$$
If key confirmation is required, then U_i computes and broadcasts $Auth_i = \mathsf{H}(i||K_i)$.
4. After receiving every $Auth_j$ $(1 \leq j \leq n-1, j \neq i)$, U_i checks whether it equals $\mathsf{H}(j||K_i)$. If all the checks succeed, U_i confirms that the protocol has succeeded. Otherwise, U_i terminates the protocol as a failure.

5.5.3.4 A security vulnerability in EKE-M

In EKE-M, a malicious participant U_j, for any $1 \leq j \leq n-1$, can mount an online dictionary attack against any other participant U_i $(1 \leq i \leq n-1, i \neq j)$ without being detected by any entity.

To mount the attack, U_j initiates an instance of the protocol, and blocks all messages sent to U_i. In the first step, U_j guesses a possible password π_i^* possessed by U_i, and impersonates U_i to broadcast $\mathsf{Enc}(g^{x_i}, \pi_i^*)$. In the third step, U_j impersonates U_i to broadcast the key confirmation message $Auth_i = \mathsf{H}(i||K_i)$. We now have the following result.

Lemma 4 *As a result of the above attack, U_j can test whether $\pi_i^* = \pi_i$, the protocol instance will end successfully, and all participants except U_j compute the same session key.*

Proof. In the EKE-M protocol, the session key material N is independently sent to each participant and the session key is a function of N and other information. It is thus straightforward to verify that the protocol will successfully end and all participants except U_i will compute the same session key.

After intercepting $\mathsf{Enc}(g^{s_i}, \pi_i)$ and $m_i = sk_i \oplus N$ sent by S, U_j first computes the guessed ephemeral session key between U_i and S as

$$sk_i^* = \mathsf{H}_1(sid'||(D(\mathsf{Enc}(g^{s_i}, \pi_i)), \pi_i^*)^{x_i})$$

U_j then checks whether $N = m_i \oplus sk_i^*$. Based on the properties of the block cipher $(\mathsf{Enc}, \mathsf{Dec})$, if the check succeeds then U_j can confirm that $\pi_i^* = \pi_i$; otherwise $\pi_i^* \neq \pi_i$. □

5.5.4 Analysis of RSA-AKE protocol

The RSA-AKE protocol, proposed by Shin, Kobara and Imai [216], is designed to be used in an client-server environment, where the following conditions hold.

1. A client C, who can only remember a password, wishes to communicate with several servers.
2. C has insecure devices with very restricted computing power and built-in memory capacity. The servers have significant computing power, but they might be compromised.
3. Neither a PKI (Public Key Infrastructure) nor a TRM (Tamper-Resistant Module) is available.

5.5.4.1 Description of the RSA-AKE protocol

C possesses identity ID_C, and S possesses identity ID_S. The user initialisation sub-protocol generates a full-domain hash function H, and four different hash functions $\mathsf{H_i}$ $(1 \leq i \leq 4) : \{0,1\}^* \rightarrow \{0,1\}^\ell$. In addition, this sub-protocol also enables S and U to generate the following parameters: S generates its RSA public/private key pair (e, N) and (d, N), and sends (e, N) to C. C registers a password verifier $p_1 = (\alpha_1 + \pi) \bmod N$ at S, where α_1 is randomly selected from $\mathbb{Z}_N$. C stores α_1 and (e, N) on some insecure device such as a PDA, which is not necessarily securely protected and may leak the stored information. S stores p_1 and (d, N) in its database, which is also not necessarily securely protected and may leak the stored information (both p_j and (d, N)). Note that the values of p_j, $j = 1, 2, \cdots$ are defined recursively — see step 4 below. Finally, C and S both also store a counter j, initially set to 1.

In the j-th ($j \geq 1$) execution of the key establishment sub-protocol, C and S perform as follows.

1. C first computes the password verifier $p_j = \alpha_j + \pi \bmod N$. Note that the values of α_j, $j = 1, 2, \cdots$ are defined recursively — see step 3 below. Then C chooses a random $x \in_R \mathbb{Z}_N^*$ and computes $W = \mathsf{H}(j, p_j)$, $y = x^e \bmod N$, and $z = y \cdot W \bmod N$. Finally, C sends ID_C, j, z to S.
2. S first checks whether j is the correct counter value. If the check succeeds, S computes $y' = z \cdot W^{-1} \bmod N$, $x' = (y')^d \bmod N$, and $V_S = \mathsf{H_1}(ID_C, ID_S, j, z, p_j, x')$, and then sends ID_S, V_S to C. Otherwise, S terminates the protocol execution.

3. After receiving S and V_S, C first checks whether the following equation is valid:

$$V_S = \mathsf{H}_1(ID_C, ID_S, j, z, p_j, x)$$

If the check succeeds, C computes and sends V_C to S, where

$$V_C = \mathsf{H}_2(ID_C, ID_S, j, z, p_j, x)$$

Otherwise, C terminates the protocol execution.

C computes the session key as $SK_j = \mathsf{H}_3(ID_C, ID_S, j, z, p_j, x)$, and replaces the stored data α_j with α_{j+1}:

$$\alpha_{j+1} = \alpha_j + \mathsf{H}_4(ID_C, ID_S, j, z, p_j, x) \bmod N$$

C sets the counter value to $j + 1$.

4. After receiving V_C, S first checks whether the following equation is valid:

$$V_C = \mathsf{H}_2(ID_C, ID_S, j, z, p_j, x')$$

If the check succeeds, S computes the session key as

$$SK_j = \mathsf{H}_3(ID_C, ID_S, j, z, p_j, x'),$$

and replaces the password verifier p_j with p_{j+1}:

$$p_{j+1} = p_j + \mathsf{H}_4(ID_C, ID_S, j, z, p_j, x') \bmod N$$

S sets the counter value to $j + 1$. Otherwise, S terminates the protocol execution as a failure.

5.5.4.2 Possible weaknesses in RSA-AKE

Shin, Kobara and Imai [216] claim that the RSA-AKE protocol is provably secure in the random oracle model under the notion of LR-AKE security, where the attacker is given the client's stored secret and the server's RSA private key.

However, we nevertheless show that the RSA-AKE protocol suffers from certain potential security problems. It is, however, important to note that some of these vulnerabilities are outside the scope of the security model used in [216]. Note that these vulnerabilities appear in the work of Tang and Mitchell [236].

1. We show that, given the client's stored secret and the server's RSA private key, an attacker can mount an offline dictionary attack. Without loss of generality, we suppose that the attacker is given α_j and (d, N) just before the j-th run of the RSA-EKE protocol.

   During the j-th run of the RSA-EKE protocol, the attacker collects j, z, and V_S. The attacker then performs the following steps:

   (a) The attacker guesses a possible password π^*, computes $p_j^* = \pi^* + \alpha_j \bmod N$, $x^* = ((\mathsf{H}(j, p_j^*))^{-1} \cdot z)^d \bmod N$, and tests whether $V_S = \mathsf{H_1}(ID_C, ID_S, j, z, p_j^*, x^*)$ holds.

   (b) If the test succeeds, the attacker confirms that $\pi^* = \pi$ and stops; otherwise go to the first step.

   It is straightforward to verify that the above attack will succeed in identifying the correct password with very high probability.

2. Observe that p_j is the only secret used for authentication in the j-th run of the RSA-AKE protocol. So, if the attacker has compromised S and obtained p_j, then he can successfully impersonate C to S in the subsequent protocol executions without the need to have access to π. If this occurs, the legitimate client will no longer be able to authenticate itself, because the password verifier held by S will change. However, if the legitimate client authenticates itself before the attacker uses the stolen p_j, then the attacker cannot launch the above attack because the stolen password verifier p_j will no longer be valid.

   This attack means that leakage of p_j from S may enable an attacker to mount an impersonation attack. Hence the RSA-AKE protocol does not appear to be suitable for use in environments where the server is not securely protected.

3. As stated above, the protocol is designed under the assumption that the user's device is not tamper resistant. Shin, Kobara and Imai point out that measures should be adopted to restrict an attacker's ability to replace the RSA public key (e, N) on the client's device; otherwise they show that an eth-residue attack can be mounted. They also propose a means to thwart the e-th residue attack if the attacker does succeed in replacing the RSA public key (e, N) with (e', N'). However, we show below that, in some extreme circumstances, more serious vulnerabilities exist in this case.

Suppose, for example, that the attacker has obtained α_j and replaced the RSA public key (e, N) with (e', N'), where $e' = \phi(N')$, just before the j-th execution of the RSA-AKE protocol. In this case, the attacker can exhaustively search for the password using the intercepted message z. This is because $x^{e'} \bmod N' = 1$ for every x (since $e' = \phi(N')$), and hence $z = \mathsf{H}(j, \pi + \alpha_j) \bmod N'$. That is, the only unknown value used to compute z is π. The measures proposed in [216] do not eliminate this vulnerability.

4. Shin, Kobara and Imai suggest that C can use the same password π with a number of servers. However, it is potentially dangerous to do this. Suppose the client shares the same password with m servers S_i $(1 \leq i \leq m)$. Then an attacker can successfully guess the password with a probability p by mounting n/m dictionary attacks in parallel at each server S_i $(1 \leq i \leq m)$, assuming that it would need to mount n dictionary attacks against one specific server in order to achieve the same goal. This attack means that the client might need to change his password much more frequently (if m is very large) in order to prevent undetected dictionary attacks.

   This attack is of particular concern in environments where m is large and the client chooses the password from a small password set, e.g. based on personal preferences.

5.6 Some Concluding Remarks

In this chapter we have described a number of attacks against existing schemes. Some of these attacks, namely those in Section 5.2.1, 5.2.2, 5.3, 5.5.1, 5.5.2, and 5.5.3 are due to the lack of a rigorous security analysis. The attacks in Section 5.4 arise because the underlying security models do not cover all the security properties that might be required in practice.

Chapter 6

A Novel Security Model for Key Establishment Protocols

Contents

In this chapter we propose a novel, unified, indistinguishability-based security model for key establishment protocols, and present a number of associated security definitions. We also give a comparison between the novel model and existing security models.

6.1 Motivation

In Chapter 4, we showed that none of the existing security models is capable of capturing the full list of security properties, as described in Section 3.3. The other major shortcoming of existing models is that the notion of partnership has been defined too stringently in these models. In this chapter, we aim to provide a fine-grained indistinguishability-based security model, which models all the security properties given in Section 3.3, and also mitigates the shortcomings of the existing models.

The main motivation for employing an indistinguishability-based approach is that we can model the threats to each security property individually.

The rest of this chapter is organised as follows. In Section 6.2 we describe the novel security model for key establishment protocols. In Section 6.3 we provide security definitions for key establishment protocols, and compare them with those used in existing models. In the final section we conclude this chapter.

6.2 A Novel Unified Security Model

6.2.1 Preliminary assumptions and definitions

Given any n-party ($n \geq 2$) key establishment protocol, then, without loss of generality, let the potential users be labelled as U_i ($i \geq 1$), and their (unique) identities be ID_i, respectively. We assume a very general execution environment for the protocol, where any n different users can run the protocol, multiple instances of the protocol can be run concurrently, and the communication is over an open network.

If a user starts a session of the key establishment protocol, we assume that it knows the identities of the partners for this session, and the session is partially determined by the identity information of all the involved users; we refer to the set of identifiers for the n participants in a session as the participant set, abbreviated to UID. Since the same group of users may run multiple instances of the protocol concurrently, we assume that some (possibly random) information RSID is also available to identify a specific session, together with the identity information UID. Hence we assume throughout that a session identifier takes the form (UID, RSID), and this pair uniquely identifies a session. If a user participates in a successfully-ended session of the protocol, then we also say that the user accepts in this session. As a result of a successfully-ended session, a user obtains a session key together with a (public) session identifier.

The information RSID can be generated in many ways. For example, it can be distributed before protocol execution as required in the Canetti-Krawcyzk model, or it can be formed by the protocol messages as in the Bellare-Rogaway model.

Following the literature, a protocol execution of U_i is defined to be an oracle Π_i^{s,sid_i}, where s means it is the s-th oracle of U_i and sid_i is the session identifier. A threat to a security property is evaluated by an attack game played between a hypothetical challenger $\mathcal{C}$ and an attacker $\mathcal{A}$, where the challenger simulates the protocol executions and the attacker can possibly intervene in the protocol execution through various types of oracle queries. In general, an attacker may have access to the following types of oracle queries.

- create, which, on the input of ID_i and a participant set identifier UID ($ID_i \in$ UID), creates an oracle Π_i^{s,sid_i} for U_i, where s means that this is the s-th oracle of U_i.
- send, which, on the input of an active oracle Π_i^{s,sid_i} (see definition below) and a message m, delivers m to Π_t^{s,sid_i}, and returns an outgoing message m' or a decision to indicate acceptance or rejection of the session.
- reveal, which, on the input of an accepted oracle Π_i^{s,sid_i}, returns the session key held by this oracle.
- corrupt-ttp, which, on the input of the identifier of the TTP, returns the TTP's long-term private key.
- corrupt-user, which, on the input of the identifier ID_i of a user U_i, returns U_i's long-term private key.
- corrupt-state, which, on the input of the identifier for an active oracle Π_i^{s,sid_i}, returns the ephemeral internal state of this oracle.
- test, which, on the input of the identifier sid_i of a fresh oracle (see the definition below), returns a string which is computed as follows: choose a random bit b from the set $\{0, 1\}$, return the session key if $b = 1$, or otherwise return a random string from the session key space.

During its life cycle of an oracle, Π_i^{s,sid_i} may be in one (and only one) of the following states:

- Active: the oracle has been created by a create query, but has not ended successfully or unsuccessfully (it is still waiting for inputs from other oracles).

- Accepted: the oracle has ended successfully with a session key and a session identifier, and all ephemeral internal state has been erased.
- Aborted: the oracle has stopped as a failure, and all ephemeral internal state has been erased.

As in other security models, the concept of partnership and freshness are very important for the security formalisation. The partnership of oracles is formally defined as follows.

Definition 25 *Two oracles* Π_i^{s,sid_i} *and* Π_j^{t,sid_j}*, for any* $i \neq j$*, are said to be partners if and only if* $sid_i = sid_j$*.*

Definition 26 *An oracle* Π_i^{s,sid_i} *is fresh if it satisfies the following requirements:*

1. *It has accepted and has not been issued a* reveal *query.*
2. *No partner oracle of* Π_i^{s,sid_i} *has been issued a* reveal *query.*
3. *No user* U_j*, where* $ID_j \in UID$*, has been issued a* corrupt-user *query before* Π_i^{s,sid_i} *accepts.*
4. *Neither* Π_i^{s,sid_i} *nor its partner oracle has been issued a* corrupt-state *query.*

In the above definition, if the attacker is allowed to corrupt U_i through a corrupt-user query, then the oracle Π_i^{s,sid_i} is said to be semi-fresh.

As for other types of cryptographic protocols, we regard soundness as a pre-requisite for any useful key establishment protocol. Formally, soundness is defined as follows:

Definition 27 *A key establishment protocol is defined to be sound if, when the protocol is run in the possible presence of a passive inside attacker (and all messages are successfully delivered without error), all oracles involved in a session accept and output the same session key and session identifier, which is unique for every user.*

6.2.2 Formulations of the Security Properties

Without loss of generality, we formalise the security properties by assuming that the protocol is executed by a group of n ($n \geq 2$) users, U_i ($1 \leq i \leq n$) say, to negotiate a session key. The participant set is thus UID $= (ID_1, ID_2, \cdots, ID_n)$. With respect to the definition in Section 3.2.1, if an attacker has no access to the private information of any of the principals U_i ($i \geq 1$) and the TTP, then it is an outside attacker; if it has no access to the private information of any of the principals U_i ($1 \leq i \leq n$), then it is an inside attacker, and if the attacker has access to the private information of any user U_i, for some $1 \leq i \leq n$, then it is a dishonest partner.

All the properties are formalised as a game between a hypothetical challenger $\mathcal{C}$ and an attacker $\mathcal{A}$. Note that known-key resilience has been included in every attack game, so that we do not define a specific attack game for it.

6.2.2.1 Modelling Key Randomness

For key randomness, we wish to guarantee that the session key in any session is uniformly distributed in the key space, if the attacker is a passive outsider. Formally, the attack game for key randomness is as follows:

1. Setup: The challenger $\mathcal{C}$ generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Phase 1: The attacker $\mathcal{A}$ executes with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting a fresh oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, and reveal. We further require that the attacker is passive, i.e., the attacker is not allowed to manipulate the delivered message through send queries.

The attacker wins if the distribution of the session key is not uniformly distributed in the key space.

6.2.2.2 Modelling Key Control

For key control, we wish to guarantee that no proper subset of the intended users in a session should be able to force the session key to be equal to a pre-defined value. Formally, the attack game for key control is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. Before making any other oracle query, $\mathcal{A}$ selects a string sk^*. At some point, $\mathcal{A}$ terminates by outputting sk^* and an accepted oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and corrupt-ttp.

The attacker's advantage is defined to be $\Pr[sk = sk^*]$, where sk is the session key held by Π_j^{t,sid_j}.

6.2.2.3 Modelling Key Authentication

For key authentication, we wish to guarantee that an entity that is not involved in a session can learn nothing about the session key. Since properties such as (perfect) forward secrecy and unknown key share resilience are also relevant to the key authentication property, we model these properties in a single game. Formally, the attack game for key authentication is as follows, where the attacker has two stages, i.e., $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Phase 1: The first stage attacker $\mathcal{A}_1$ executes with input $param_1$. $\mathcal{A}_1$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and corrupt-ttp. At some point, $\mathcal{A}_1$ terminates by issuing a test query to a fresh

oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$, and outputting some state information *state*.

3. Challenge: The challenger returns the output of $\mathsf{test}(\Pi_j^{t,sid_j})$.

4. Phase 2: The second stage attacker $\mathcal{A}_2$ executes with input *state* and the output of the challenger. $\mathcal{A}_2$ can make the same types of query as $\mathcal{A}_1$. However, the attacker is not permitted to issue a corrupt-state query to any partner oracle of Π_j^{t,sid_j}, and to issue a reveal query to Π_j^{t,sid_j} and its partner oracle. $\mathcal{A}_2$ terminates by outputting a guess bit b'.

The attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Note that it is important to forbid the attacker to issue a corrupt-state query to any partner oracle of Π_j^{t,sid_j}, because, when Π_j^{t,sid_j} accepts, its partner oracles may still be active. Without this requirement, the attacker may trivially win the game by issuing this type of query.

In this game, (perfect) forward secrecy is modelled because the attacker is allowed to corrupt all users in the session identified by sid_j, and unknown key-share resilience is modelled because the attacker is allowed to issue reveal queries.

6.2.2.4 Modelling Backward Secrecy

If the backward secrecy property holds, a corrupted user can be assured that the attacker can learn nothing about its session key in a session after corruption. Formally, the attack game for backward secrecy is as follows, where the attacker has two stages, i.e., $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.

2. Phase 1: The first stage attacker $\mathcal{A}_1$ executes with input $param_1$. $\mathcal{A}_1$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and

corrupt-ttp. At some point, $\mathcal{A}_1$ terminates by issuing a test query with a semi-fresh oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$, and outputting some state information *state*.

3. Challenge: The challenger returns the output of $\mathsf{test}(\Pi_j^{t,sid_j})$.

4. Phase 2: The second stage attacker $\mathcal{A}_2$ executes with input *state* and the output of the challenger. $\mathcal{A}_2$ can make the same types of query as $\mathcal{A}_1$. However, the attacker is not permitted to issue a corrupt-state query to any partner oracle of Π_j^{t,sid_j}, or to issue a reveal query to Π_j^{t,sid_j} and its partner oracle. $\mathcal{A}_2$ terminates by outputting a guess bit b'.

The attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

6.2.2.5 Modelling DPUKS Resilience

For DPUKS resilience, we wish to guarantee that dishonest partners cannot make two honest users, who participate in different sessions, compute the same session key. Formally, the attack game for DPUKS resilience is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.

2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. $\mathcal{A}$ terminates by outputting two accepted oracles Π_j^{t,sid_j} and Π_x^{y,sid_x}, for some $1 \leq j \leq n$, $t \geq 1$, $y \geq 1$, and $x \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and corrupt-ttp. However, the attacker is not permitted to issue a corrupt-user query to U_i or U_x, or to issue a corrupt-state query to Π_j^{t,sid_j} or Π_x^{y,sid_x}.

The attacker wins if $sid_j \neq sid_x$, and Π_j^{t,sid_j} and Π_x^{y,sid_x} possess the same session key.

6.2.2.6 Modelling Mutual Entity Authentication

For mutual entity authentication, we wish to guarantee that users which have a successfully-ended session can be assured that all uncorrupted intended users have the same session identifier[1]. Since the key compromise impersonation resilience property is also relevant to mutual entity authentication, we model both properties in a single game. Formally, the attack game for mutual entity authentication is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting an accepted oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and corrupt-ttp.

The attacker wins, if for some $1 \leq x \leq n$, where $x \neq j$ and no corrupt-user query has been issued to U_x, there is no oracle Π_x^{y,sid_x} which satisfies $sid_x = sid_j$.

In this game key compromise impersonation resilience property is modelled because the attacker is allowed to corrupt U_j.

6.2.2.7 Modelling Key Confirmation

For key confirmation, we wish to guarantee that any user which has a successfully-ended session, can be assured that all uncorrupted intended users have the same session identifier and session key[2]. Formally, the attack game for key confirmation is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.

[1]Note that it is not guaranteed that these sessions will successfully end.
[2]Note that it is not guaranteed that these sessions will successfully end.

2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting an accepted oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, reveal, corrupt-user, corrupt-state, and corrupt-ttp.

The attacker wins, if for some $1 \leq x \leq n$, where $x \neq j$ and no corrupt-user query has been issued to U_x, there is no oracle Π_x^{y,sid_x} which satisfies $sid_x = sid_j$ and Π_x^{y,sid_x} possesses the same session key as that of Π_j^{t,sid_j}.

6.2.2.8 Modelling Password Guessing Resilience

If a password is involved in the protocol, we assume that the password is chosen from the set $\mathcal{PW} = \{\pi^{(1)}, \pi^{(2)}, \cdots, \pi^{(m)}\}$, where $\pi^{(i)}$ has an associated probability of selection p_i and $p_i \leq p_j$ if $i > j$. It is straightforward to compute that, if an attacker has been permitted to have the results of h guesses regarding the password to be established, then the probability that the attacker can successfully guess the password is at most $\sum_{j=1}^{h+1} p_j$.

In a session of an n-party protocol, an oracle may communicate with the other $n-1$ oracles, so that an attacker may test more than one password by intervening in the input of one oracle although the ideal case is that the attacker can only test one password. Generally, we assume that the security against password guessing attacks is parameterised with a parameter ℓ_1 (the maximum number of passwords the attacker can test by by intervening in the input of an oracle), which is referred to as the resilience parameter against password guessing attacks. Informally, a password-based protocol is said to be secure against password guessing attacks if the attacker's advantage over the password is negligibly larger than $\sum_{j=1}^{\ell_1 n_1+1} p_j$ for $\ell_1 \geq 1$ and $n_1 \geq 1$, where ℓ_1 is the resilience parameter and n_1 is the total number of aborted oracles[3]. It is clear that if a protocol has a smaller value of ℓ_1 then it has a higher level security against password guessing attacks.

Formally, the attack game for modelling password guessing attacks is carried out

[3]It is implied that the attacker's advantage in guessing the password is (computationally) irrelevant to the total number of accepted oracles.

between the challenger and a polynomial-time attacker $\mathcal{A}$ as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and private system parameters $param_2$, where $param_2$ includes a password $\pi \in \mathcal{PW}$.
2. Challenge: The attacker runs $\mathcal{A}$ with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting a guessed password π'. During its execution, $\mathcal{A}$ can make the following types of query: create, send, and reveal.

Suppose that there are n_1 aborted oracles at the end of the game. Then the attacker's advantage over the password in the game is defined to be $\mathcal{F}(\ell_1 n_1, \Pr[\pi = \pi'])$, where the function $\mathcal{F}$ is defined as follows. On the input of an integer a and a value b, $\mathcal{F}(a, b)$ is computed as:

$$\mathcal{F}(a,b) = \begin{cases} 0, & \text{if } b \leq \sum_{j=1}^{a+1} p_j \\ b - \sum_{j=1}^{a+1} p_j, & \text{otherwise} \end{cases}$$

6.2.2.9 Modelling Key Authentication (password-based)

Formally, the attack game for key authentication (password-based) is as follows, where the attacker has two stages, i.e., $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Phase 1: The first stage attacker $\mathcal{A}_1$ executes with input $param_1$. $\mathcal{A}_1$ can make the following types of query: create, send, and reveal. At some point, $\mathcal{A}_1$ terminates by issuing a test query to a fresh oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$, and outputting some state information *state*.
3. Challenge: The challenger returns the output of $\mathsf{test}(\Pi_j^{t,sid_j})$.
4. Phase 2: The second stage attacker $\mathcal{A}_2$ executes with input *state* and the output of the challenger. In addition to the same types of query as $\mathcal{A}_1$, $\mathcal{A}_2$

can also make any type of corrupt queries, namely corrupt-user, corrupt-state, and corrupt-ttp. However, the attacker is not permitted to issue a corrupt-state query to any partner oracle of Π_j^{t,sid_j}, or to issue a reveal query to Π_j^{t,sid_j} or its partner oracle. $\mathcal{A}_2$ terminates by outputting a guess bit b'.

Note that if there are n_1 aborted oracles at the end of the game, then the probability that the attacker knows the correct password is at most $\sum_{j=1}^{\ell_1 n_1+1} p_j$ before issuing the test query. If the attacker knows the correct password, then it can trivially guess the bit; otherwise we wish the attacker can only guess b with a probability negligibly larger than $\frac{1}{2}$.

Therefore, the attacker wins if $b' = b$, and its advantage is defined to be the maximum of $\{0, \Pr[b' = b] - \frac{1+\sum_{j=1}^{\ell_1 n_1+1} p_j}{2}\}$ given that there are n_1 aborted oracles at the end of the game.

In this game, (perfect) forward secrecy is modelled because the attacker is allowed to corrupt all users in the session identified by sid_j, and unknown key-share resilience is modelled because the attacker is allowed to issue reveal queries.

6.2.2.10 Modelling Mutual Entity Authentication (password-based)

Formally, the attack game for mutual entity authentication (password-based) is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting an accepted oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, and reveal.

Note that if there are n_1 aborted oracles at the end of the game, then the probability that the attacker knows the correct password is $\sum_{j=1}^{\ell_1 n_1+1} p_j$. If the attacker knows

the correct password, then it can trivially impersonate a user; otherwise we wish the attacker can only succeed with a negligible probability.

Let E_1 be the event that, for some $1 \leq x \leq n$, where $x \neq j$, there is no oracle Π_x^{y,sid_x} which satisfies $sid_x = sid_j$. The attacker wins if E_1 occurs, and its advantage is defined to be the maximum of $\{0, \Pr[E_1] - \sum_{j=1}^{\ell_1 n_1 + 1} p_j\}$ given that there are n_1 aborted oracles at the end of the game.

6.2.2.11 Modelling Key Confirmation (password-based)

Formally, the attack game for key confirmation (password-based) is as follows:

1. Setup: The challenger generates both the public system parameters $param_1$ and the private system parameters $param_2$.
2. Challenge: The attacker $\mathcal{A}$ executes with input $param_1$. At some point, $\mathcal{A}$ terminates by outputting an accepted oracle Π_j^{t,sid_j}, for some $1 \leq j \leq n$ and $t \geq 1$. During its execution, $\mathcal{A}$ can make the following types of query: create, send, and reveal.

Note that if there are n_1 aborted oracles at the end of the game, then the probability that the attacker knows the correct password is $\sum_{j=1}^{\ell_1 n_1 + 1} p_j$. If the attacker knows the correct password, then it can trivially impersonate a user and forge any key confirmation message; otherwise we wish the attacker can only succeed with a negligible probability.

Let E_1 be the event that, for some $1 \leq x \leq n$, where $x \neq j$, there is no oracle Π_x^{y,sid_x} which satisfies $sid_x = sid_j$ and possesses the same session key as that of Π_j^{t,sid_j}. The attacker wins if E_1 occurs, and its advantage is defined to be the maximum of $\{0, \Pr[E_1] - \sum_{j=1}^{\ell_1 n_1 + 1} p_j\}$ given that there are n_1 aborted oracles at the end of the game.

6.3 Security Definitions for Key Establishment Protocols

Using the attack games defined in the previous section, we now present a number of security definitions for key establishment protocols and compare them with the definitions given in Chapter 4.

6.3.1 Security Definitions for General Case

Informally, an authenticated key establishment protocol should guarantee that, in any session, only the intended users can possibly compute the session key, even if the attacker is able to corrupt users after the session ends. Formally, security is defined as follows.

Definition 28 *An authenticated key establishment protocol is* $\mathsf{AK}^{\dagger}$*-secure if it is sound and satisfies the following requirements:*

1. *Any polynomial-time attacker only has a negligible advantage in the attack game for key randomness.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for key control.*
3. *Any polynomial-time attacker only has a negligible advantage in the attack game for key authentication.*
4. *Any polynomial-time attacker only has a negligible advantage in the attack game for backward secrecy.*
5. *Any polynomial-time attacker only has a negligible advantage in the attack game for DPUKS resilience.*

Informally, a secure authenticated key establishment protocol with mutual entity authentication should be $\mathsf{AK}^{\dagger}$-secure and further guarantee that all the uncorrupted users in a session have the same belief regarding the identifier set UID, i.e. they all

agree on who the other participants are in the session. Formally, security is defined as follows.

Definition 29 *An authenticated key establishment protocol with mutual entity authentication is* $\mathsf{AKE}^{\dagger}$*-secure, if it satisfies the following requirements:*

1. *It is* $\mathsf{AK}^{\dagger}$*-secure.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for mutual entity authentication.*

Informally, a secure authenticated key establishment protocol with key confirmation should be $\mathsf{AK}^{\dagger}$-secure and further guarantee that all intended uncorrupted users compute the same session key and session identifier in any protocol execution. Formally, security is defined as follows.

Definition 30 *An authenticated key establishment protocol with key confirmation is* $\mathsf{AKC}^{\dagger}$*-secure, if it satisfies the following requirements:*

1. *It is* $\mathsf{AK}^{\dagger}$*-secure.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for key confirmation.*

6.3.2 Security Definitions for Password Case

Analogously to the general case, we can give a corresponding series of security definitions, namely ℓ_1-$\mathsf{PAK}^{\dagger}$-security, ℓ_1-$\mathsf{PAKE}^{\dagger}$-security, and ℓ_1-$\mathsf{PAKC}^{\dagger}$-security, for the password case, where ℓ_1 is the resilience parameter. The main difference is that we should take into account the low-entropy property of the password and the property of password guessing resilience.

Definition 31 *A password-based authenticated key establishment protocol is ℓ_1-$\mathsf{PAK}^{\dagger}$-secure, if it is sound and satisfies the following requirements:*

1. *Any polynomial-time attacker only has a negligible advantage in the attack game for key randomness.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for key control.*
3. *Any polynomial-time attacker only has a negligible advantage in the attack game for key authentication (password-based) where the resilience parameter is ℓ_1.*
4. *Any polynomial-time attacker only has a negligible advantage in the attack game for password guessing resilience where the resilience parameter is ℓ_1.*

Definition 32 *A password-based authenticated key establishment protocol with mutual entity authentication is ℓ_1-$\mathsf{PAKE}^{\dagger}$-secure, if it satisfies the following requirements:*

1. *It is ℓ_1-$\mathsf{PAK}^{\dagger}$-secure.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for mutual entity authentication (password-based) where the resilience parameter is ℓ_1.*

Definition 33 *An authenticated key establishment protocol with key confirmation is ℓ_1-$\mathsf{PAKC}^{\dagger}$-secure, if it satisfies the following requirements:*

1. *It is ℓ_1-$\mathsf{PAK}^{\dagger}$-secure.*
2. *Any polynomial-time attacker only has a negligible advantage in the attack game for key confirmation (password-based) where the resilience parameter is ℓ_1.*

	Strangio*	Shoup	Canetti-Krawczyk	Bresson*	Ours
KA	yes	yes	yes	yes	yes
FS	yes	yes	yes	yes	yes
PFS	yes	yes	yes	yes	yes
BS	yes	no	no	no	yes
UKS	yes	yes	yes	yes	yes
DPUKS	no	no	no	no	yes
KR	yes	yes	yes	yes	yes
KCL	no	no	no	no	yes
EA	no	no	no	yes	yes
KK	yes	yes	yes	yes	yes
KCM	no	no	yes	no	yes
KCI	yes	no	no	no	yes

Table 6.1: Comparison of Security Models

6.3.3 Remarks and Comparisons

It is worth mentioning that the above security formulations in password-based setting are aimed at protocols which have only one password. The extension to cover other cases is beyond the scope of this thesis. Security properties such as backward secrecy, key compromise impersonation resilience, and DPUKS resilience, have not been discussed, because these properties cannot be achieved (at least by most of the existing password-based key establishment protocols). Take the key compromise impersonation resilience as an example. Recall the definition in Section 3.3, where key-compromise impersonation resilience is defined to be the property that, in any session, the compromise of a user's long-term private key does not enable the attacker to impersonate any other non-compromised intended user to the user whose key has been comprised. If a password is the only shared secret (which is the case for most of password-based protocols), clearly, it is impossible to achieve key compromise impersonation resilience.

In Table 6.1, we provide a comparison of the security properties captured by five existing security models (described in Sections 4.5) and the novel model described above.

One important property of the new model is that the session identifier can be defined in a more flexible way than in Bellare-Rogaway-like and Canetti-Krawczyk-like models. Some examples of this flexibility are given in Chapter 7.

In our security model, security is modelled in a modular way: different properties are modelled by different games, and the security definitions are based on the attacker's advantages in attack games; in each game, the attacker's privileges depend on the types of oracle queries available to the attacker. Hence, it is easy to relax our definitions to reflect changes in the powers of an attacker. One example is that, if the key establishment protocol is carried out over an authenticated network, then we can simply assume that the attacker is passive in the attack games. Another example is that, in order to model the security of traditional key distribution protocols, where a TTP distributes a session key to each user in a session, we may modify the attack games in the following ways:

1. In all the attack games, the attacker is forbidden to issue any corrupt-ttp queries.
2. Establishing the key control and DPUKS resilience properties is not required since the session key is chosen by the TTP.

6.4 Conclusions

In this chapter we have described a novel security model for key establishment protocols for both the general and passoword-based cases. We have compared it with the existing models and shown that it possesses a number of merits. The security analysis in Chapter 7 is conducted using this model.

Chapter 7

Construction of New Key Establishment Protocols

Contents

In this chapter we first describe two novel compilers for transforming key establishment protocols secure against passive attackers into protocols secure against active attackers. We then present three extensions to the Burmester-Desmedt protocol.

7.1 Motivation

As described in Section 3.1, a large number of key establishment protocols have been proposed. In addition, a number of protocol compiliers, such as those in [145, 187],

have also been proposed. Many of these protocols have been evaluated in existing security models, such as those given in [29, 68, 218]. However, in Section 4, we showed that these security models do not cover all the security properties described in Section 3.3. It is therefore an interesting challenge to design key establishment protocols that can be proven secure in the security model described in Chapter 6, since this model covers all the security properties we have identified as potentially valuable for key establishment protocols.

In Section 7.2, we review the Katz-Yung compiler [145], and propose two new compilers in an attempt to improve the efficiency of the resulting protocols. The first compiler is for environments where no TTP is involved in the key establishment process, while the other is for environments where a TTP is involved in every protocol execution. In Section 7.3 we propose a general extension to the Burmester-Desmedt protocol and prove its security in the security model introduced in Chapter 6 under the DBDH assumption in the random oracle model. In Section 7.4 we propose an extension to the Burmester-Desmedt protocol, where a shared password is used for authentication. The protocol is proved 2-PAKC†-secure under the DDH and KE assumptions in the random oracle model. In Section 7.5 we propose a modified version of the protocol described in Section 7.4, in which a number of techniques are used to reduce the risk of password guessing attacks. The protocol is proved 1-PAKC†-secure under the DDH assumption in the random oracle model.

7.2 Compilers for Protocol Transformation

7.2.1 The Katz-Yung compiler

We first review the Katz-Yung compiler [145] and discuss the security and efficiency of the protocols it produces. We then describe an example of a key establishment protocol output by this compiler, and show how to simplify this protocol.

7.2.1.1 Description of the Katz-Yung Compiler

Given any n-party key establishment protocol $\mathcal{P}$ secure against passive attackers, the Katz-Yung compiler generates a new key establishment protocol $\mathcal{P}'$ which is secure against active attackers. The security analysis of the Katz-Yung compiler, as given in [145], is in a modified version of the model of Bresson *et al.* [60].

For $\mathcal{P}'$, the TTP initialisation sub-protocol and the user initialisation sub-protocol are identical to those of $\mathcal{P}$, except that, in the user initialisation sub-protocol, U_i generates a verify/sign key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. Let ℓ be the security parameter. Suppose a set of users U_i $(1 \leq i \leq n)$ run the protocol to establish a session key.

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and broadcasts $ID_i||0||r_i$. After receiving the initial broadcast message from all the other parties, U_i sets $nonce_i = ID_1||r_1||\cdots||ID_n||r_n$ and stores this as part of its state information.
2. U_i executes the key establishment sub-protocol of $\mathcal{P}$ modified in the following ways.
   - When U_i is required by $\mathcal{P}$ to broadcast $ID_i||x||m_i$, it broadcasts $ID_i||x||m_i||\sigma_{ix}$, where $\sigma_{ix} = \mathsf{Sign}(x||m_i||nonce_i, sk_i)$.
   - When U_i receives the message $ID_j||x||m_j||\sigma_{jx}$, for any $1 \leq j \leq n$ and $j \neq i$, it checks that: (1) U_j is an intended partner, (2) x is the next expected sequence number for a message from U_j, and (3) $\mathsf{Verify}(x||m_j||nonce_i, pk_j, \sigma_{jx}) = 1$. If any of these checks fail, U_i aborts the protocol; otherwise, U_i continues.

Katz and Yung [145] claim that their compiler provides a scalable way of transforming a key establishment protocol secure against passive attackers into an authenticated protocol which is secure against active attackers. With respect to efficiency, we make the following observations on the protocols produced by the Katz-Yung compiler.

1. From the second round onwards, the compiler requires each user to sign all the

messages it sends in the protocol run, and to verify all the messages it receives. Since the total number of signature verifications in one round is proportional to the group size, the signature verifications will potentially use a significant amount of computational resource when the group size is large.

2. The compiler adds an additional round to the original key establishment sub-protocol; however, it does not provide key confirmation. As Katz and Yung state [145], in order to achieve key confirmation a further additional round is required.

7.2.1.2 An example application of the compiler

The following authenticated group key exchange protocol was obtained by applying the Katz-Yung compiler to the Burmester-Desmedt protocol.

The TTP runs the TTP initialisation sub-protocol to generate $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a multiplicative cyclic group of prime order q and g is a generator of $\mathbb{G}$. In addition, the TTP also generates its public/private signature key pair, as used to certify users' public keys. Every user U_i $(i \geq 1)$ runs the user initialisation sub-protocol to generate a key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and get pk_i certified by the TTP.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and broadcasts $ID_i||0||r_i$. After receiving the initial broadcast message from all other parties, U_i sets $nonce_i = ID_1||r_1||\cdots||ID_n||r_n$ and stores it as part of its state information.

2. U_i chooses $s_i \in_R Z_q$ and computes $Z_i = g^{s_i}$. U_i then computes a signature $\sigma_{i1} = \mathsf{Sign}(1||Z_i||nonce_i, sk_i)$ and broadcasts $ID_i||1||Z_i||\sigma_{i1}$.

3. When U_i receives a message $ID_j||1||Z_j||\sigma_{j1}$ from user U_j $(1 \leq j \leq n, j \neq i)$, it checks that: (1) U_j is an intended partner, (2) 1 is the next expected sequence

number for a message from U_j, and (3) $\mathsf{Verify}(1||Z_j||nonce_i, pk_j, \sigma_{j1}) = 1$. If any of these checks fail, U_i aborts the protocol. Otherwise, U_i computes $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$ and the signature $\sigma_{i2} = \mathsf{Sign}(2||X_i||nonce_i, sk_i)$, and broadcasts $ID_i||2||X_i||\sigma_{i2}$.

4. When U_i receives a message $ID_j||2||X_j||\sigma_{j2}$ from user U_j $(1 \leq j \leq n, j \neq i)$, it checks that: (1) U_j is an intended partner, (2) 2 is the next expected sequence number for a message from U_j, and (3) $\mathsf{Verify}(2||X_j||nonce_i, pk_j, \sigma_{j2}) = 1$. If any of these checks fail, U_i aborts the protocol; otherwise, U_i computes the session key as:

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

It is straightforward to verify that the above protocol has three rounds. Each user needs to sign two messages and verify $2n$ signatures, besides the other computations involved in performing the protocol $\mathcal{P}$.

7.2.1.3 Simplified Version of the Example Scheme

In our simplified version of the above scheme, the TTP initialisation sub-protocol and the user initialisation sub-protocol are unchanged, while the key establishment sub-protocol is as follows:

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and $s_i \in_R \mathbb{Z}_q$, computes $Z_i = g^{s_i}$, and then broadcasts $ID_i||0||r_i||Z_i$.

2. U_i sets $nonce_i = ID_1||r_1||\cdots||ID_n||r_n$ and stores it as state information. U_i computes $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$, $h_i = Z_1||Z_2||\cdots||Z_n$, and $\sigma_{i1} = \mathsf{Sign}(1||X_i||nonce_i||h_i, sk_i)$. U_i then broadcasts $ID_i||1||X_i||\sigma_{i1}$.

3. When U_i receives a message $ID_j||1||X_j||\sigma_{j1}$ from user U_j $(1 \leq j \leq n, j \neq i)$, it checks that: (1) U_j is an intended partner, (2) 1 is the expected sequence number, and (3) $\mathsf{Verify}(1||X_j||nonce_i||h_i, pk_j, \sigma_{j1}) = 1$. If any of these checks

fail, U_i aborts the protocol; otherwise, U_i computes the session key as:

$$\begin{aligned} K_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

It is straightforward to verify that this simplified sub-protocol is more efficient than that obtained from directly applying the compiler in two respects: the protocol has only two rounds, and each user only needs to compute one signature and verify n signatures. To assess the security of this protocol, we first compare our simplified scheme with that created by the Katz-Yung compiler. There are two main differences between the key establishment sub-protocols:

1. One difference is that in the simplified algorithm we combine the first two rounds of the original key establishment sub-protocol into one round, and avoid the signature computation.
2. The other difference is that we require each user to compute the signature on both X_i and $Z_1||\cdots||Z_n$ in the second round.

In summary, the main change is that, compared with the original key establishment sub-protocol, the authentication of messages sent in the first round is conducted in the second round. As a result, it straightforward to verify that the simplified protocol is as secure as the example protocol. This simplification motivates the design of the two novel compilers we next describe.

7.2.2 Compiler without Centralised Verification

In this section we describe a new compiler, which can transform a key establishment protocol secure against passive attackers into a protocol secure against active attackers, where security is analysed in the security model described in Chapter 6. The resulting protocol is designed for use in an environment where no TTP is involved in the key establishment process.

7.2.2.1 Description of the compiler

Given a key establishment protocol $\mathcal{P}$ which is secure against passive attackers, this compiler generates a new key establishment protocol $\mathcal{P}'$ secure against active attackers. We suppose that, at the end of a successful session, every user outputs a session key and a session identifier. Suppose also that if two users possess the same session identifier and transcript then they compute the same session key.

For $\mathcal{P}'$, the TTP initialisation sub-protocol and user initialisation sub-protocol are identical to those of $\mathcal{P}$, except that, in the user initialisation sub-protocol, U_i $(i \geq 1)$ also generates a verify/sign key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and in the TTP initialisation sub-protocol the TTP also selects two hash functions $\mathsf{H_1} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$ and $\mathsf{H_2} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol of $\mathcal{P}'$ is as follows.

1. U_i $(1 \leq i \leq n)$ executes the key establishment sub-protocol of $\mathcal{P}$, in which protocol messages are broadcast to every user.

2. After computing the session key K_i', U_i broadcasts its key confirmation message $ID_i||sid_i||\mathsf{H_1}(K_i')||\sigma_i$, where K_i' is the session key, sid_i is the session identifier, $trans_i$ is the concatenation of all protocol messages, and $\sigma_i = \mathsf{Sign}(ID_i||sid_i||trans_i||\mathsf{H_1}(K_i'), sk_i)$

3. After receiving $ID_j||sid_j||\mathsf{H_1}(K_j')||\sigma_j$, for every $1 \leq j \leq n$ and $j \neq i$, U_i checks whether the following equation holds for j:

$$\mathsf{Verify}(ID_j||sid_i||trans_i||\mathsf{H_1}(K_i'), pk_j, \sigma_j) = 1.$$

   If all the verifications succeed, U_i computes the session key as $K_i = \mathsf{H_2}(K_i'||sid_i)$ and sets the session identifier to sid_i. If any of these checks fail, U_i aborts the protocol.

7.2.2.2 Security and Performance Analysis

In the security model described in Chapter 6, we have the following security result.

Theorem 5 *If the hash functions are modelled as random oracles, the compiler transforms a* $\mathsf{AK}^{\dagger}$*-secure protocol* $\mathcal{P}$ *secure against passive attackers into an* $\mathsf{AKC}^{\dagger}$ *secure protocol* $\mathcal{P}'$ *secure against active attackers.*

Proof. From the definitions, it is straightforward to verify that key randomness and key control are guaranteed for $\mathcal{P}'$, given that H_2 is a random oracle. We next prove that an active attacker has only a negligible advantage in the attack games for DPUKS resilience, key confirmation, key authentication, and backward secrecy.

- For $\mathcal{P}'$, consider an attack game for DPUKS resilience (described in Section 6.2.2.5). The attacker wins if, for some $1 \leq j \leq n$, $t \geq 1$, $y \geq 1$, and $x \geq 1$, $sid_j \neq sid_x$, and Π_j^{t,sid_j} and Π_x^{y,sid_x} possess the same session key.

  According to the definition of the compiler described in Section 7.2.2.1, $K_j = \mathsf{H}_2(K'_j||sid_j)$ and $K_x = \mathsf{H}_2(K'_x||sid_x)$. It is straightforward to verify that the attacker's advantage is negligible given that H_2 is collision-resistant.

- For $\mathcal{P}'$, consider an attack game for key confirmation (described in Section 6.2.2.7). The attacker wins, if for some $1 \leq x \leq n$, there is no oracle for U_x where $sid_x = sid_j$ and $\mathsf{H}_2(K'_x||sid_x) = \mathsf{H}_2(K'_j||sid_j)$. Hence, if the attacker wins then the probability ϵ that the pair $(ID_x||sid_j||trans_j||\mathsf{H}_1(K'_j), \sigma_x)$ is generated by U_x (simulated by the challenger) is negligible given that H_1 and H_2 are collision-resistant.

  According to the definition of the compiler described in Section 7.2.2.1, before an oracle Π_j^{t,sid_j} accepts, it needs to verify the signatures σ_i $(1 \leq i \leq n, i \neq j)$, where $\sigma_i = \mathsf{Sign}(ID_i||sid_j||trans_j||\mathsf{H}_1(K'_j), sk_i)$. Therefore, if the attacker wins with probability δ, then we can conclude that it has successfully forged a pair $(ID_x||sid_j||trans_j||\mathsf{H}_1(K'_j), \sigma_x)$ with a probability δ', where the value of $|\delta - \delta'| = \epsilon$ is the probability that the pair is generated by the challenger. From the previous analysis, $|\delta - \delta'|$ is negligible. As a result, the attacker's advantage

δ is negligible given that the signature scheme is existentially unforgeable under a chosen message attack.

Observation 1 *From the above analysis, it is straightforward to verify that the probability that* Π_j^{t,sid_j} *and its partner oracle possess different transcripts is negligible, given that the signature scheme is existentially unforgeable under a chosen message attack.*

- For $\mathcal{P}'$, consider an attack game for key authentifcation (described in Section 6.2.2.3). Let the advantage of the attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be δ. We construct an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the key authentication property of $\mathcal{P}$. Without loss of generality, suppose that $\mathcal{A}$ initiates at most m oracles for each user in Phase 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and proceeds as follows:

  1. $\mathcal{B}$ runs the attack game for key authentication for $\mathcal{P}$ with the challenger. In the game, $\mathcal{B}$ acts as a passive attacker. Let the tested oracle be Π_j^{t,sid_j} and the challenge be C_b', which is either K_j' or a random string. Note that $\mathcal{B}_2$ can obtain all the long-term credentials $param_2$ for $\mathcal{P}$ by issuing corrupt-user and corrupt-ttp queries in Phase 2.
  2. $\mathcal{B}_2$ generates the parameters $param_1'$ and $param_2'$ for protocol $\mathcal{P}'$, where $param_1'$ includes $param_1$ and the private signing keys sk_i $(i \geq 1)$, and $param_2'$ includes $param_2$, H_1, H_2, and the verification keys pk_i $(i \geq 1)$. Then $\mathcal{B}_2$ runs $\mathcal{A}_1$ with input $param_1'$ and honestly answers the oracle queries issued by $\mathcal{A}_1$. During the simulation, for every $1 \leq i \leq n$, $\mathcal{B}$ randomly chooses $1 \leq o_i \leq m$, and makes $\Pi_j^{o_j,sid_j}$ output the same messages as those of Π_j^{t,sid_j}. In addition, $\Pi_i^{o_i,sid_i}$, for every $1 \leq i \leq n, i \neq j$, outputs the same messages as those of Π_j^{t,sid_j}'s corresponding partner oracle which belongs to U_i. $\mathcal{B}$ aborts if any of the following events occur.
     - The tested oracle is not in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$.
     - There are two oracles in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$ which are not partner oracles to each other, possess different session keys, or possess different transcripts.
  3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ with input the state information (generated by $\mathcal{A}_1$) and $\mathsf{H}_2(C_b'||sid_i)$, where sid_i is the session identifier of the tested oracle $\Pi_i^{o_i,sid_i}$ for some $1 \leq i \leq n$. $\mathcal{B}_2$ answers the oracle queries issued by $\mathcal{A}_2$ following the protocol specification. Note that $\mathcal{A}_2$ is forbidden to issue a

corrupt-state or reveal query to the tested oracle or its partner oracle, therefore, $\mathcal{B}_2$ can faithfully answer any query which may be issued by $\mathcal{A}_2$.

4. If $\mathcal{A}_2$ terminates by outputting b', then $\mathcal{B}_2$ terminates by outputting b'.

Let E_1 be the event that $\mathcal{B}_2$ does not abort in step 2. Because $\mathcal{P}'$ achieves key confirmation (and also from Observation 1), it is straightforward to verify that $\Pr[E_1] = \frac{1}{m^n} - \epsilon_1$ where ϵ_1 is negligible. If E_1 occurs, $\mathcal{B}$ has faithfully answered the oracle queries issued by $\mathcal{A}$. If δ' is the probability that $\mathcal{B}$ wins when it does not abort in step 2, then $\delta' = \delta$. In summary, $\mathcal{B}$ can win the attack game with advantage δ'', where

$$\begin{aligned} \delta'' &= \Pr[E_1]\delta' \\ &= (\frac{1}{m^n} - \epsilon_1)\delta \\ &\geq \frac{\delta}{m^n} - \epsilon_1 \end{aligned}$$

Since $\frac{1}{m^n}$ is a constant, then if δ is non-negligible then so is δ'' (given that ϵ_1 is negligible). However, this contradicts the assumption that $\mathcal{P}$ achieves key authentication, and hence δ is negligible.

- For $\mathcal{P}'$, consider an attack game for backward secrecy (described in Section 6.2.2.4). Let the advantage of the attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be δ. We construct an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the key authentication property of $\mathcal{P}$. Without loss of generality, suppose that $\mathcal{A}$ initiates at most m oracles for each user in Phase 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and runs as follows:

  1. $\mathcal{B}$ runs the attack game for backward secrecy for $\mathcal{P}$ with the challenger. In the game, $\mathcal{B}$ acts as a passive attacker. Let the tested oracle be Π_j^{t,sid_j} and the challenge be C'_b, which is either K'_j or a random string. Note that $\mathcal{B}_2$ can obtain all the long-term credentials $param_2$ for $\mathcal{P}$, by issuing corrupt-user and corrupt-ttp queries in Phase 2.

  2. $\mathcal{B}_2$ generates the parameters $param'_1$ and $param'_2$ for protocol $\mathcal{P}'$, where $param'_1$ includes $param_1$ and the private signing keys sk_i ($i \geq 1$), and $param'_2$ includes $param_2$, H_1, H_2, and the verification keys pk_i ($i \geq 1$). Then $\mathcal{B}_2$ runs $\mathcal{A}_1$ with input $param'_1$ and honestly answers the oracle queries issued by $\mathcal{A}_1$. During the simulation, for every $1 \leq i \leq n$, $\mathcal{B}$ randomly chooses $1 \leq o_i \leq m$, and makes $\Pi_j^{o_j,sid_j}$ output the same messages

as those of Π_j^{t,sid_j}. In addition, $\Pi_i^{o_i,sid_i}$, for every $1 \leq i \leq n, i \neq j$, outputs the same messages as those of Π_j^{t,sid_j}'s corresponding partner oracle which belongs to U_i. $\mathcal{B}$ aborts if any of the following events occur.

- The tested oracle is not in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$.
- There are two oracles in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$ which are not partner oracles to each other, possess different session keys, or possess different transcripts.

3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ with input the state information (generated by $\mathcal{A}_1$) and $\mathsf{H}_2(C'_b||sid_i)$, where sid_i is the session identifier of the tested oracle $\Pi_i^{o_i,sid_i}$ for some $1 \leq i \leq n$. $\mathcal{B}_2$ answers the oracle queries issued by $\mathcal{A}_2$ following the protocol specification.
4. If $\mathcal{A}_2$ terminates by outputting b', then $\mathcal{B}_2$ terminates by outputting b'.

Let E_1 be the event that $\mathcal{B}_2$ does not abort in step 2. When $\mathcal{A}_1$ issues its $\mathsf{test}(\Pi_j^{o_j,sid_j})$ query, if no corrupt-user query has been issued to U_j, then from the security analysis of key confirmation (and also from Observation 1), $|\Pr[E_1] - \frac{1}{m^n}|$ is negligible. Otherwise, if $\mathcal{A}_1$ has issued a corrupt-user query to U_j, the probability that $\Pi_j^{o_j,sid_j}$ possesses the same transcript with its partner oracle is also overwhelming because no user of the tested oracle's partner oracle is permitted to be corrupted (in the game for backward secrecy). As a result, in both case, we can conclude that $\Pr[E_1] = \frac{1}{m^n} - \epsilon_1$ where ϵ_1 is negligible.

Note that $\mathcal{A}_2$ is forbidden to issue a corrupt-state or reveal query to the tested oracle and its partner oracle, therefore, in step 3, $\mathcal{B}_2$ can faithfully answer any query which may be issued by $\mathcal{A}_2$.

If E_1 occurs, $\mathcal{B}$ has faithfully answered the oracle queries issued by $\mathcal{A}$. If δ' is the probability that $\mathcal{B}$ wins when it does not abort in step 2, then $\delta' = \delta$. In summary, $\mathcal{B}$ can win the attack game with advantage δ'', where

$$\begin{aligned} \delta'' &= \Pr[E_1]\delta' \\ &= (\frac{1}{m^n} - \epsilon_1)\delta \\ &\geq \frac{\delta}{m^n} - \epsilon_1 \end{aligned}$$

Since $\frac{1}{m^n}$ is a constant, then if δ is non-negligible then so is δ'' (given that ϵ_1 is negligible). However, this contradicts the assumption that $\mathcal{P}$ achieves backward secrecy, and hence δ is negligible.

In summary, we have shown that an attacker has only a negligible probability in the attack games for key randomness, key control, key authentication, backward secrecy, DPUKS resilience, and key confirmation. As a result, $\mathcal{P}'$ is $\mathsf{AKC}^{\dagger}$ secure, and the theorem follows. □

The key establishment sub-protocol of $\mathcal{P}'$ adds one round to the original protocol and achieves key confirmation. Every participant needs to sign one message and verify n signatures, in addition to the computations involved in performing the original protocol. Thus this compiler yields protocols that are more efficient than those produced by the Katz-Yung compiler.

7.2.2.3 An Example Application of this Compiler

We first introduce a modified version of the Burmester-Desmedt protocol (described in Section 3.1.2.2). The modified protocol is identical to the original protocol except that the session identifier is defined to be $(\mathrm{UID}, \mathrm{RSID})$, where $\mathrm{RSID} = Z_1||Z_2||\cdots||Z_n||X_1||X_2||\cdots||X_n$, and the session key is $K_i' = \mathsf{H_0}(K_i''||sid_i)$, where $\mathsf{H_0}$ is a hash function and

$$\begin{aligned} K_i'' &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1s_2+s_2s_3+s_3s_4+\cdots+s_ns_1}. \end{aligned}$$

From the security of the Burmester-Desmedt protocol, the modified protocol achieves key authentication and backward secrecy against passive attackers. The modified protocol achieves key randomness given that $\mathsf{H_0}$ is a random oracle. The modified protocol achieves key control and DPUKS resilience against passive attackers, because the session key is computed as a function of K_i'' and the unique session identifier sid_i. As a result, the modified protocol is $\mathsf{AK}^{\dagger}$-secure.

The following key establishment protocol is obtained by applying the compiler given in Section 7.2.2.1 to the above protocol.

The TTP runs the TTP initialisation sub-protocol to generate $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is a multiplicative cyclic group of prime order q and g is a generator of $\mathbb{G}$, and three

hash functions $\mathsf{H}_0 : \{0,1\}^* \longrightarrow \{0,1\}^\ell$, $\mathsf{H}_1 : \{0,1\}^* \longrightarrow \{0,1\}^\ell$, and $\mathsf{H}_2 : \{0,1\}^* \longrightarrow \{0,1\}^\ell$. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and get pk_i certified by the TTP.

Suppose a set of users U_i ($1 \leq i \leq n$) wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $s_i \in_R \mathbb{Z}_q$ and broadcasts $Z_i = g^{s_i}$.
2. After receiving every Z_j ($1 \leq j \leq n, j \neq i$), U_i broadcasts $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$.
3. After receiving every X_j ($1 \leq j \leq n, j \neq i$), U_i sets $sid_i = (\mathrm{UID}, \mathrm{RSID})$ where $\mathrm{RSID} = Z_1||Z_2||\cdots||Z_n||X_1||X_2||\cdots||X_n$, $trans_i = RSID$, and $K_i' = \mathsf{H}_0(K_i''||sid_i)$, where

$$\begin{aligned} K_i'' &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1} \end{aligned}$$

   and broadcasts σ_i, where

$$\sigma_i = \mathsf{Sign}(ID_i||sid_i||trans_i||\mathsf{H}_1(K_i'), sk_i).$$

4. After receiving σ_j ($1 \leq j \leq n, j \neq i$), U_i checks whether the following equation holds for every j ($j \neq i$):

$$\mathsf{Verify}(ID_j||sid_i||trans_i||\mathsf{H}_1(K_i'), pk_j, \sigma_j) = 1.$$

   If all the verifications succeed, then U_i computes the session key $K_i = \mathsf{H}_2(K_i'||sid_i)$. Otherwise, if any of these verifications fail, then U_i aborts the protocol execution.

The key establishment sub-protocol has three rounds. Each user needs to conduct three exponentiations, sign one message, and verify n signatures.

7.2.3 Compiler with Centralised Verification

In this section we describe a new compiler which can transform a key establishment protocol secure against passive attackers into a protocol secure against active attackers, where security is analysed in the model described in Chapter 6. The resulting protocol is designed for use in an environment where a TTP is involved in key establishment process.

7.2.3.1 Description of the compiler

Given a key establishment protocol $\mathcal{P}$ which is secure against passive attackers, this compiler generates a new key establishment protocol $\mathcal{P}'$ secure against active attackers. We suppose that, at the end of a successful session, every user outputs a session key and a session identifier. Suppose also that if two users possess the same session identifier and transcript then they compute the same session key.

For $\mathcal{P}'$, the TTP initialisation sub-protocol and the user initialisation sub-protocol are identical to those of $\mathcal{P}$, except that, in the user initialisation sub-protocol every user U_i, for $i \geq 1$, generates a verify/sign key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and in the TTP initialisation sub-protocol, the TTP generates a verify/sign key pair (pk_{ttp}, sk_{ttp}) for the same signature scheme and two hash functions $\mathsf{H}_1 : \{0,1\}^* \longrightarrow \{0,1\}^\ell$ and $\mathsf{H}_2 : \{0,1\}^* \longrightarrow \{0,1\}^\ell$.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol of $\mathcal{P}'$ is as follows.

1. U_i executes the key establishment sub-protocol of $\mathcal{P}$.
2. After computing the session key K_i', U_i sends the key confirmation message $ID_i||sid_i||\mathsf{H}_1(trans_i)||\mathsf{H}_1(K_i')||\sigma_i$ to the TTP, where K_i' is the session key, sid_i is the session identifier, $trans_i$ is the concatenation of all protocol messages, and $\sigma_i = \mathsf{Sign}(ID_i||sid_i||\mathsf{H}_1(trans_i)||\mathsf{H}_1(K_i'), sk_i)$.
3. The TTP verifies that $sid_1 = sid_2 = \cdots = sid_n$, $\mathsf{H}_1(trans_1) = \mathsf{H}_1(trans_2) = \cdots = \mathsf{H}_1(trans_n)$, $\mathsf{H}_1(K_1') = \mathsf{H}_1(K_2') = \cdots = \mathsf{H}_1(K_n')$, and every signature is

valid. If all the verifications succeed, the TTP broadcasts $sid_1||\mathsf{H_1}(trans_1)||\mathsf{H_1}(K_1')||\sigma_{ttp}$, where

$$\sigma_{ttp} = \mathsf{Sign}(sid_1||\mathsf{H_1}(trans_1)||\mathsf{H_1}(K_1'), sk_{ttp}).$$

Otherwise, the TTP broadcasts a failure message.

4. U_i verifies that the signature from the TTP is valid, $sid_1 = sid_i$, $\mathsf{H_1}(trans_1) = \mathsf{H_1}(trans_i)$, and $\mathsf{H_1}(K_1') = \mathsf{H_1}(K_i')$. If all these verifications succeed, U_i computes the session key as $K_i = \mathsf{H_2}(K_i'||sid_i)$ and sets its session identifier to sid_i. If any of these verifications fail, or if U_i receives a failure message from the TTP, then U_i aborts the protocol.

7.2.3.2 Security and Performance Analysis

In the security model described in Chapter 6, we have the following security result.

Theorem 6 *If the hash functions are modelled as random oracles, the compiler transforms a* $\mathsf{AK}^\dagger$ *secure protocol* $\mathcal{P}$ *secure against passive attackers into an* $\mathsf{AKC}^\dagger$ *secure protocol* $\mathcal{P}'$ *secure against active attackers, which are not permitted to corrupt the TTP except with respect to the forward secrecy and backward secrecy properties*[1].

Proof. From the definitions, it is straightforward to verify that key randomness and key control are guaranteed for $\mathcal{P}'$, given that $\mathsf{H_2}$ is collision-resistant. We next prove that an active attacker has only a negligible advantage in the attack games for DPUKS resilience, key confirmation, key authentication, and backward secrecy.

- For $\mathcal{P}'$, consider an attack game for DPUKS resilience (described in Section 6.2.2.5). The attacker wins if, for some $1 \leq j \leq n$, $t \geq 1$, $y \geq 1$, and $x \geq 1$, $sid_j \neq sid_x$, and Π_j^{t,sid_j} and Π_x^{y,sid_x} possess the same session key.

  According to the definition of the compiler described in Section 7.2.3.1, $K_j = \mathsf{H_2}(K_j'||sid_j)$ and $K_x = \mathsf{H_2}(K_x'||sid_x)$. It is straightforward to verify that the attacker's advantage is negligible given that $\mathsf{H_2}$ is collision-resistant.

[1] Use of corrupt-ttp is only permitted in phase 2 in the attack games for key authentication and backward secrecy.

- For $\mathcal{P}'$, consider an attack game for key confirmation (described in Section 6.2.2.7). The attacker wins if, for some $1 \leq x \leq n$, there is no oracle for U_x where $sid_x = sid_j$ and $\mathsf{H}_2(K'_x||sid_x) = \mathsf{H}_2(K'_j||sid_j)$. Hence, if the attacker wins, then the probability that the pair $(ID_x||sid_j||\mathsf{H}_1(trans_j)||\mathsf{H}_1(K'_j), \sigma_x)$ is generated by U_x (simulated by the challenger) is negligible given that H_1 and H_2 are collision-resistant.

  According to the definition of the compiler described in Section 7.2.3.1, before an oracle Π_j^{t,sid_j} accepts, it needs to verify $sid_1||\mathsf{H}_1(trans_1)||\mathsf{H}_1(K'_1)||\sigma_{ttp}$, where

  $$\sigma_{ttp} = \mathsf{Sign}(sid_1||\mathsf{H}_1(trans_1)||\mathsf{H}_1(K'_1), sk_{ttp}).$$

  Therefore, if the attacker wins with probability δ, then we can conclude that it has forged $(ID_x||sid_j||\mathsf{H}_1(trans_j)||\mathsf{H}_1(K'_j), \sigma_x)$ or $(sid_1||\mathsf{H}_1(trans_1)||\mathsf{H}_1(K'_1), \sigma_{ttp})$ with a probability δ', where the value of $|\delta - \delta'|$ is the probability that one of these pairs is generated by the challenger. Given that the signature is existentially unforgeable under a chosen message attack, the probability $|\delta - \delta'|$ is negligible. As a result, the attacker's advantage δ is negligible in winning the game.

  Observation 2 *From the above analysis, it is straightforward to verify that the probability that Π_j^{t,sid_j} and its partner oracle possess different transcripts is negligible, given that the signature scheme is existentially unforgeable under a chosen message attack.*

- For $\mathcal{P}'$, consider an attack game for key authentication (described in Section 6.2.2.3). Let the advantage of the attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be δ. We construct an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the key authentication property of $\mathcal{P}$. Without loss of generality, suppose that $\mathcal{A}$ initiates at most m oracles for each user in Phase 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and works as follows:

  1. $\mathcal{B}$ runs the attack game for key authentifcation for $\mathcal{P}$ with the challenger. In the game, $\mathcal{B}$ acts as a passive attacker. Let the tested oracle be Π_j^{t,sid_j} and the challenge be C'_b, which is either K'_j or a random string. Note that $\mathcal{B}_2$ can obtain all the long-term credentials $param_2$ for $\mathcal{P}$, by issuing corrupt-user and corrupt-ttp queries in Phase 2.
  2. $\mathcal{B}_2$ generates the parameters $param'_1$ and $param'_2$ for protocol $\mathcal{P}'$, where $param'_1$ includes $param_1$, the private signing keys sk_i $(i \geq 1)$, and the

private signing key sk_{ttp} of the TTP, and $param_2'$ includes $param_2$, H_1, H_2, the verification keys pk_i $(i \geq 1)$, and the verification key pk_{ttp} of the TTP. Then $\mathcal{B}_2$ runs $\mathcal{A}_1$ with input $param_1'$ and honestly answers the oracle queries issued by $\mathcal{A}_1$. During the simulation, for every $1 \leq i \leq n$, $\mathcal{B}$ randomly chooses $1 \leq o_i \leq m$, and makes $\Pi_j^{o_j,sid_j}$ output the same messages as those of Π_j^{t,sid_j}. In addition, $\Pi_i^{o_i,sid_i}$, for every $1 \leq i \leq n, i \neq j$, outputs the same messages as those of Π_j^{t,sid_j}'s corresponding partner oracle which belongs to U_i. $\mathcal{B}$ aborts if any of the following events occur.

- The tested oracle is not in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$.
- There are two oracles in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$ which are not partner oracles to each other, possess different session keys, or possess different transcripts.

3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ with input the state information (generated by $\mathcal{A}_1$) and $\mathsf{H}_2(C_b'||sid_i)$, where sid_i is the session identifier of the tested oracle $\Pi_i^{o_i,sid_i}$ for some $1 \leq i \leq n$. $\mathcal{B}_2$ answers the oracle queries issued by $\mathcal{A}_2$ following the protocol specification. Note that $\mathcal{A}_2$ is forbidden to issue a corrupt-state or reveal query to the tested oracle or its partner oracle, therefore, $\mathcal{B}_2$ can faithfully answer any query which may be issued by $\mathcal{A}_2$.
4. If $\mathcal{A}_2$ terminates by outputting b', then $\mathcal{B}_2$ terminates by outputting b'.

Let E_1 be the event that $\mathcal{B}_2$ does not abort in step 2. Because $\mathcal{P}'$ achieves key confirmation (and also from Observation 2), it is straightforward to verify that $\Pr[E_1] = \frac{1}{m^n} - \epsilon_1$, where ϵ_1 is negligible. If E_1 occurs, $\mathcal{B}$ has faithfully answered the oracle queries issued by $\mathcal{A}$. If δ' is the probability that $\mathcal{B}$ wins when it does not abort in step 2, then $\delta' = \delta$. In summary, $\mathcal{B}$ can win the attack game with advantage δ'', where

$$\begin{aligned} \delta'' &= \Pr[E_1]\delta' \\ &= (\frac{1}{m^n} - \epsilon_1)\delta \\ &\geq \frac{\delta}{m^n} - \epsilon_1 \end{aligned}$$

Since $\frac{1}{m^n}$ is a constant, then if δ is non-negligible then so is δ'' (given that ϵ_1 is negligible). However, this contradicts the assumption that $\mathcal{P}$ achieves key authentication, and hence δ is negligible.

- For $\mathcal{P}'$, consider an attack game for backward secrecy (described in Section 6.2.2.4). Let the advantage of the attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be δ. We construct

an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ for the backward secrecy property of $\mathcal{P}$. Without loss of generality, suppose that $\mathcal{A}$ initiates at most m oracles for each user in Phase 1. $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and works as follows:

1. $\mathcal{B}$ runs the attack game for key authentication for $\mathcal{P}$ with the challenger. In the game, $\mathcal{B}$ acts as a passive attacker. Let the tested oracle be Π_j^{t,sid_j} and the challenge be C'_b, which is either K'_j or a random string. Note that $\mathcal{B}_2$ can obtain all the long-term credentials $param_2$ for $\mathcal{P}$ by issuing corrupt-user and corrupt-ttp queries in Phase 2.
2. $\mathcal{B}_2$ generates the parameters $param'_1$ and $param'_2$ for protocol $\mathcal{P}'$, where $param'_1$ includes $param_1$, the private signing keys sk_i $(i \geq 1)$, and the private signing key sk_{ttp} of the TTP, and $param'_2$ includes $param_2$, H_1, H_2, the verification keys pk_i $(i \geq 1)$, and the verification key pk_{ttp} of the TTP. Then $\mathcal{B}_2$ runs $\mathcal{A}_1$ with input $param'_1$ and honestly answers the oracle queries issued by $\mathcal{A}_1$. During the simulation, for every $1 \leq i \leq n$, $\mathcal{B}$ randomly chooses $1 \leq o_i \leq m$, and makes $\Pi_j^{o_j,sid_j}$ output the same messages as those of Π_j^{t,sid_j}. In addition, $\Pi_i^{o_i,sid_i}$, for every $1 \leq i \leq n, i \neq j$, outputs the same messages as those of Π_j^{t,sid_j}'s corresponding partner oracle which belongs to U_i. $\mathcal{B}$ aborts if any of the following events occur.
   - The tested oracle is not in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$.
   - There are two oracles in the set $\{\Pi_i^{o_i,sid_i}\ (1 \leq i \leq n)\}$ which are not partner oracles to each other, possess different session keys, or possess different transcripts.
3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ with input the state information (generated by $\mathcal{A}_1$) and $\mathsf{H}_2(C'_b||sid_i)$, where sid_i is the session identifier of the tested oracle $\Pi_i^{o_i,sid_i}$ for some $1 \leq i \leq n$. $\mathcal{B}_2$ answers the oracle queries issued by $\mathcal{A}_2$ following the protocol specification.
4. If $\mathcal{A}_2$ terminates by outputting b', then $\mathcal{B}_2$ terminates by outputting b'.

Let E_1 be the event that $\mathcal{B}_2$ does not abort in step 2. When $\mathcal{A}_1$ issues its $\mathsf{test}(\Pi_j^{o_j,sid_j})$ query, if no corrupt-user query has been issued to U_j, then from the security analysis of key confirmation (and also from Observation 2), $|\Pr[E_1] - \frac{1}{m^n}|$ is negligible. Otherwise, if $\mathcal{A}_1$ has issued a corrupt-user query to U_j, the probability that $\Pi_j^{o_j,sid_j}$ possesses the same transcript with its partner oracle is also overwhelming, because the attacker is forbidden to corrupt the user

corresponding to the tested oracle's partner oracle and the TTP. As a result, in both case, we can conclude that $\Pr[E_1] = \frac{1}{m^n} - \epsilon_1$ where ϵ_1 is negligible.

Note that $\mathcal{A}_2$ is forbidden to issue a corrupt-state or reveal query to the tested oracle or its partner oracle, therefore, in step 3 $\mathcal{B}_2$ can faithfully answer any query which may be issued by $\mathcal{A}_2$.

If E_1 occurs, $\mathcal{B}$ has faithfully answered the oracle queries issued by $\mathcal{A}$. If δ' is the probability that $\mathcal{B}$ wins when it does not abort in step 2, then $\delta' = \delta$. In summary, $\mathcal{B}$ can win the attack game with advantage δ'', where

$$\begin{aligned} \delta'' &= \Pr[E_1]\delta' \\ &= (\frac{1}{m^n} - \epsilon_1)\delta \\ &\geq \frac{\delta}{m^n} - \epsilon_1 \end{aligned}$$

Since $\frac{1}{m^n}$ is a constant, then if δ is non-negligible then so is δ'' (given that ϵ_1 is negligible). However, this contradicts the assumption that $\mathcal{P}$ achieves backward secrecy, and hence δ is negligible.

In summary, we have shown that an attacker has only a negligible probability in the attack games for key randomness, key control, key authentication, backward secrecy, DPUKS resilience, and key confirmation. As a result, $\mathcal{P}'$ is $\mathsf{AKC}^{\dagger}$ secure, and the theorem follows. □

The key establishment sub-protocol of $\mathcal{P}'$ has two more rounds than the original sub-protocol, but every participant only needs to sign one message and verify one signature, in addition to the computations involved in performing the input protocol. The TTP needs to verify n signatures and generate one signature.

7.2.3.3 An Example Application of this Compiler

The following protocol is obtained by applying the compiler given in Section 7.2.3.1 to the modified Burmester-Desmedt protocol described in Section 7.2.2.3.

The TTP runs the TTP initialisation sub-protocol to generate $(\mathbb{G}, q, g)$, where $\mathbb{G}$ is

a multiplicative cyclic group of prime order q and g is a generator of $\mathbb{G}$, and three hash functions $\mathsf{H_0} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$, $\mathsf{H_1} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$, and $\mathsf{H_2} : \{0,1\}^* \longrightarrow \{0,1\}^\ell$. In addition, the TTP also generates its public/private key pair, as used to certify users' public keys. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a key pair (pk_i, sk_i) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and get pk_i certified by the TTP.

Suppose a set of users U_i ($1 \leq i \leq n$) wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $s_i \in_R Z_q$ and broadcasts $Z_i = g^{s_i}$.
2. After receiving every Z_j ($1 \leq j \leq n$), U_i broadcasts $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$.
3. After receiving every X_j ($1 \leq j \leq n, j \neq i$), U_i sets $sid_i = (\mathrm{UID}, \mathrm{RSID})$, where $\mathrm{RSID} = Z_1||Z_2||\cdots||Z_n||X_1||X_2||\cdots||X_n$, $trans_i = RSID$, and $K_i' = \mathsf{H_0}(K_i''||sid_i)$, where

$$\begin{aligned} K_i'' &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1s_2+s_2s_3+s_3s_4+\cdots+s_ns_1}, \end{aligned}$$

   and sends $ID_i||sid_i||\mathsf{H_1}(trans_i)||\mathsf{H_1}(K_i')||\sigma_i$ to the TTP, where

$$\sigma_i = \mathsf{Sign}(ID_i||sid_i||\mathsf{H_1}(trans_i)||\mathsf{H_1}(K_i'), sk_i).$$

4. The TTP verifies that $sid_1 = sid_2 = \cdots = sid_n$, $\mathsf{H_1}(trans_1) = \mathsf{H_1}(trans_2) = \cdots = \mathsf{H_1}(trans_n)$, $\mathsf{H_1}(K_1') = \mathsf{H_1}(K_2') = \cdots = \mathsf{H_1}(K_n')$, and every signature is valid. If all the verifications succeed, the TTP broadcasts $sid_1||\mathsf{H_1}(trans_i)||\mathsf{H_1}(K_1')||\sigma_{ttp}$, where

$$\sigma_{ttp} = \mathsf{Sign}(sid_1||\mathsf{H_1}(trans_i)||\mathsf{H_1}(K_1'), sk_{ttp}).$$

   Otherwise, the TTP broadcasts a failure message.

5. U_i verifies that σ_{ttp} is valid, $sid_1 = sid_i$, $\mathsf{H_1}(trans_1) = \mathsf{H_1}(trans_i)$, and $\mathsf{H_1}(K_1') = \mathsf{H_1}(K_i')$. If all these verifications succeed, then U_i computes the session key $K_i = \mathsf{H_2}(K_i'||sid_i)$ and set its session identifier to sid_i. If any of these verifications fail, or if U_i receives a failure message from the TTP, then U_i aborts the protocol.

The key establishment sub-protocol has four rounds, and each user needs to compute three exponentiations, sign one message and verify one signature, and the TTP needs to sign one message and verify n signatures.

7.3 First extension to the Burmester-Desmedt Protocol

In this section we propose a modified Burmester-Desmedt protocol and establish its security under the DBDH assumption in the security model described in Chapter 6.

7.3.1 Description of the Protocol

The TTP runs the TTP initialisation sub-protocol to generate an additive group $\mathbb{G}_1$ of prime order q, a multiplicative group $\mathbb{G}_T$ of order q, a generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$, and two hash functions H_1: $\{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^\ell$. The TTP also generates its public/private key pair, as used to certify users' public keys, and a public key $S = sP$, where s is randomly chosen from $\mathbb{Z}_q$. Every user U_i ($i \geq 1$) runs the user initialisation sub-protocol to generate a signature key pair (pk_i, sk_i), where $sk_i \in_R \mathbb{Z}_q$ and $pk_i = sk_i P$, for the aggregate signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ proposed by Boneh *et al.* [49], and gets pk_i certified by the TTP. Note that the signature scheme by Boneh *et al.* [49] is existentially unforgeable under a chosen message attack.

Suppose a set of users U_i ($1 \leq i \leq n$) wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i selects $s_i \in_R \mathbb{Z}_q$ and broadcasts $Z_i = s_i P$.
2. After receiving every Z_j ($1 \leq j \leq n, j \neq i$), U_i computes and broadcasts X_i, where
$$X_i = \hat{e}(S, s_i(Z_{i+1} - Z_{i-1}))$$
3. After receiving every X_j ($1 \leq j \leq n, j \neq i$), U_i computes and broadcasts the

signature $\sigma_i = sk_i \mathsf{H}_1(ID_i||sid_i||trans_i||M_i)$, where $sid_i = (UID, RSID)$,

$RSID = Z_1||Z_2||\cdots||Z_n||X_1||X_1||\cdots||X_n$, $trans_i = RSID$, and

$$\begin{aligned} M_i &= \hat{e}(S, ns_i Z_{i-1}) \prod_{\ell=1}^{n-1} X_{i+\ell-1}^{n-\ell} \\ &= \hat{e}(P,P)^{s\sum_{\ell=1}^{n}(s_\ell s_{\ell+1})}. \end{aligned}$$

4. After receiving every σ_j $(1 \leq j \leq n, j \neq i)$, U_i checks that:

$$\hat{e}(\sum_{j=1, j\neq i}^{n} \sigma_j, P) = \prod_{j=1, j\neq i}^{n} \hat{e}(h_j, pk_j), \text{ where}$$

$$h_j = \mathsf{H}_1(ID_j||sid_i||trans_i||M_i).$$

If this check succeeds, U_i computes the session key $K_i = \mathsf{H}_2(sid_i||M_i)$. Otherwise, U_i aborts the protocol execution.

During the protocol execution, every participant U_i $(1 \leq i \leq n)$ needs to compute 2 full-range multiplications[2] and $n + 4$ pairings. U_1 needs to compute one additional multiplication.

7.3.2 Security Analysis

We first analyse a bilinear version of the Burmester-Desmedt protocol, which has a similar structure to the Du-Wang-Ge-Wang protocol described in Section 5.2.2.1.

7.3.2.1 A Bilinear Version of the Burmester-Desmedt Protocol

The TTP runs the TTP initialisation sub-protocol to generate an additive group $\mathbb{G}_1$ of prime order q, a multiplicative group $\mathbb{G}_T$ of order q, a generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$, and two hash functions H_1: $\{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^\ell$. The TTP also generates a public key $S = sP$, where s is randomly chosen from $\mathbb{Z}_q$.

[2]full-range multiplication means that the coefficient is randomly selected.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i selects $s_i \in_R \mathbb{Z}_q$ and broadcasts $Z_i = s_i P$.
2. After receiving every Z_j $(1 \leq j \leq n, j \neq i)$, U_i computes and broadcasts X_i, where

$$\begin{aligned} X_i &= \hat{e}(S, s_i(Z_{i+1} - Z_{i-1})) \\ &= \frac{\hat{e}(S,P)^{s_i s_{i+1}}}{\hat{e}(S,P)^{s_i s_{i-1}}}. \end{aligned}$$

3. After receiving every X_j $(1 \leq j \leq n, j \neq i)$, U_i computes the session key as

$$\begin{aligned} K_i &= \hat{e}(S, n s_i Z_{i-1}) \prod_{\ell=1}^{n-1} X_{i+\ell-1}^{n-\ell} \\ &= \hat{e}(P,P)^{s \sum_{\ell=1}^{n} (s_\ell s_{\ell+1})}. \end{aligned}$$

U_i sets the session identifier to be $sid_i = (UID, RSID)$ where

$$RSID = Z_1 || Z_2 || \cdots || Z_n || X_1 || X_1 || \cdots || X_n.$$

We first recall the parallel version of the DBDH assumption as described in Chapter 2. On the input of $(\mathbb{G}_1, \mathbb{G}_T, q, \hat{e})$, an attacker can only distinguish between the following $2n$-tuples with a negligible advantage:

$$(sP, s_1P, s_2P, \cdots, s_nP, \hat{e}(P,P)^{ss_1s_2}, \hat{e}(P,P)^{ss_2s_3}, \cdots, \hat{e}(P,P)^{ss_ns_1})$$

and

$$(sP, s_1P, s_2P, \cdots, s_nP, \hat{e}(P,P)^{r_1}, \hat{e}(P,P)^{r_2}, \cdots, \hat{e}(P,P)^{r_n}),$$

where $s, s_1, s_2, \cdots, s_n, r_1, r_2, \cdots, r_n \in_R \mathbb{Z}_q$.

Lemma 7 *The key establishment protocol given in Section 7.3.2.1 achieves the key authentication property against passive attackers under the DBDH assumption.*

Proof. In a session, the transcript is

$$(sP, s_1P, s_2P, \cdots, s_nP, \frac{\hat{e}(P,P)^{ss_1s_2}}{\hat{e}(P,P)^{ss_ns_1}}, \frac{\hat{e}(P,P)^{ss_2s_3}}{\hat{e}(P,P)^{ss_1s_2}}, \cdots, \frac{\hat{e}(P,P)^{ss_ns_1}}{\hat{e}(P,P)^{ss_{n-1}s_n}}).$$

From the parallel version of the DBDH assumption, if the transcript is replaced with

$$(sP, s_1P, s_2P, \cdots, s_nP, \frac{\hat{e}(P,P)^{s(x+r_1)}}{\hat{e}(P,P)^{s(x+r_n)}}, \frac{\hat{e}(P,P)^{s(x+r_2)}}{\hat{e}(P,P)^{s(x+r_1)}}, \cdots, \frac{\hat{e}(P,P)^{s(x+r_n)}}{\hat{e}(P,P)^{s(x+r_{n-1})}}),$$

where $x, r_1, r_2, \cdots, r_n \in_R \mathbb{Z}_q$, the attacker can only distinguish between these two cases with a negligible advantage based on the DBDH asssumption. The session key is equal to $\hat{e}(P,P)^{s\sum_{\ell=1}^{n}(x+r_\ell)}$ in the second case, where x is uniformly distributed given the transcript. As a result, in the second case, the session key is uniformly distributed given the transcript, and an attacker can only distinguish the session key from a random string with advantage 0. Therefore, it is straightforward to verify that a passive attacker has only a negligible advantage in the attack game for key authentication from the DBDH assumption. The lemma now follows. □

Backward secrecy is also achieved against passive attackers because no long-term keys are involved. As in the case of Burmester-Desmedt protocol, if n is even then the key authentication security of this protocol can be established under the BDH assumption.

7.3.2.2 The Analysis

In the security model described in Chapter 6, we have the following result.

Theorem 8 *The key establishment protocol given in Section 7.3.1 is* $\mathsf{AKC}^{\dagger}$ *secure under the DBDH assumption in the random oracle model.*

Proof. The protocol given in Section 7.3.1 achieves key randomness given that H_2 is a random oracle. The protocol given in Section 7.3.1 achieves key control and DPUKS resilience against passive attackers, because the session key is computed as a function of M_i and the unique session identifier sid_i. It is clear that the protocol given in Section 7.3.1 is the protocol obtained by applying the compiler described

in Section 7.2.2.1 to the protocol described in Section 7.3.2.1. Note that we avoid hashing M_i in the computation of σ_i because the signature is based on a hash of M_i, and security will not be affected from the proof of Theorem 5. The theorem immediately follows from Theorem 5. □

7.4 Second extension to the Burmester-Desmedt Protocol

In this section we propose a modified Burmester-Desmedt protocol, which employs only a password as the authentication secret, and establish its security under the DDH and KE assumptions in the security model described in Chapter 6.

7.4.1 Description of the Protocol

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a shared secret session key. The user initialisation sub-protocol generates $(\mathbb{G}, q)$ where $\mathbb{G}$ is a multiplicative group with large prime order q, and three hash functions H_0, H_1, and H_2, where $\mathsf{H}_0 : \{0,1\}^* \rightarrow \mathbb{G}$, $\mathsf{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^\ell$, and $\mathsf{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^\ell$. In addition, this sub-protocol also establishes a shared password $\pi \in \mathcal{PW}$ among U_i $(i \geq 1)$, where $\mathcal{PW} = \{\pi^{(1)}, \pi^{(2)}, \cdots, \pi^{(m)}\}$.

The key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i computes $g = \mathsf{H}_0(\pi||\mathrm{UID}||x)$, where $x \geq 0$ is the smallest integer that makes g a generator of $\mathbb{G}$. U_i then chooses $s_i \in_R \mathbb{Z}_q$, and broadcasts $Z_i = g^{s_i}$.

2. After receiving every Z_j $(1 \leq j \leq n, j \neq i)$, U_i checks that none of them equals 1. If the check succeeds, U_i sets $Z = \mathrm{UID}||Z_1||Z_2||\cdots||Z_n$, and broadcasts $A_{i,i-1}$ and $A_{i,i+1}$, where

$$A_{i,i-1} = \mathsf{H}_1(i||i-1||Z||g^{s_{i-1}s_i}||g), \text{ and } A_{i,i+1} = \mathsf{H}_1(i||i+1||Z||g^{s_{i+1}s_i}||g).$$

3. After receiving every $A_{j,j-1}$ and $A_{j,j+1}$ $(1 \leq j \leq n, j \neq i)$, U_i verifies the received values of $A_{i-1,i}$ and $A_{i+1,i}$ by recomputing them using s_i and the stored values of Z_{i-1} and Z_{i+1}. If all the checks succeed, U_i broadcasts $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$. Otherwise, U_i aborts its protocol execution.

4. After receiving every X_j $(1 \leq j \leq n,\ j \neq i)$, U_i sets $sid_i = (\mathrm{UID}, \mathrm{RSID})$, and computes the keying material M_i and key confirmation message σ_i, where

$$\begin{aligned} M_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1}, \end{aligned}$$

$$\mathrm{RSID} = Z_1||Z_2||\cdots||Z_n||X_1||X_2||\cdots||X_n, \text{ and } \sigma_i = \mathsf{H}_1(i||ID_i||sid_i||M_i||g).$$

   U_i then broadcasts its key confirmation message σ_i.

5. After receiving every σ_j $(1 \leq j \leq n, j \neq i)$, U_i checks whether the following equation holds:

$$\sigma_j \stackrel{?}{=} \mathsf{H}_1(j||ID_j||sid_i||M_i||g).$$

   If all the checks succeed, U_i computes its session key as $K_i = \mathsf{H}_2(sid_i||M_i||g)$. Otherwise, U_i terminates the protocol execution as a failure.

7.4.2 Security Analysis

Lemma 9 *For the password-based key establishment protocol described in Section 7.4.1, an attacker has only a negligible advantage in the attack game for password guessing resilience if the resilience parameter ℓ_1 is equal to 2 (as defined in Section 6.2.2.8), under the DDH and KE assumptions in the random oracle model.*

Proof. According to the description in Section 6.2.2.8, an attacker can always simply make a guess (without any other computation) by outputting the most probable password $\pi^{(1)}$, and hence if the attacker follows such a strategy then it will succeed with probability p_1 but with advantage 0 under our definition. The only possible way of achieving a greater advantage is by testing a possible password to reduce the domain of possible passwords. According to the protocol specification, the testing can be done either through examining the distribution of Z_* and X_*, or through re-computing $A_{*,*}$, σ_*, and K_*.

We first show that, in any session, from Z_i and X_i $(1 \leq i \leq n)$ an attacker can only succeed in testing a possible password with a negligible advantage.

Claim 1 *In any session, if Z_i $(1 \leq i \leq n)$ are faithfully delivered, from Z_* and X_* an attacker can distinguish between π and π' $(\pi' \neq \pi)$ with a negligible advantage under the DDH assumption.*

Proof. Given $Z_i = g^{s_i}$ and $Z_{i+1} = g^{s_{i+1}}$, and $g^{s_i s_{i+1}}$, a possible password π' could be tested by comparing $g^{s_i s_{i+1}}$ with $g^{s_i s_{i+1} \log_{g'} g}$, where $g' = \mathsf{H}_0(\pi'||\mathrm{UID}||x')$ and $x' \geq 0$ is the smallest integer that makes g' a generator of $\mathbb{G}$. If these two values are equal then $\pi' = \pi$ holds with overwhelming probability; otherwise $\pi \neq \pi'$. Since H_0 is modelled as a random oracle, the testing is at least as hard as distinguishing between (g^a, g^b, g^{ab}) and (g^a, g^b, g^{abc}) where $a \in_R \mathbb{Z}_q$, $b \in_R \mathbb{Z}_q$, and c is a constant integer but randomly chosen from $\mathbb{Z}_q$. For any polynomial-time algorithm $\mathcal{B}$, let α_i, for $1 \leq i \leq 3$, be the probability that $\mathcal{B}$ outputs 1 when it takes input (g^a, g^b, g^{ab}), (g^a, g^b, g^{abc}), and (g^a, g^b, g^z), where $a \in_R \mathbb{Z}_q$, $b \in_R \mathbb{Z}_q$, $b \in_R \mathbb{Z}_q$, and c is a constant integer but randomly chosen from $\mathbb{Z}_q$. From the DDH assumption, $|\alpha_1 - \alpha_3|$ is negligible. From $\mathcal{B}$, a new DDH distinguisher $\mathcal{D}$ can be constructed as follows: Given an input (g^a, g^b, g^d) where $d = ab$ or $d \in_R \mathbb{Z}_q$, run $\mathcal{B}$ with input (g^a, g^b, g^{cd}) and output the response from $\mathcal{B}$. It is clear that $\mathcal{D}$ has the advantage $|\alpha_2 - \alpha_3|$, which is negligible from the DDH assumption. Hence, from the triangle inequality theorem, $|\alpha_1 - \alpha_2|$ is also negligible. Therefore, from $Z_i = g^{s_i}$ and $Z_{i+1} = g^{s_{i+1}}$, and $g^{s_i s_{i+1}}$, an attacker can only succeed in testing a possible password π' with a negligible advantage under the DDH assumption.

Because the parallel version of DDH is equivalent to the DDH (described in Section 2.1.3), it is straightforward to verify the attacker can only distinguish between the following tuples with a negligible advantage under the DDH assumption, where

$$(g^{s_1}, g^{s_2}, \cdots, g^{s_n}, g^{s_1 s_2}, g^{s_2 s_3}, \cdots, g^{s_n s_1}),$$

$$(g^{s_1}, g^{s_2}, \cdots, g^{s_n}, g^{s_1 s_2 c}, g^{s_2 s_3 c}, \cdots, g^{s_n s_1 c}),$$

and $s, s_1, s_2, \cdots, s_n \in_R \mathbb{Z}_q$ and c is a constant integer but randomly chosen from $\mathbb{Z}_q$. In our protocol, for $1 \leq i \leq n$, $X_i = \frac{g^{s_{i+1} s_i}}{g^{s_{i-1} s_i}}$. Therefore, if the attacker can succeed in testing a possible password with a non-negligible advantage, then it is

straightforward to construct an algorithm to distinguish the above two tuples. From the DDH assumption, the claim now follows. □

It is straightforward to verify that, if Z_* are faithfully delivered, even with Z_* and X_* from more than one session the attacker can only distinguish between π and π' with a negligible advantage.

We next focus on the case where the test is through re-computing $A_{*,*}$, σ_*, and K_*. For an attack game for password guessing resilience as defined in section 6.2.2.8, we study the relationships between password test attempts and the attacker's advantage in the various possible cases.

- The first case is where the attacker faithfully delivers the messages for all oracles, i.e., the attacker is passive. Because H_k $(1 \leq k \leq 3)$ are modelled as random oracles, in order to test a possible password, the attacker needs to compute either $g^{s_{j-1}s_j}$ or M_j for some $j \geq 1$ in some session, and then to re-compute some $A_{*,*}$, σ_*, or K_*. Note that, if the attacker can compute $g^{s_{j-1}s_j}$ or M_j, then it can compute the corresponding session key (and we can construct a passive attacker which wins the attack game for key authentication of the Burmester-Desmedt protocol). In this case, from the DDH assumption, it follows that the attacker has only a negligible probability of successfully testing a possible password.

- The second case is where the attacker faithfully delivers the messages for all oracles in the first step of all protocol executions. We consider the following two possibilities.

  1. Suppose that an oracle Π_j^{t,sid_j} $(j \geq 1)$ computes the value M_j in the fourth step of its protocol execution, where M_j is computed as a function of Z_{j-1}, s_j, X_j and some other public values (namely $X_{j+1}, X_{j+2}, \cdots, X_{j-2}$). The attacker can compute M_j with only a negligible probability from the DDH assumption; otherwise, it is straightforward to construct a passive attacker (with non-negligible advantage) for key authentication of the Burmester-Desmedt protocol. Hence, the attacker has only a negligible probability of successfully testing a possible password from the messages

σ_* and K_*.

2. For any oracle Π_j^{t,sid_j} $(j \geq 1)$, if the attacker can compute $g^{s_{j-1}s_j}$ in a session then it can also compute M_j. Hence, from the above analysis, the attacker has only a negligible probability of successfully testing a possible password from the messages $A_{*,*}$.

- The third case is where the attacker can manipute all protocol messages, including those sent in the first step of protocol executions. We consider the following two possibilities.

  1. We first consider a simple case. Suppose that, in some session, $\mathcal{A}$ replaces Z_{j+1} sent to Π_j^{t,sid_j} with Z'_{j+1}, where $Z'_{j+1} = (g')^s$, g' is computed based on a guessed password π', and s is chosen by $\mathcal{A}$. Without loss of generality, suppose that $\mathcal{A}$ postpones forging $A_{j+1,j}$ until it obtains $A_{j,j+1}$. With $A_{j,j+1}$, $\mathcal{A}$ can test the guessed password by checking whether the following equation holds:

$$A_{j,j+1} \stackrel{?}{=} \mathsf{H}_1(j||j+1||Z'||(Z_j)^s||g'), \text{ where}$$

$$Z' = UID||Z_1||Z_2||\cdots||Z_j||Z'_{j+1}||Z_{j+2}||\cdots||Z_n.$$

  Claim 2 *If $\pi' \neq \pi$, $\mathcal{A}$ can only succeed in testing a different password π'' with a negligible probability, and Π_j^{t,sid_j} accepts with a negligible probability based on the DDH assumption.*

  Proof. If $\mathcal{A}$ wishes to test another possible password π'', then it needs to compute

$$\begin{aligned}(Z'_{j+1})^{\log_{g''} Z_j} &= (g')^{s \log_{g''} g' \log_{g'} Z_j} \\ &= ((g')^{\log_{g''} g'} (g')^{\log_{g'} Z_j})^s \\ &= ((g')^{(\log_{g'} g'')^{-1}} (g')^{\log_{g'} Z_j})^s,\end{aligned}$$

  where g'' is computed as a function of π''. Since we asssumed that H_0 behaves as a random oracle, g'' is a random element from $\mathbb{G}$, and Z_j is also a random element since it is generated by the challenger. Hence, if the attacker can compute $(Z'_{j+1})^{\log_{g''} Z_j}$ from g'' and Z_j, then it is straightforward to construct an algorithm to solve the DDH problem (since $g^a = g^{(\log_g g^b)^{-1}} g^{\log_g g^{ab}}$ for any a, b). As a result, $\mathcal{A}$ can only succeed in this test with a negligible probability.

If the attacker's guess is wrong ($\pi \neq \pi'$), the attacker needs to forge a key authentication message σ_{j+1} for Π_j^{t,sid_j} to make this oracle accept. To do this, the attacker needs to compute the value $(Z'_{j+1})^{\log_g Z_j}$, where g is computed as a function of π. From the above analysis, $\mathcal{A}$ can only compute a forgery with a negligible probability, and, as a result, Π_j^{t,sid_j} accepts with a negligible probability. □

2. We then consider a general case, where $\mathcal{A}$ replaces Z_{j+1} with some $Z'_{j+1} \in \mathbb{G}$, where Z'_{j+1} is not generated by any oracle and may be generated by the attacker by any method. Given $A_{j,j+1}$, if $\mathcal{A}$ can test a guessed password π' with probability δ_1, then it implies that $\mathcal{A}$ can compute $(Z'_{j+1})^{\log_{g'} Z_j}$ with probability δ_1, where g' is computed as a function of π'. From the KE assumption (as described in Section 2.1.3), if the attacker can compute $(Z'_{j+1})^{\log_{g'} Z_j}$ then it knows $\log_{g'} Z'_{j+1}$. From Claim 2, if δ_1 is non-negligible then the attacker can test another possible password with a negligible probability. From the analysis in the simple case, and we can conclude that $\mathcal{A}$ can only succeed in testing at most one password π', and if $\pi' \neq \pi$ then Π_j^{t,sid_j} accepts with a negligible probability based on the DDH and KE assumptions.

For an oracle Π_j^{t,sid_j}, besides modifying Z_{j+1}, the attacker can also modify the value of Z_{i-1} sent to this oracle. Using the same method as described above, it is straightforward to verify that the attacker can test only one password by modifying this message, and Π_j^{t,sid_j} will abort if the guessed password is wrong. Therefore, by intervening in the inputs to Π_j^{t,sid_j}, the attacker can test at most two possible passwords through re-computing $A_{*,*}$, σ_*, and K_*.

In this case if the attacker has not guessed the correct password, then all X_* received by the attacker are generated based on faithfully delivered Z_*. Therefore, from Claim 1, it is straightforward to verify that the attacker can test a possible password through examining the distribution of Z_* and X_*.

The above cases cover all the possible means by which the attacker might try to test a possible password. We have shown that, by modifying the input to one oracle, the attacker can only test two guessed passwords, and the oracle will abort with an overwhelming probability if either of these guessed passwords is wrong. If there are

n_1 aborted oracles at the end of the attack game, the adversary can test at most $2n_1$ possible passwords. As a result, at the end of the game, the attacker can guess the password with the probability $\delta = \sum_{j=1}^{2n_1+1} p_j + \epsilon$ where ϵ is negligible from the DDH and KE assumptions, and the attacker's advantage $\mathcal{F}(2n_1, \delta)$ is negligible. The lemma now follows. □

Theorem 10 *The password-based key establishment protocol given in Section 7.4.1 is* 2-$\mathsf{PAKC}^{\dagger}$ *secure under the DDH assumption in the random oracle model.*

Proof. From the definitions, it is straightforward to verify that key randomness and key control are guaranteed. Password guessing resilience was established in Lemma 9. We next prove that an active attacker has only a negligible advantage in the attack games for key confirmation (password-based) and key authentication (password-based) if the resilience parameter ℓ_1 is equal to 2.

Consider an attack game for key confirmation (password-based), as described in Section 6.2.2.11, for the proposed protocol. It is clear that an attacker $\mathcal{A}$ has identical privileges to those given to the attacker in the attack game for password guessing resilience as described in Section 6.2.2.8. Therefore, from Lemma 9, $\mathcal{A}$ can guess the password with probability $\sum_{j=1}^{2n_1+1} p_j$ if there are n_1 aborted oracles. If Π_j^{t,sid_j} is the tested oracle, then let E_1 be the event that, for some x $(1 \leq x \leq n, x \neq j)$, there is no oracle Π_x^{y,sid_x} which satisfies $sid_x = sid_j$ and possesses the same session key as that of Π_j^{t,sid_j}, and the attacker has forged the message σ_x. Since H_1 is modelled as a random oracle, it is clear that the forgery will succeed with probability $\sum_{j=1}^{2n_1+1} p_j + \epsilon$ and $\Pr[E_1] = \sum_{j=1}^{2n_1+1} p_j + \epsilon$, where ϵ is negligible. As a result, key confirmation (password-based) is guaranteed under the DDH and KE assumptions in the random oracle model.

Consider the attack game for key authentication (password-based) described in Section 6.2.2.9. It is clear that an attacker $\mathcal{A}_1$ has less privileges than the attacker in the attack game for key confirmation (password-based) given in Section 6.2.2.11. Therefore, from the above analysis, $\mathcal{A}$ can guess the password with probability at most $\sum_{j=1}^{2n_1+1} p_j$ in the first phase, given that there are n_1 aborted oracles at the

end of the game. If Π_j^{t,sid_j} is the tested oracle, then let E_2 be the event that Π_j^{t,sid_j} has $n-1$ partner oracles. Then the maximum of $\{0, 1 - \Pr[E_2] - \sum_{j=1}^{2n_1+1} p_j\}$ is negligible because the protocol achieves key confirmation (password-based). If E_2 occurs, $|\Pr[b' = b] - \frac{1}{2}|$ is negligible (otherwise, it is straightforward to construct an algorithm to solve the DDH problem). If E_2 does not occur, then we assume $\Pr[b' = b] = 1$ (the best the attacker can achieve). The probability $b' = b$ can be computed as follows.

$$\begin{aligned}
\Pr[b' = b] &\leq \Pr[E_2](\frac{1}{2} + \epsilon_1) + 1 - \Pr[E_2] \\
&= (1 - \sum_{j=1}^{2n_1+1} p_j - \epsilon_2)(\frac{1}{2} + \epsilon_1 - 1) + 1 \\
&\leq \frac{1 + \sum_{j=1}^{2n_1+1} p_j}{2} + \epsilon_1 + \epsilon_2,
\end{aligned}$$

where ϵ_1 and ϵ_2 are negligible. As a result, it is clear that $max\{0, \Pr[b' = b] - \frac{1+\sum_{j=1}^{2n_1+1} p_j}{2}\}$ is negligible, and key authentication (password-based) is guaranteed under the DDH and KE assumptions in the random oracle model. □

7.5 Third extension to the Burmester-Desmedt Protocol

In this section we modify the password-based protocol described in Section 7.4.1 using CAPTCHA techniques, and establish its robustness against password guessing attacks. We make the following heuristic assumption about the security of the CAPCHA tests in use. Suppose a new CAPTCHA test pz is sent to an entity, where pz has not been sent to this entity before. Then, if the entity gives a valid response, then we can assume with probability close to 1 that the entity is a human being.

7.5.1 Description of the Protocol

The TTP runs the TTP initialisation sub-protocol to generate a verify/sign signature key pair (pk_{ttp}, sk_{ttp}) for a signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, and a key pair (pk'_{ttp}, sk'_{ttp}) for a public-key encryption key scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. In addition,

this sub-protocol also generates the parameters for the generation and verification of CAPTCHA tests [12]. The user initialisation sub-protocol is identical to that described in Section 7.4.1.

Suppose a set of users U_i $(1 \leq i \leq n)$ wish to establish a shared secret session key; the key establishment sub-protocol is as follows. Note that the indices of users (and values exchanged between users) are taken modulo n.

1. U_i chooses $r_i \in_R \{0,1\}^\ell$ and sends it to the TTP.
2. The TTP generates a new CAPTCHA test pz and sends it to U_1.
3. U_1 solves pz and sends the solution, sz say, back to the TTP.
4. The TTP checks whether sz is the correct solution for pz. If the check succeeds, it chooses $S \in_R \{0,1\}^\ell$, and then broadcasts $(S, R, \sigma_{ttp,1})$, where $R = r_1||r_2||\cdots||r_n$ and $\sigma_{ttp,1} = \mathsf{Sign}(\mathrm{UID}||R||S, sk_{ttp})$.
5. After receiving $(S, R, \sigma_{ttp,1})$ from the TTP, U_i first verifies the signature. If the verification succeeds, U_i continues; otherwise, U_i aborts the protocol execution. U_i chooses $s_i \in_R \mathbb{Z}_q$, and broadcasts $Z_i = g^{s_i}$, where $g = \mathsf{H}_0(\pi||\mathrm{UID}||x)$ and $x \geq 0$ is the smallest integer that makes g a generator of $\mathbb{G}$..
6. After receiving every Z_j $(1 \leq j \leq n, j \neq i)$, U_i checks that none of them equals 1. If all the checks succeed, U_i chooses $r_i', r_i'' \in_R \{0,1\}^\ell$, and sends

   $$E_i = \mathsf{Enc}(ID_i||A_{i,i-1}||A_{i,i+1}||A_{i-1,i}||A_{i+1,i}||r_i'||r_i''||S, pk_{ttp}')$$

   to the TTP, where $A_{i,i-1}$, $A_{i,i+1}$, $A_{i-1,i}$, and $A_{i,i+1}$ are defined in the same way as in the extended protocol specified in Section 7.4.1.
7. After receiving E_i, for every $1 \leq i \leq n$, the TTP first decrypts them, and checks that all values of S are identical and exist in its memory, and then checks whether $A_{i,i+1}$ (provided by U_i and U_{i+1}) and $A_{i,i-1}$ (provided by U_{i-1} and U_i) are consistent, for all $1 \leq i \leq n$. If these checks succeed, TTP broadcasts a message $(R', \sigma_{ttp,2})$, where $Z = \mathrm{UID}||Z_1||Z_2||\cdots||Z_n$, $R' = r_1'||r_2'||\cdots||r_n'$, and $\sigma_{ttp,2} = \mathsf{Sign}(Z||S||R', sk_{ttp})$. Otherwise TTP broadcasts a failure message. Simultaneously, the TTP erases S and the relevant information.

8. If it receives a failure message, U_i aborts the protocol execution. Otherwise, U_i verifies the signature $\sigma_{ttp,2}$ and checks that R' contains r'_i. If the verification succeeds, U_i broadcasts $X_i = (\frac{Z_{i+1}}{Z_{i-1}})^{s_i}$.

9. After receiving every X_j $(1 \leq j \leq n, j \neq i)$, U_i sets $sid_i = (\mathrm{UID}, \mathrm{RSID})$, and computes the keying material M_i and key confirmation message σ_i, where

$$\begin{aligned} M_i &= (Z_{i-1})^{ns_i} \cdot (X_i)^{n-1} \cdot (X_{i+1})^{n-2} \cdots X_{i+n-2} \\ &= g^{s_1 s_2 + s_2 s_3 + s_3 s_4 + \cdots + s_n s_1}, \end{aligned}$$

   $\mathrm{RSID} = S||Z_1||Z_2||\cdots||Z_n||X_1||X_2||\cdots||X_n$, and $\sigma_i = \mathsf{H}_1(i||ID_i||sid_i||M_i||g)$.

   U_i then broadcasts its key confirmation message σ_i.

10. After receiving every σ_j $(1 \leq j \leq n, j \neq i)$, U_i checks whether the following equation holds:

$$\sigma_j \stackrel{?}{=} \mathsf{H}_1(j||ID_j||sid_i||M_i||g).$$

    If all the checks succeed, U_i computes its session key as $K_i = \mathsf{H}_2(sid_i||M_i||g)$. Otherwise, U_i terminates the protocol execution as a failure.

7.5.2 Security Analysis

Lemma 11 *For the password-based key establishment protocol described in Section 7.5.1, an attacker has only a negligible advantage in the attack game for password guessing resilience, if the resilience parameter ℓ_1 is equal to 1 (as defined in Section 6.2.2.8), under the DDH assumption in the random oracle model.*

Proof. As in the description in Section 6.2.2.8, an attacker can always simply make a guess (without any other computation) by outputting the most probable password $\pi^{(1)}$, and hence will succeed in an attack with probability p_1 but with advantage 0 under our definition. The only possible way of achieving a greater advantage is by testing a possible password to reduce the domain of possible passwords. According to the protocol specification, the testing can be done either through examining the distribution of Z_* and X_*, or through re-computing $A_{*,*}$, σ_*, and K_*.

We first show that, in any session, from Z_i and X_i $(1 \leq i \leq n)$ an attacker can only succeed in testing a possible password with a negligible advantage. The security proof of the following claim is identical to that of Claim 1 in the proof of Lemma 9.

Claim 3 *In any session, if Z_i $(1 \leq i \leq n)$ are faithfully delivered, from Z_* and X_* a attacker can distinguish between π and π' $(\pi' \neq \pi)$ with a negligible advantage under the DDH assumption.*

It is straightforward to verify that, if Z_* are faithfully delivered, even with Z_* and X_* from more than one session the attacker can only distinguish between π and π' with a negligible advantage.

We next focus on the case where the test is through re-computing $A_{*,*}$, σ_*, and K_*. For an attack game for password guessing resilience as defined in Section 6.2.2.8, we study the relationships between password test attempts and the attacker's advantage in the following cases.

- The first case is where the attacker faithfully delivers the messages for all oracles. As in the first case of the proof of Lemma 9, it is straightforward to verify that the attacker has only a negligible probability of successfully testing a possible password from the DDH assumption. It is also clear that, even if it is given the decryption key sk'_{ttp}, the attacker can only succeed with a negligible probability.
- The second case is where the attacker faithfully delivers the messages for all oracles in step 5 of the protocol executions. For the same reason as in the second case of the proof of Lemma 9, the attacker has only a negligible probability of successfully testing a possible password from the DDH assumption. As in the first case, even if it is given the decryption key sk'_{ttp}, the attacker can only succeed with a negligible probability.
- The third case is where the attacker can manipute all protocol messages, including those sent in step 5 of all the protocol executions. Suppose that, in step 5 of a protocol execution, $\mathcal{A}$ replaces Z_{j-1} sent to Π_j^{t,sid_j} with Z'_{j-1}. We consider the following possibilities.

– After receiving E_j from Π_j^{t,sid_j}, the attacker directly re-computes $A_{j,j-1}$ or $A_{j-1,j}$ and compares them with those in E_j. Because the public-key encryption scheme is IND-CCA2 secure, the attacker has only a negligible probability of successfully testing a possible password.

– The attacker forges E'_{j-1} and submits E_j and E'_{j-1} to the TTP as a test. If the guess is wrong (which means that the values of $A_{j,j-1}$ and $A_{j-1,j}$ are different in E_j and E'_{j-1}), then it is straightforward to verify that Π_j^{t,sid_j} will abort in step 8 with an overwhelming probability, since S is randomly chosen and the signature scheme is existentially unforgeable (the attacker can only forge a signature with negligible probability).

  If the guessed password is wrong, the attacker may try to test another possible password as in the third case of the proof of Lemma 9. Since S will be discarded after verifications of E_i $(1 \leq i \leq n)$, the attacker needs to forge E_j^*, where S' is another identifier generated by the TTP,

$$E_j^* = \mathsf{Enc}(ID_j||Info||S', pk'_{ttp}),$$

  and $Info$ is some information determined by $A_{j,j+1}$ and $A_{j+1,j}$, where the attacker knows how to compute $Info$ from $A_{j,j+1}$ and $A_{j+1,j}$[3]. Then the attacker submits E_{j-1}^* and E_j^* for another test, where E_{j-1}^* is computed as a function of the new testing password. It is clear that the attacker can only succeed the forgery of E_j^* with a negligible probability because the public-key encryption scheme is IND-CCA2 secure (so that the probability of forging E_j^* is negligible).

– There is another possibility that the attacker $\mathcal{A}$ may simultaneously replace Z_{j-1} and Z_{j+1} sent to Π_j^{t,sid_j} with Z'_{j-1} and Z'_{j+1} which are computed based on two possible passwords. From the above analysis, it is straightforward to verify that the TTP will return a failure with an overwhelming probability so that Π_j^{t,sid_j} will abort. As a result, the attacker obtains no information regarding which guess is wrong. Note that, in the password-based protocol described in Section 7.4.1, the attacker can simultaneously test two possible passwords by modifying Z_{j-1} and Z_{j+1}.

In summary we have shown that, through re-computing $A_{*,*}$, σ_*, and K_*, the attacker can test only one password by modifying the input to an oracle

[3]The attacker can only test its guess if it knows the the computation of $Info$ based on $A_{j,j+1}$ and $A_{j+1,j}$.

Π_j^{t,sid_j}, and that this oracle will abort with an overwhelming probability if the guessed password is wrong. If an oracle Π_j^{t,sid_j} $(j \geq 1)$ does not abort in step 8, then either the attacker has successfully guessed the password or the values Z_{j-1} and Z_{j+1} received by this oracle are faithfully generated without being modified by the attacker.

In this case if the attacker has not guessed the correct password, then all X_* received by the attacker are generated based on faithfully delivered Z_*. Therefore, from Claim 3, it is straightforward to verify that the attacker can test a possible passowrd through examining the distribution of Z_* and X_*.

The above cases cover all the possible means by which the attacker might try to test a possible password. We have shown that, by modifying the input to one oracle, the attacker can only test one guessed password, and the oracle will abort with an overwhelming probability if the guessed password is wrong. If there are n_1 aborted oracles at the end of the attack game, the adversary can test at most n_1 possible passwords. As a result, at the end of the game, the attacker can guess the password with probability at most $\delta = \sum_{j=1}^{n_1+1} p_j + \epsilon$, where ϵ is negligible, and the attacker's advantage $\mathcal{F}(n_1, \delta)$ is negligible. The lemma now follows. □

Theorem 12 *The password-based key establishment protocol given in Section 7.5.1 is* 1-PAKC† *secure under the DDH assumption in the random oracle model.*

Proof. From the definitions, it is straightforward to verify that key randomness and key control are guaranteed. Password guessing resilience was established in Lemma 11. Using the same method as employed in the proof of Theorem 10, it is straightforward to verify that key confirmation (password-based) and key authentication (password-based) are guaranteed for the case where the resilience parameter ℓ_1 is equal to 1. The theorem then follows. □

Compared with the protocol given in Section 7.4.1, the protocol described in Section 7.5.1 has two different features. One is that it is necessary to solve a puzzle at the

beginning of every session, and the other is that the TTP verifies $A_{*,*}$ for all oracles in every session.

Assuming the hardness of the CAPTCHA, the possibilities for automated online password guessing attacks are now significantly reduced, since we assume that only a human being is able to compute the correct response to the CAPTCHA. In other words, we are assured that the initiator of the protocol is actually a human being, and not just a computer program. Moreover, assuming the security of the underlying signature scheme, we can verify that U_1 has successfully solved a CAPTCHA if, and only if, we receive a valid signature. Therefore, an automated system is prevented from mounting an automated online password guessing attack if the underlying CAPTCHA used is sufficiently strong. Another alternative (without using a CAPTCHA) is to require the initiator to solve a computational puzzle [140] prior to a protocol execution with each protocol participant. However, such an approach is computationally expensive, and it is difficult to adjust the hardness of the puzzles if the participants have very different computing power.

In any session following a successful puzzle solution, the federated verification of $A_{*,*}$ ensures that $\mathcal{A}$ can only test at most one password. Assuming the security of the underlying public-key encryption scheme, $\mathcal{A}$ is unable to recover the encrypted messages or try its guess with the encrypted messages. In order to test a candidate password, $\mathcal{A}$ must submit forged messages to S for verification. However, if the attacker tests two or more passwords then the verification step by S will fail, and this failure will provide no information about which tested password is wrong. The participants can indirectly verify the authentication messages by verifying the signature from S, because of the presence of the value r_i' in the signature. However, they cannot identify which authentication message has gone wrong in the event that the verification fails.

7.6 Conclusions

In this chapter we proposed two new compilers which transform key establishment protocols secure against passive attackers into key establishment protocols secure against active attackers. We further proposed three extensions of the Burmester-

Desmedt protocol and proved their security in the security model defined in Chapter 6.

Part II
Timed-Release Encryption Schemes

Chapter 8

Security Definitions for TRE-PC Schemes

Contents

In this chapter we investigate the Hwang-Yum-Lee model for Timed-Release Encryption schemes with Pre-open Capability (TRE-PC schemes), and show that this model possesses a number of defects, and fails to model certain potentially practical security vulnerabilities faced by TRE-PC schemes. We then propose a new security model for TRE-PC schemes which models security against four types of attacker, and avoids the defects of the Hwang-Yum-Lee model. Finally, we establish the relationships between the security notions defined in the new model.

8.1 Motivation

The concept of Timed-Release Encryption (TRE), i.e. sending a message which can only be decrypted after a certain release time, is due to May [186]. Subsequently, Rivest, Shamir, and Wagner elaborated the concept and gave a number of possible applications, including electronic auctions, key escrow, chess moves, release of documents over time, payment schedules, press releases, etc. [212]. Informally, a TRE scheme is a public-key encryption scheme which achieves the following two goals:

1. An outside attacker cannot decrypt the ciphertext. An outside attacker is any entity other than the sender and intended receiver.
2. A legitimate receiver can only decrypt the ciphertext after the release time.

It is worth mentioning that other timed primitives have been developed, for example, "price via processing" scheme of Dwork and Naor [100], timed key escrow, as proposed by Bellare and Goldwasser [24, 29], and timed commitments, due to Boneh and Naor [50].

In the literature, two approaches have been used to construct TRE schemes. The first approach is based on the time-lock puzzle technique, originally proposed by Merkle [191] to protect communications against passive attackers. This technique was then extended in [26, 71, 212] to construct TRE schemes. The idea of this approach is that a secret is transformed in such a way that any computing device (serial or parallel) will take at least a certain amount of time to solve the underlying computational problem (puzzle) in order to recover the secret. The release time is equal to the time at which the puzzle is released plus the minimum amount of time that it would take to solve the puzzle. However, this means that not all users are capable of decrypting the ciphertext at the release time, as they may have different computational power. The other approach is to use a trusted time server, which, at an appointed time, will assist in releasing a secret to help decrypt the ciphertext (see, for example [70, 212]). Generally, time-server-based schemes require interaction between the server and the users, and measures need to be put in place to prevent possible malicious behaviour by the time server.

8.1 Motivation

In a standard Timed-Release Encryption (TRE) scheme, the receiver can only decrypt the ciphertext at (or after) the release time. If the sender changes its mind after the ciphertext is sent, and wishes the receiver to decrypt the message immediately, then the only thing that the sender can do is to re-send the plaintext to the receiver in such a way that the receiver can immediately decrypt the message. However, in some circumstances, we may need a special kind of TRE scheme, in which a mechanism enables the receiver to decrypt the ciphertext before the release time without requiring the sender to re-send the plaintext. Recently, Hwang, Yum, and Lee [124] extended the concept of TRE scheme and proposed a security model (referred to as the Hwang-Yum-Lee model) for TRE schemes with Pre-Open Capability (TRE-PC schemes). Informally, a TRE-PC scheme is a public-key encryption scheme which achieves the following goals:

1. At any time, an outside attacker cannot decrypt the ciphertext.
2. At (or after) the release time, when the plaintext is intended to be released, the receiver can decrypt the ciphertext. However, before the release time, the receiver cannot decrypt the ciphertext.
3. Before the release time, the sender can enable the receiver to decrypt the ciphertext by publishing a pre-open key.

We show that the Hwang-Yum-Lee model possesses a number of defects, and fails to model certain potentially practical security vulnerabilities faced by TRE-PC schemes.

The rest of this chapter is organised as follows. In Section 8.2 we review the Hwang-Yum-Lee model. In Section 8.3, we propose a new security model for TRE-PC schemes which models security against four types of attacker and avoids the defects of the Hwang-Yum-Lee model. In Section 8.4, we establish the complete relationships between the security notions defined in the new model. In the final section we conclude this chapter.

8.2 Review of the Hwang-Yum-Lee model and schemes

8.2.1 Description of the Hwang-Yum-Lee model

In the Hwang-Yum-Lee model, two types of entities are involved in a TRE-PC scheme: a trusted time server and users. The trusted time server periodically publishes timestamps, and every user may act as either a sender or a receiver. The following two types of attacker are also considered:

- An outside attacker without the receiver's private key, which models either a dishonest time server, or an eavesdropper who tries to decrypt the legal receiver's ciphertext.
- An inside attacker with the receiver's private key, which models a legal receiver who tries to decrypt the ciphertext before the release time without the pre-open key.

In the Hwang-Yum-Lee model, a TRE-PC scheme consists of the following six polynomial-time algorithms:

- Setup: The setup algorithm takes a security parameter ℓ as input, and returns the master key mk and the system parameters $param$, where mk is kept secret by the time server and $param$ is published.
- $\mathsf{Ext}_{\mathsf{TS}}$: Run by the trusted time server, this timestamp extraction algorithm takes the system parameters $param$, the master key mk, and the release time t as input, and returns the timestamp TS_t. The time server is required to publish TS_t at time t.
- $\mathsf{Gen}_{\mathsf{PK}}$: Run by a user, this key generation algorithm takes ℓ as input, and the system parameters $param$ as input, and returns a public/private key pair (pk_r, sk_r).
- Enc: Run by a sender, this encryption algorithm takes a message m, a release time t, and a randomly-chosen pre-open secret value v, and returns a ciphertext C.

- $\mathsf{Gen_{RK}}$: Run by a sender, this pre-open key generation algorithm takes the pre-open secret value v and the release time t as input, and returns the pre-open key V_t.
- Dec: Run by a receiver, this decryption algorithm runs in two modes. Before the release time, on input of the pre-open key V_t, the ciphertext C, and the receiver's private key sk_r, this algorithm returns either the plaintext or an error message. Otherwise, on the input of the timestamp TS_t, the ciphertext C, and the receiver's private key sk_r, this algorithm returns either the plaintext or an error message.

In the Hwang-Yum-Lee model, three semantic security properties are modelled, although we show later that certain other security properties (see Section 3) also need to be considered. The modelled semantic security properties are: security under a chosen ciphertext attack against outside attackers (IND-TR-CCA$_{\mathrm{OS}}$ security), security under a chosen plaintext attack against outside attackers (IND-TR-CPA$_{\mathrm{OS}}$ security), and security under a chosen ciphertext attack against inside attackers (IND-TR-CCA$_{\mathrm{IS}}$ security).

Definition 34 *Suppose $\mathcal{A}$ is a polynomial-time outside attacker; then a TRE-PC scheme E is IND-TR-CCA$_{OS}$ secure if $\mathcal{A}$ only has a negligible advantage in the following game.*

1. Game setup: The challenger takes a security parameter ℓ as input, and runs Setup to generate the master key mk and the system parameters $param$. The challenger also runs $\mathsf{Gen_{PK}}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker $\mathcal{A}$ executes with input $(pk_r, param)$, and can make the following queries.
   - $\mathcal{A}$ makes timestamp extraction queries for any time t. On receiving a query, the challenger runs $\mathsf{Ext_{TS}}$ and returns the output.
   - $\mathcal{A}$ makes decryption queries on (t, C). On receiving a query, the challenger runs Dec and returns the output.

3. Challenge: $\mathcal{A}$ selects two equal length messages m_0, m_1 and a release time t^*. The challenger picks a random bit b, encrypts m_b for release at time t^*, and returns the ciphertext C^*.

4. Phase 2: $\mathcal{A}$ can continue to make extraction and decryption queries as in Phase 1. However, $\mathcal{A}$ is not allowed to make a decryption query on (t^*, C^*).

5. Guess: $\mathcal{A}$ outputs a guess bit b'.

In this game the attacker wins the game if $b = b'$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 35 *A TRE-PC scheme E is said to be IND-TR-CPA$_{OS}$ secure if it is IND-TR-CCA$_{OS}$ secure against attackers that make no decryption queries.*

In [125] the attack game for IND-TR-CCA$_{\mathrm{IS}}$ security is informally defined; however, we can reconstruct a formal definition as follows.

Definition 36 *Suppose $\mathcal{A}$ is a polynomial-time inside attacker; then a TRE-PC scheme E is IND-TR-CCA$_{IS}$ secure if $\mathcal{A}$ only has a negligible advantage in the following game.*

1. Game setup: The challenger takes a security parameter ℓ as input, and runs Setup to generate the master key mk and the system parameters $param$. The challenger runs $\mathsf{Gen_{PK}}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}$ executes with input $(pk_r, sk_r, param)$, and can make timestamp extraction queries for any time t. On receiving a query, the challenger runs $\mathsf{Ext_{TS}}$ and returns the output.

3. Challenge: $\mathcal{A}$ selects two equal length messages m_0 and m_1, and a release time t^* which has not been queried to the $\mathsf{Ext_{TS}}$ oracle. The challenger picks a random bit b, encrypts m_b for release at time t^*, and returns the ciphertext C^*.

4. Phase 2: $\mathcal{A}$ continues to make the same types of queries as in Phase 1. However, $\mathcal{A}$ is not allowed to make an $\mathsf{Ext}_{\mathsf{TS}}$ query for the time t^*.

5. Guess: $\mathcal{A}$ terminates by outputting a guess bit b'.

In this game the attacker wins the game if $b = b'$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

8.2.2 Remarks on the Hwang-Yum-Lee model

We have the following remarks regarding the Hwang-Yum-Lee model.

1. In the Hwang-Yum-Lee model, the decryption process is described by a single algorithm, which, however, works in two different modes depending on the input. It would be more appropriate to formalise the decryption process as two independent algorithms.

2. In a security model for TRE-PC schemes, it should be assumed that an outside attacker will have access to the pre-open key for the challenge ciphertext. This models the real-world situation where the outside attacker observes the release key as it is being sent to the receiver, after the sender chooses to allow the receiver to pre-open the ciphertext. However, in the Hwang-Yum-Lee model, it is not explicitly specified that the attacker has access to the pre-open key.

3. In the Hwang-Yum-Lee model, $\mathsf{Gen}_{\mathsf{RK}}$ is formalised but never used in the security model. One possible way of eliminating this defect is to allow the attacker to access the $\mathsf{Gen}_{\mathsf{RK}}$ oracle (as a result the above defect can also be eliminated). Alternatively, we can incorporate the functionality of $\mathsf{Gen}_{\mathsf{RK}}$ into Enc, by requiring the latter to output both the ciphertext and the pre-open key (as in the new model given in Section 8.3).

4. In the Hwang-Yum-Lee model, the Enc algorithm does not take the receiver's public key as input; however, it is clear that the receiver's public key should be part of the input. This is probably just a typographical error.

5. In the attack game for IND-TR-CCA$_{\mathrm{IS}}$ security, the attacker $\mathcal{A}$ is allowed to make $\mathsf{Ext_{TS}}$ queries for any release time t except t^*. However, in reality a receiver only needs to mount this attack before the release time t^*, since it will be able to decrypt the ciphertext with the timestamp TS_{t^*} at (and after) t^*. Therefore, the attacker $\mathcal{A}$ should only be allowed to make $\mathsf{Ext_{TS}}$ queries for any time t earlier than the release time t^*.

6. It is claimed that the outside attacker models either a dishonest time server or an eavesdropper which tries to decrypt the legal receiver's ciphertext. However, a malicious time server has not been considered in the attack game which is used to evaluate IND-TR-CCA$_{\mathrm{OS}}$ security, because the attacker has no knowledge of the master key mk. As a result, a TRE-PC scheme which is proved IND-TR-CCA$_{\mathrm{OS}}$ secure, might be insecure against a malicious attacker which knows the time server's secret key.

7. One of the objectives of a TRE-PC scheme is to allow the sender to enable the receiver to decrypt a ciphertext before its release time. In some circumstances, the sender may wish to make the receiver decrypt a message different from that which was originally sent, by sending a false pre-open key to the receiver. An example attack is shown in Section 8.2.3.2. To make the TRE-PC scheme work properly, the possibility of malicious sender should also be considered in the security model. In the new security model (described in Section 8.3), we define a TRE-PC scheme to be binding if it is secure against such a malicious sender attack.

8.2.3 Analysis of the Hwang-Yum-Lee basic scheme

8.2.3.1 Description of the scheme

Hwang, Yum, and Lee [125] proposed the following TRE-PC scheme (the Hwang-Yum-Lee basic scheme) which is claimed to be IND-TR-CPA$_{\mathrm{OS}}$ and IND-TR-CCA$_{\mathrm{IS}}$ secure. The scheme works as follows.

- Setup: Given a security parameter ℓas input, the following parameters are generated:

- an additive group $\mathbb{G}_1$ of prime order q, a generator P of $\mathbb{G}_1$, and a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$,
- a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$,
- two cryptographic hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, and $\mathsf{H}_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n,
- a master key $s \in_R \mathbb{Z}_q$ for the time server, and the public key $S = sP$.

The message space and the ciphertext space are $\{0,1\}^n$ and $\mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^n$, respectively. The system parameters $param = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, S, n, \mathsf{H}_1, \mathsf{H}_2)$ are published.

- $\mathsf{Ext_{TS}}$: At time t, the time server computes $Q_t = \mathsf{H}_1(t)$ and publishes the timestamp $TS_t = sQ_t$.
- $\mathsf{Gen_{PK}}$: A user runs this algorithm to generate its public/private key pair (Y, x), where $x \in_R \mathbb{Z}_q$ and $Y = xP$.
- Enc: Suppose a sender wishes to send a message m; it first selects a release time t and a secret value $v \in_R \mathbb{Z}_q$, and outputs the ciphertext $C = (rP, vP, m \oplus \mathsf{H}_2(g_t))$ where $g_t = \hat{e}(rY + vS, Q_t)$ and $r \in_R \mathbb{Z}_q$.
- $\mathsf{Gen_{RK}}$: When the sender wants the ciphertext C to be decrypted before the release time t, it computes and publishes the pre-open key $V_t = vQ_t$.
- Dec: Before the release time, given the pre-open key V_t, the receiver decrypts $C = (R, V, W)$ by computing

$$m = W \oplus \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S))$$

Otherwise, given the timestamp TS_t, the receiver decrypts $C = (R, V, W)$ by computing

$$m = W \oplus \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V, TS_t))$$

8.2.3.2 Cryptanalysis

We show that a malicious sender can mount an attack to make the receiver decrypt a false message, which is different to the message the sender originally sent.

Suppose the sender sent the encrypted message $C = (rP, vP, m \oplus \mathsf{H}_2(g_t))$, and then before the release time t, it publishes a false pre-open key $V_t' \neq V_t$. With V_t', the receiver will compute the plaintext m' as:

$$\begin{aligned} m' &= m \oplus \mathsf{H}_2(g_t) \oplus \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \\ &= m \oplus \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S)) \oplus \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \end{aligned}$$

It is straightforward to verify that the probability that the following equation holds is negligible

$$\mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t, S)) = \mathsf{H}_2(\hat{e}(R, xQ_t)\hat{e}(V_t', S)) \,,$$

so that the probability that $m = m'$ is also negligible.

It should be noted that the other Hwang-Yum-Lee scheme (referred to as the full TRE-PC scheme), which is claimed in [125] to be IND-TR-CCA$_{\mathrm{OS/IS}}$ secure, appears to be resistant to a malicious sender attack, because the receiver checks the validity of the plaintext at the end of the decryption procedure.

Besides this potentially undesirable property, the decryption algorithm of this scheme does not satisfy the definition in the Hwang-Yum-Lee model. In the decryption algorithm, t should be included in the inputs to the decryption process, because $Q_t = \mathsf{H}_1(t)$ is used in the computation; however t is not required as an input for the decryption algorithm in the Hwang-Yum-Lee model. In fact, this inconsistency also applies to the full TRE-PC scheme.

8.3 New Security Model for TRE-PC Schemes

8.3.1 Description of the Model

As in the Hwang-Yum-Lee model, we assume that the following two types of entities are involved in a TRE-PC scheme:

- The users, each of which may act as both a sender and a receiver.
- A trusted time server which is required to publish timestamps periodically. We assume that the time server acts correctly in generating its parameters

and publishing the timestamps. However, when discussing semantic security, we take into account the fact that the time server may be curious, i.e. it may try to decrypt the ciphertext. Apart from this, the time server will do nothing else malicious.

In the new model, we consider the following four types of attackers:

- Outside attackers who do not know the master key of the time server. In the rest of this paper, the term *outside attacker* refers to this type of attacker, while the term *curious time server* (see below) refers to the special type of outside attacker which knows the master key of the time server.
- A curious time server which knows the master key of the time server.
- Authorised but curious receivers which try to decrypt the ciphertext before the release time without the pre-open key.
- Authorised but malicious senders who try to make the receiver recover a message different from that which was originally sent.

A TRE-PC scheme consists of the following six polynomial-time algorithms[1].

1. Setup: Run by the time server, the setup algorithm takes a security parameter ℓ as input, and generates a secret master-key mk and the global parameters $param$.
2. $\mathsf{Gen_U}$: Run by a user, the user key generation algorithm takes ℓ as input, and generates a public/private key pair (pk_r, sk_r).
3. $\mathsf{Ext_{TS}}$: Run by the time server, the timestamp extraction algorithm takes mk and a time t as input, and generates a timestamp TS_t for the time t.
4. Enc: Run by a sender, the encryption algorithm takes a message m, a release time t, and the receiver's public key pk_r as input, and returns a ciphertext C

[1]Note that we are giving here a definition for a TRE-PC scheme which is slightly different to the definition given previously; however, it should be clear from the context which definition applies.

and its pre-open key V_C. It should be noted that, initially, the sender should send the ciphertext C in conjunction with the release time t to the receiver, and therefore the receiver knows the release time of C. The sender stores the pre-open key V_C, and publishes it when pre-opening the ciphertext C.

5. $\mathsf{Dec_{RK}}$: Run by the receiver, the decryption algorithm takes a ciphertext C, the pre-open key V_C, and the receiver's private key sk_r as input, and returns either the plaintext or an error message $\perp$. In reality, the receiver can only run this algorithm after the sender releases the pre-open key V_C.

6. $\mathsf{Dec_{PK}}$: Run by the receiver, the decryption algorithm takes a ciphertext C, a timestamp TS_t which is determined by the release time accompanied with C, and the receiver's private key sk_r as input, and returns either the plaintext or an error message $\perp$.

It should be noted that *param* is an implicit input of all the algorithms except for Setup.

8.3.2 Security Notions for TRE-PC

The security definitions of IND-TR-CCA$_{\mathrm{OS}}$ and IND-TR-CPA$_{\mathrm{OS}}$ are refinements of those in the Hwang-Yum-Lee model, while IND-TR-CPA$_{\mathrm{IS}}$ is a refinement of the definition of IND-TR-CCA$_{\mathrm{IS}}$ in the Hwang-Yum-Lee model[2].

8.3.2.1 Soundness of a TRE-PC scheme

Informally, the decryption algorithms of a sound TRE-PC scheme should always "undo" the output of the encryption algorithm. Formally, soundness is defined as follows.

Definition 37 *A TRE-PC scheme is sound if, for any time t, message m, and*

[2]As for the definition of a TRE-PC scheme, we give here definitions for IND-TR-CCA$_{\mathrm{OS}}$ and IND-TR-CPA$_{\mathrm{OS}}$ which are slightly different from the definitions given previously; however, it should be clear from the context which definition applies.

(K, C) *where*

$$(C, V_C) = \mathsf{Enc}(m, t, pk_r),$$

the following two requirements are satisfied

$$m = \mathsf{Dec}_{\mathsf{RK}}(C, V_C, sk_r),$$

$$m = \mathsf{Dec}_{\mathsf{PK}}(C, TS_t, sk_r).$$

8.3.2.2 Binding of a TRE-PC scheme

As we have pointed out in the previous analysis, a sender may act maliciously when it tries to pre-open the ciphertext. In order to make a TRE-PC scheme work correctly, this type of malicious sender attack should be prevented. We give the following definition of *binding* to formalise this property.

Definition 38 *A TRE-PC scheme E is binding if any polynomial-time attacker $\mathcal{A}$ only has a negligible probability of winning the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters: $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).
2. Challenge: The attacker $\mathcal{A}$ executes with input $(pk_r, param)$. At some point, $\mathcal{A}$ generates a ciphertext C^* for release at time t^* and a pre-open key V_{C^*}, and then terminates by outputting (C^*, t^*, V_{C^*}). During its execution, $\mathcal{A}$ has access to the following oracles:
   - An oracle for $\mathsf{Ext_{TS}}$, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec_{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{Dec_{PK}}$, which, on receiving a query for (C, t'), where t' may not be the release time for C, returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$.

Suppose $O_1 = \mathsf{Dec}_{\mathsf{RK}}(C^*, V_{C^*}, sk_r)$ and $O_2 = \mathsf{Dec}_{\mathsf{PK}}(C^*, TS_{t^*}, sk_r)$. In this game $\mathcal{A}$ wins if $O_1 \neq \perp$, $O_2 \neq \perp$, and $O_1 \neq O_2$.

It is worth stressing that we have adopted the term "binding", which is also used to describe a property of commitment schemes, such as that descrribed in [50]. The binding property for a TRE-PC scheme guarantees that, if the attacker has encrypted some message, then it cannot release a pre-open key to force the receiver to decrypt a message which is different from that which was originally sent. This is clearly analogous to the binding property in commitment schemes. The difference is that explicit proofs are usually required in commitment schemes, while no such proofs are required here (as shown below). We further point out that, if the receiver obtains $\perp$ as a result of the decryption procedure, then it can confirm that the sender has malfunctioned. The formalisation of ciphertext validity, as that in [85], is outside the scope of this paper.

In fact, the binding property for a TRE-PC scheme also relates to the secure transportation of the pre-open key, when the sender decides to open the encrypted message before the pre-defined release time. If the TRE-PC scheme is binding, then the pre-open key does not need to be integrity protected; otherwise, the pre-open key should be integrity protected to guarantee that the receiver will obtain the message which the sender intended to send.

8.3.2.3 Security against malicious outsiders

In this subsection we define the notion of semantic security against outside attackers which do not know the time server's master key. Specifically, we define semantic security under an adaptive chosen ciphertext attack (IND-TR-CCA$_{\mathrm{OS}}$) and semantic security under an adaptive chosen plaintext attack (IND-TR-CPA$_{\mathrm{OS}}$).

Definition 39 *A TRE-PC scheme E is IND-TR-CCA$_{OS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ only has a negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master

key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - An oracle for $\mathsf{Ext_{TS}}$, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec_{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{Dec_{PK}}$, which, on receiving a query for (C, t'), returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$. Note that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by selecting two equal length messages m_0, m_1 and a release time t^*, and outputting (m_0, m_1, t^*). In addition, $\mathcal{A}_1$ also outputs some state information $state$.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, and returns $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$.

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracle as those available in Phase 1. However, $\mathcal{A}_2$ is not allowed to make the following two queries: a query to the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*), and a query to the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}).

   $\mathcal{A}_2$ terminates by outputting a guess bit $b' \in \{0, 1\}$.

$\mathcal{A}$ wins this game if $b = b'$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 40 *A TRE-PC scheme E is IND-TR-CPA$_{OS}$ secure, if it is IND-TR-CCA$_{OS}$ secure against attackers that make no decryption queries.*

8.3.2.4 Security against a curious time server

In this subsection we define the notion of semantic security against a curious time server. Specifically, we define semantic security under an adaptive chosen ciphertext attack (IND-TR-CCA$_{\mathrm{TS}}$) and semantic security under an adaptive chosen plaintext attack (IND-TR-CPA$_{\mathrm{TS}}$).

Definition 41 *A TRE-PC scheme E is IND-TR-CCA$_{TS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the following types of oracle.

   - An oracle for $\mathsf{Dec_{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{Dec_{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{Dec_{PK}}$, which, on receiving a query for (C, t'), returns $\mathsf{Dec_{PK}}(C, TS_{t'}, sk_r)$. Note that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by selecting two equal length messages m_0, m_1 and a release time t^*, and outputting (m_0, m_1, t^*). In addition, $\mathcal{A}_1$ also outputs some state information $state$.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, and returns $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$.

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same kinds of oracle as those available in Phase 1. However, $\mathcal{A}_2$ is not allowed to make the following two queries: a query to the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*), and a query to the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}).

$\mathcal{A}_2$ terminates by outputing a guess bit $b' \in \{0, 1\}$.

$\mathcal{A}$ wins this game if $b = b'$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 42 *A TRE-PC scheme E is IND-TR-CPA$_{TS}$ secure if it is IND-TR-CCA$_{TS}$ secure against attackers that make no decryption queries.*

From the above definitions, it follows immediately that IND-TR-CCA$_{\text{TS}}$/ IND-TR-CPA$_{\text{TS}}$ security correspondingly implies IND-TR-CCA$_{\text{OS}}$/ IND-TR-CPA$_{\text{OS}}$ security, because the attacker is granted more privileges in the former security definitions. However, observe that the notion of IND-TR-CCA$_{\text{TS}}$/ IND-TR-CPA$_{\text{TS}}$ security is irrelevant in an environment in which users can tolerate the time server being able to decrypt the ciphertext.

8.3.2.5 Security against a malicious receiver

If a TRE-PC scheme is to be deemed secure, then it should resist attacks in which the legitimate receiver attempts to decrypt the ciphertext before the release time and without the pre-open key. In this case, since the attacker will know the receiver's private key, it does not make sense to distinguish between the CCA and CPA notions of security. However, for the sake of consistency, we will use the term "IND-TR-CPA$_{\text{IS}}$" to describe semantic security of a scheme against a malicious receiver (an inside attacker).

Definition 43 *A TRE-PC scheme E is IND-TR-CPA$_{IS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs Setup to generate the time server's master key mk and the public system parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to an oracle for $\mathsf{Ext_{TS}}$, which, on receiving a query for time t, returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length messages m_0, m_1 and a release time t^* which is larger than all the inputs to the $\mathsf{Ext_{TS}}$ oracle. In addition, $\mathcal{A}_1$ also outputs some state information *state*.

3. Challenge: The challenger picks a random bit $b \in \{0, 1\}$, computes $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk_r)$, and returns C^*.

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(C^*, state)$. $\mathcal{A}_2$ has access to the $\mathsf{Ext_{TS}}$ oracle, which, on receiving a query for time $t < t^*$, returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_2$ terminates by outputting a guess bit $b' \in \{0, 1\}$.

$\mathcal{A}$ wins this game if $b = b'$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

8.4 Relationships between the Security Notions

We first give two definitions: "A $\longrightarrow$ B" means that if a scheme is secure in the sense of A then it is secure in the sense of B; "A $\not\longrightarrow$B" means that, even if a scheme is secure in the sense of A, it may not be secure in the sense of B. In other words, "A $\not\longrightarrow$B" means that we can construct a scheme which is secure in the sense of A, but not secure in the sense of B. It is clear to see that the "$\longrightarrow$" relation is transitive, which means that if A $\longrightarrow$ B and B $\longrightarrow$ C then A $\longrightarrow$ C.

We prove that the relationships indicated in Figure 8.1 hold for the security notions, where the arrow represents the relation "$\longrightarrow$" while the hatched arrow represents the relationship "$\not\longrightarrow$ ". Given the relationships shown, it is straightforward to deduce the relationship between any two security notions.

From the definitions in Section 8.3.2, it follows immediately that the following relationships hold.

1. IND-TR-CCA$_{\mathrm{TS}}$ $\longrightarrow$ IND-TR-CCA$_{\mathrm{OS}}$ $\longrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$
2. IND-TR-CCA$_{\mathrm{TS}}$ $\longrightarrow$ IND-TR-CPA$_{\mathrm{TS}}$ $\longrightarrow$ IND-TR-CPA$_{\mathrm{OS}}$

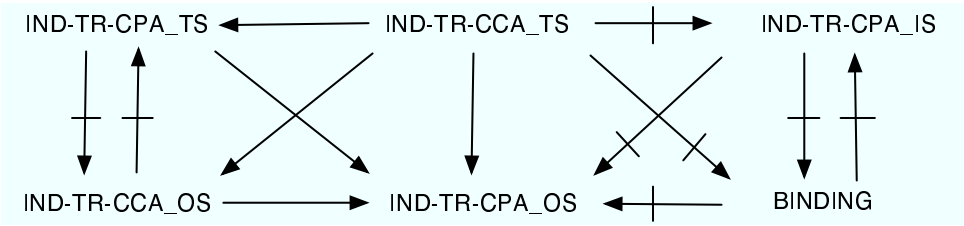

Figure 8.1: Relationships among the security notions

In the remainder of this chapter we establish the other relationships indicated in Figure 8.1. We implicitly assume throughout that a TRE-PC scheme is sound.

8.4.1 Relationship between IND-TR-CPA$_{\mathsf{TS}}$ and IND-TR-CCA$_{\mathsf{OS}}$

In this section we consider the relationships between IND-TR-CPA$_{\mathrm{TS}}$ and IND-TR-CCA$_{\mathrm{OS}}$ for TRE-PC schemes.

Construction 1 *Suppose* $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ *is a TRE-PC scheme, then we can construct a new TRE-PC scheme* $E' = (\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}', \mathsf{Dec}'_{\mathsf{RK}}, \mathsf{Dec}'_{\mathsf{PK}})$*, where the algorithms are defined as follows:*

- *The algorithms* Setup',$\mathsf{Gen}'_{\mathsf{U}}$,$\mathsf{Ext}'_{\mathsf{TS}}$ *are defined in the same way as in* E*.*
- $\mathsf{Enc}'(m, t, pk'_r) = (C||0, V_C)$*, where* $(C, V_C) = \mathsf{Enc}(m, t, pk'_r)$*.*
- $\mathsf{Dec}'_{\mathsf{RK}}(C||b, V_C, sk'_r) = \mathsf{Dec_{RK}}(C, V_C, sk'_r)$*, where* $b \in \{0, 1\}$*.*
- $\mathsf{Dec}'_{\mathsf{PK}}(C||b, TS_t, sk'_r) = \mathsf{Dec_{PK}}(C, TS_t, sk'_r)$*, where* $b \in \{0, 1\}$*.*

Lemma 13 *If* E *is IND-TR-CPA*$_{TS}$ *secure and* E' *is obtained from* E *using Construction 1, then* E' *is IND-TR-CPA*$_{TS}$ *secure.*

Proof. Suppose an IND-TR-CPA$_{\mathrm{TS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{TS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. We thus

can conclude that δ is negligible since E is IND-TR-CPA$_{\mathrm{TS}}$ secure. The attacker $\mathcal{A}_1$ is defined as follows.

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the master key mk.
2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $mk' = mk$.
3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ with input mk', pk'_r, and $param'$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.
4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger chooses $b \in_R \{0,1\}$ and computes the challenge $(C^*, V_C^*) = \mathsf{Enc}(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^*, the pre-open key V_{C^*}, and the state information $state$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*||0, V_{C^*}, state)$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.
3. $\mathcal{A}_2$ terminates by outputting the bit b'.

$\mathcal{A}$ is a legitimate IND-TR-CPA$_{\mathrm{TS}}$ attacker, and $\mathcal{A}$'s advantage is equal to δ. Since E is IND-TR-CPA$_{\mathrm{TS}}$ secure, then δ is negligible, and the lemma follows. □

Lemma 14 *If E' is obtained from a TRE-PC scheme E using Construction 1, then E' is not IND-TR-CCA$_{OS}$ secure.*

Proof. To prove the claim, we only need to show that an IND-TR-CCA$_{\mathrm{OS}}$ attacker has a non-negligible advantage in attacking E'. The attack is elementary. On receiving the challenge $(C^*||0, V_{C^*})$ from the challenger, $\mathcal{A}_2$ makes a query to the oracle

$\mathsf{Dec}'_{\mathsf{PK}}$ on the input $(C^*||1, t^*)$. It is easy to see that the oracle $\mathsf{Dec}'_{\mathsf{PK}}$ returns m_b, which allows the attacker to recover the bit b with probability 1. The lemma now follows □

Theorem 15 *If the set of IND-TR-CPA$_{TS}$ secure TRE-PC schemes is non-empty, then IND-TR-CPA$_{TS}$ $\not\rightarrow$ IND-TR-CCA$_{OS}$.*

Proof. Suppose E is an IND-TR-CPA$_{\mathrm{TS}}$ secure TRE-PC scheme, and generate E' from E using Construction 1. Then E' is IND-TR-CPA$_{\mathrm{TS}}$ secure (by Lemma 13) but is not IND-TR-CCA$_{\mathrm{OS}}$ secure (by Lemma 14). The result follows. □

Construction 2 *Suppose that $E = (\mathsf{Setup}, \mathsf{Gen}_{\mathsf{U}}, \mathsf{Ext}_{\mathsf{TS}}, \mathsf{Enc}, \mathsf{Dec}_{\mathsf{RK}}, \mathsf{Dec}_{\mathsf{PK}})$ is a TRE-PC scheme and $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a public key encryption scheme; we can construct a TRE-PC scheme E', where the algorithms are defined as follows.*

1. *The Setup' algorithm takes a security parameter ℓ as input, and computes $(param, mk) = \mathsf{Setup}(\ell)$ and $(pk, sk) = \mathcal{K}(\ell)$. The public parameters are defined to be $param' = (param, pk)$. The master key is defined to be $mk' = (mk, sk)$.*
2. *The $\mathsf{Gen}'_{\mathsf{U}}$ algorithm takes ℓ as input, and computes $(pk'_r, sk'_r) = \mathsf{Gen}_{\mathsf{U}}(\ell)$.*
3. *The $\mathsf{Ext}'_{\mathsf{TS}}$ algorithm take as input mk' and a release time t, and returns $\mathsf{Ext}_{\mathsf{TS}}(mk, t)$.*
4. *The Enc' algorithm takes as input a message m, a release time t and the receiver's public key pk'_r, and returns a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, where:*

$$\begin{aligned} C_1 &= \mathcal{E}(m, pk) \\ (C_2, V_C) &= \mathsf{Enc}(C_1||m, t, pk'_r) \end{aligned}$$

We assume that C_1 is drawn from a prefix-free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bit string $C_1||m$.

5. *The $\mathsf{Dec}'_{\mathsf{RK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a pre-open key V_C, and a private key sk'_r, and computes*

$$C'_1||m = \mathsf{Dec}_{\mathsf{RK}}(C_2, V_C, sk'_r)\,.$$

If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

6. *The $\mathsf{Dec}'_{\mathsf{PK}}$ algorithm takes as input a ciphertext $C = (C_1, C_2)$, a timestamp TS_t, and a private key sk'_r, and computes*

$$C'_1||m = \mathsf{Dec}_{\mathsf{PK}}(C_2, TS_t, sk'_r)\,.$$

If $C'_1 \neq C_1$, then the algorithm outputs $\perp$. Otherwise it returns m.

Lemma 16 *If E' is obtained from IND-TR-CCA$_{OS}$ secure TRE-PC scheme E using Construction 2, then E' is IND-TR-CCA$_{OS}$ secure.*

Proof. Consider the following game played between an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ and a hypothetical challenger. The game is parameterised by two bits $(b_1, b_2) \in_R \{0,1\}^2$ and ℓ, and runs as follows:

1. Game Setup: The challenger runs Setup' to generate public parameters $param' = (param, pk)$ and a private key $mk' = (mk, sk)$. The challenger also runs $\mathsf{Gen}'_{\mathsf{U}}$ to generate a public/private key pair (pk'_r, sk'_r).

2. Phase 1: The attacker $\mathcal{B}_1$ executes with input $(pk'_r, param')$. $\mathcal{B}_1$ has access to the following oracles:

   - An oracle for $\mathsf{Ext}'_{\mathsf{TS}}$, which takes as input a release time t and returns $\mathsf{Ext}'_{\mathsf{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec}'_{\mathsf{RK}}$, which takes as input a ciphertext C and a pre-open key V_C, and returns $\mathsf{Dec}'_{\mathsf{RK}}(C, V_C, sk'_r)$.
   - An oracle for $\mathsf{Dec}'_{\mathsf{PK}}$, which takes as input a ciphertext C and a release time t, and returns $\mathsf{Dec}'_{\mathsf{PK}}(C, TS_t, sk'_r)$.

$\mathcal{B}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^* and some state information *state*.

3. Challenge: The challenger computes $C_1^* = \mathcal{E}(m_{b_1}, pk)$ and (C_2^*, V_{C^*}), where

$$(C_2^*, V_{C^*}) = \mathsf{Enc}(C_1^* || m_{b_2}, t^*, pk_r').$$

The challenge ciphertext is $C^* = (C_1^*, C_2^*)$. The challenge pre-open key is V_{C^*}.

4. Phase 2: The attacker $\mathcal{B}_2$ executes with input $(C^*, V_{C^*}, state)$. $\mathcal{B}_2$ has access to the same types of oracle as in Phase 1; however, $\mathcal{B}_2$ is not permitted to query the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle on the input (C^*, V_{C^*}) or the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle on the input (C^*, t^*). $\mathcal{B}_2$ terminates by outputting a guessing bit b' for b_2.

When $b_1 = b_2$, this is a legitimate IND-TR-CCA$_{\mathrm{OS}}$ game for E'. Let $Exp(b_1, b_2)$ be the event that $\mathcal{B}$ outputs 1 in the above game. We next prove the following three claims.

Claim 4 *$\mathcal{B}$'s advantage in winning the IND-TR-CCA$_{OS}$ game is equal to*

$$\frac{1}{2} \left| \Pr[Exp(0,0)] - \Pr[Exp(1,1)] \right|$$

Proof. In an IND-TR-CCA$_{\mathrm{OS}}$ attack game, the attacker's advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$, where b is the bit randomly selected by the challenger in constructing the challenge ciphertext, and b' is the bit output by the attacker at the end of the game. The probability $Pr[b = b']$ can be computed as:

$$\begin{aligned} Pr[b = b'] &= Pr[b = b' = 0] + Pr[b = b' = 1] \\ &= \frac{1}{2} Pr[b' = 0 | b = 0] + \frac{1}{2} Pr[b' = 1 | b = 1] \\ &= \frac{1}{2}(1 - Pr[b' = 1 | b = 0] + Pr[b' = 1 | b = 1]) \\ &= \frac{1}{2} + \frac{1}{2}(Pr[b' = 1 | b = 1] - Pr[b' = 1 | b = 0]) \end{aligned}$$

However, by inspection of the game which defines the event $Exp(b_1, b_2)$, it is clear that

$$Pr[b' = 1 | b = 0] = Pr[Exp(0,0)] \text{ and } Pr[b' = 1 | b = 1] = Pr[Exp(1,1)].$$

Therefore, $\mathcal{B}$'s advantage is equal to $\frac{1}{2}\,|\Pr[Exp(0,0)] - \Pr[Exp(1,1)]|$. □

Note that, by the triangle inequality, $\mathcal{B}$'s advantage is bounded by

$$\frac{1}{2}\,|\Pr[Exp(0,0)] - \Pr[Exp(0,1)]| + \frac{1}{2}\,|\Pr[Exp(0,1)] - \Pr[Exp(1,1)]|$$

Claim 5 *If E is IND-TR-CCA$_{OS}$ secure, then $|\Pr[Exp(0,0)] - \Pr[Exp(0,1)]|$ is negligible as a function of ℓ.*

Proof. We show that there exists an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that has advantage

$$\frac{1}{2}\,|\Pr[Exp(0,0)] - \Pr[Exp(0,1)]|\ .$$

Hence, we can conclude that $|\Pr[Exp(0,0)] - \Pr[Exp(0,1)]|$ is negligible, since E is IND-TR-CCA$_{\mathrm{OS}}$ secure. The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$ and the public key pk'_r.
2. $\mathcal{A}_1$ computes an encryption key pair $(pk, sk) = \mathcal{K}(\ell)$.
3. $\mathcal{A}_1$ sets $param' = (param, pk)$.
4. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input pk'_r and $param'$.
   - If $\mathcal{B}_1$ makes a extraction oracle query for a time t, then $\mathcal{A}_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C_2, V_C). Suppose it receives $C'_1||m$ from the oracle. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$ to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C_2, t). Suppose it receives $C'_1||m$ from the oracle. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$ to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m to $\mathcal{B}_1$.

$\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, and some state information $state'$.

5. $\mathcal{A}_1$ computes $C_1^* = \mathcal{E}(m_0, pk)$.
6. $\mathcal{A}_1$ terminates by outputting the messages $C_1^*||m_0$ and $C_1^*||m_1$, a release time t^* and the state information $state = (state', C_1^*, pk, sk)$.

The challenger chooses $b_2 \in_R \{0,1\}$ and computes the challenge $(C_2^*, V_C^*) = \mathsf{Enc}(m_{b_2}, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C_2^*, the challenge pre-open key V_{C^*} and the state information $state = (state', C_1^*, pk, sk)$. It sets $C^* = (C_1^*, C_2^*)$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state')$.
   - If $\mathcal{B}_2$ makes a extraction oracle query for a time t, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then
     - If $C_2 = C_2^*$ and $V_C = V_{C^*}$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$.
     - Otherwise $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C_2, V_C). Suppose it receives $C_1'||m$ from the oracle. If $C_1' \neq C_1$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then
     - If $C_2 = C_2^*$ and $t = t^*$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$.
     - Otherwise $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C_2, t). Suppose it receives $C_1'||m$ from the oracle. If $C_1' \neq C_1$ then $\mathcal{A}_2$ returns $\perp$ to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m to $\mathcal{B}_2$.

   $\mathcal{B}_2$ eventually terminates by outputting a bit b'.
3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to verify that the decryption oracles that $\mathcal{A}$ perfectly simulates the decryption oracles for $\mathcal{B}$, providing that $\mathcal{A}_2$ does not incorrectly respond $\perp$ for a ciphertext (C_1, C_2). There are two case in which this might occur:

- If $\mathcal{B}_2$ makes a $\mathsf{Dec}_{\mathsf{RK}}$ query on a ciphertext (C_1, C_2) with $C_1 \neq C_1^*$, $C_2 = C_2^*$ and $V_C = V_{C^*}$.
- If $\mathcal{B}_2$ makes a $\mathsf{Dec}_{\mathsf{PK}}$ query on a ciphertext (C_1, C_2) with $C_1 \neq C_1^*$, $C_2 = C_2^*$ and $t = t^*$.

In either case, we would recover $C_1^*||m_{b_2}$ after we apply the TRE-PC decryption algorithm. Hence, the ciphertext is invalid because $C_1 \neq C_1^*$. Hence, the response of $\perp$ is correct.

Therefore, $\mathcal{A}$ is a legitimate IND-TR-CCA$_{\mathrm{OS}}$ attacker, and its advantage is

$$\frac{1}{2} \left| \Pr[\mathcal{A} \text{ outputs } 1 | b_2 = 0] - Pr[\mathcal{A} \text{ outputs } 1 | b_2 = 1] \right|,$$

and so this value is negligible, since E is an IND-TR-CCA$_{\mathrm{OS}}$ TRE-PC scheme. However, this value is equal to

$$\frac{1}{2} \left| \Pr[Exp(0,0)] - \Pr[Exp(0,1)] \right|$$

The claim now follows. □

Claim 6 *If $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure, then $|\Pr[Exp(0,1)] - \Pr[Exp(1,1)]|$ is negligible as a function of ℓ.*

Proof. We show that there exists an IND-CPA attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, which makes use of $\mathcal{B}$ as a subroutine, that has advantage

$$\frac{1}{2} \left| \Pr[Exp(0,1)] - \Pr[Exp(1,1)] \right|.$$

Hence, we conclude that $|\Pr[Exp(0,1)] - \Pr[Exp(1,1)]|$ is negligible since $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ is IND-CPA secure. The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public key pk from the challenger.
2. $\mathcal{A}_1$ computes $(param, mk) = \mathsf{Setup}(\ell)$ and $(pk'_r, sk'_r) = \mathsf{Gen}'_{\mathsf{U}}(\ell)$.
3. $\mathcal{A}_1$ sets $param' = (param, pk)$.
4. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input of pk'_r and $param'$.
   - If $\mathcal{B}_1$ makes an $\mathsf{Ext}'_{\mathsf{TS}}$ query for a time t, then $\mathcal{A}_1$ computes $TS_t = \mathsf{Ext}_{\mathsf{TS}}(mk, t)$ and returns TS_t to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ makes a $\mathsf{Dec}_{\mathsf{RK}}$ query for a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, then $\mathcal{A}_1$ computes $C'_1||m = \mathsf{Dec}_{\mathsf{RK}}(C_2, V_C, sk_r)$. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$; otherwise $\mathcal{A}_1$ returns m to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ makes a $\mathsf{Dec}_{\mathsf{PK}}$ query for a ciphertext $C = (C_1, C_2)$ and a release time t, then $\mathcal{A}_1$ computes $C'_1||m = \mathsf{Dec}_{\mathsf{PK}}(C_2, TS_t, sk_r)$. If $C'_1 \neq C_1$ then $\mathcal{A}_1$ returns $\perp$; otherwise $\mathcal{A}_1$ returns m to $\mathcal{B}_1$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^* and some state information $state'$.
5. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, and some state information $state = (state', pk, param, mk, pk_r, sk_r, m_0, m_1, t^*)$.

The challenger chooses $b_1 \in_R \{0, 1\}$ and computes the challenge $C_1^* = \mathcal{E}(m_{b_1}, pk)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C_1^* and the state information $state = (state', pk, param, mk, pk_r, sk_r, m_0, m_1, t^*)$.
2. $\mathcal{A}_2$ computes $(C_2^*, V_{C^*}) = \mathsf{Enc}(C_1^*||m_1, t^*, pk_r)$ and sets $C^* = (C_1^*, C_2^*)$.
3. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state')$. If $\mathcal{B}_2$ makes any oracle queries, then $\mathcal{A}_2$ answers them in exactly the same way as $\mathcal{A}_1$ would have done. $\mathcal{B}_2$ terminates by outputting a bit b'.
4. $\mathcal{A}_2$ terminates by outputting the bit b'.

Clearly, for the same reasons as in the previous claim, $\mathcal{A}$'s advantage is equal to $\frac{1}{2}|\Pr[Exp(0,1)] - Exp(1,1)]|$ and hence this value is negligible, and the claim follows. □

From the above three claims, we have shown that any legitimate attacker, namely $\mathcal{B}$ when $b_1 = b_2$, only has a negligilble advantage in an IND-TR-CCA$_{\mathrm{OS}}$ attack game for E', and the lemma now follows. □

We also have the following.

Lemma 17 *If E' is obtained from a TRE-PC scheme using Construction 2, then E' is not IND-TR-CPA$_{TS}$ secure.*

Proof. To prove the claim, we only need to show that the an IND-TR-CPA$_{\mathrm{TS}}$ attacker has a non-negligible advantage in attacking E'. The attack is elementary. Suppose that the challenge is (C^*, V_{C^*}), where $C^* = (C_1^*, C_2^*)$. Since the attacker has access to sk, then it can immediately compute $m_b = \mathcal{D}(C_1^*, sk)$, which means it can succeed in guessing b with probability 1. The lemma now follows. □

Theorem 18 *If the set of IND-TR-CCA$_{OS}$ secure TRE-PC schemes is non-empty, then IND-TR-CCA$_{OS}$ $\not\rightarrow$ IND-TR-CPA$_{TS}$.*

Proof. Suppose E is an IND-TR-CCA$_{\mathrm{OS}}$ secure TRE-PC scheme, and generate E' from E using Construction 2. Then E' is IND-TR-CCA$_{\mathrm{OS}}$ secure (by Lemma 16) but is not IND-TR-CPA$_{\mathrm{TS}}$ secure (by Lemma 17). The result now follows. □

8.4.2 Relationship between IND-TR-CCA$_{\mathrm{TS}}$ and IND-TR-CPA$_{\mathrm{IS}}$

We prove that IND-TR-CCA$_{\mathrm{TS}}$ does not imply IND-TR-CPA$_{\mathrm{IS}}$ for TRE-PC schemes.

Construction 3 *Suppose that* $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ *is a TRE-PC scheme and* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ *is a public key encryption scheme. We construct a TRE-PC scheme* E'*, where the algorithms are defined as follows.*

1. *The* Setup' *algorithm takes a security parameter* ℓ *as input, and computes the public/private parameters* $(param', mk') = \mathsf{Setup}(\ell)$.

2. *The* $\mathsf{Gen}'_{\mathsf{U}}$ *algorithm takes as input the public parameters* $param'$*, and computes* $(pk_r, sk_r) = \mathsf{Gen_U}(\ell)$ *and* $(pk, sk) = \mathcal{K}(\ell)$*. The user's public key is defined to be* $pk'_r = (pk_r, pk)$*. The user's private key is defined to be* $sk'_r = (sk_r, sk)$.

3. *The* $\mathsf{Ext}'_{\mathsf{TS}}$ *algorithm take* mk' *and a release time* t *as input, and returns* $\mathsf{Ext_{TS}}(mk', t)$.

4. *The* Enc' *algorithm takes as input a message* m*, a release time* t *and the receiver's public key* pk'_r*, and returns a ciphertext* $C = (C_1, C_2)$ *and a pre-open key* V_C*, where:*

$$\begin{aligned} C_1 &= \mathcal{E}(m, pk) \\ (C_2, V_C) &= \mathsf{Enc}(C_1||m, t, pk_r) \end{aligned}$$

   We assume that C_1 *is drawn from a prefix-free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bitstring* $C_1||m$.

5. *The* $\mathsf{Dec}'_{\mathsf{RK}}$ *algorithm takes as input a ciphertext* $C = (C_1, C_2)$*, a pre-open key* V_C*, and a private key* sk'_r*, and computes*

$$C'_1||m = \mathsf{Dec_{RK}}(C_2, V_C, sk_r)\,.$$

   If $C'_1 \neq C_1$*, then the algorithm outputs* $\perp$*. Otherwise it returns* m.

6. *The* $\mathsf{Dec}'_{\mathsf{PK}}$ *algorithm takes as input a ciphertext* $C = (C_1, C_2)$*, a timestamp* TS_t*, and a private key* sk'_r*, and computes*

$$C'_1||m = \mathsf{Dec_{PK}}(C_2, TS_t, sk_r)\,.$$

   If $C'_1 \neq C_1$*, then the algorithm outputs* $\perp$*. Otherwise it returns* m.

E' is constructed in the same way as in Construction 2, except that (pk, sk) is possessed by the receiver instead of by the trusted time server. Therefore, following exactly the same procedure, we can establish the following lemma (the proof is omitted because of its similarity to Lemma 16).

Lemma 19 *If E' is obtained from an IND-TR-CCA$_{TS}$ secure TRE-PC scheme E using Construction 3, then E' is IND-TR-CCA$_{TS}$ secure.*

We also have the following.

Lemma 20 *If E' is obtained from a TRE-PC scheme using Construction 3, then E' is not IND-TR-CPA$_{IS}$ secure.*

Proof. To prove the lemma, we only need to show that the an IND-TR-CPA$_{\mathrm{IS}}$ attacker has non-negligible advantage in attacking E'. The attack is elementary. Suppose that the challenge is $C^* = (C_1^*, C_2^*)$. Since the attacker has access to sk, then it can immediately compute $m_b = \mathcal{D}(C_1^*, sk)$, which means it can succeed in guessing b with probability 1. The lemma now follows. □

Theorem 21 *If the set of IND-TR-CCA$_{TS}$ secure TRE-PC schemes is non-empty, then IND-TR-CCA$_{TS}$ $\not\rightarrow$ IND-TR-CPA$_{IS}$.*

Proof. Suppose E is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme, and generate E' from E using Construction 3. Then E' is IND-TR-CCA$_{\mathrm{TS}}$ secure (by Lemma 19) but is not IND-TR-CPA$_{\mathrm{IS}}$ secure (by Lemma 20). The result follows. □

8.4.3 Relationship between IND-TR-CPA$_{\mathrm{IS}}$ and IND-TR-CPA$_{\mathrm{OS}}$

We prove that IND-TR-CPA$_{\mathrm{IS}}$ does not imply IND-TR-CPA$_{\mathrm{OS}}$ for TRE-PC schemes.

Construction 4 *Suppose that* $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ *is a TRE-PC scheme. We can construct a TRE-PC scheme* E'*, where the algorithms are defined as follows.*

1. *The algorithms* $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}$, $\mathsf{Dec}'_{\mathsf{PK}}$ *are defined in the same way as in* E*.*

2. *The* Enc' *algorithm takes a message* m*, a release time* t *and the receiver's public key* pk'_r *as input, and returns a ciphertext* C *and a pre-open key* V_C*, where:*

$$\begin{aligned}(C, V'_C) &= \mathsf{Enc}(m, t, pk'_r)\\ V_C &= V'_C||m\end{aligned}$$

   We assume that V'_C *is drawn from a prefix-free set (such as a set of strings of a fixed length) so that it may be recovered from the arbitrary bitstring* $V'_C||m$*.*

3. *The* $\mathsf{Dec}'_{\mathsf{RK}}$ *algorithm takes a ciphertext* C*, a pre-open key* $V'_C||m$*, and a private key* sk'_r *as input, and returns* $\mathsf{Dec_{RK}}(C, V'_C, sk'_r)$*.*

Lemma 22 *If* E' *is obtained from an IND-TR-CPA*$_{IS}$ *secure TRE-PC scheme* E *using Construction 4, then* E' *is IND-TR-CPA*$_{IS}$ *secure.*

Proof. Suppose an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we can conclude that δ is negligible, since E is IND-TR-CPA$_{\mathrm{IS}}$ secure. The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$ and the public/private key pair (pk_r, sk_r).

2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $sk'_r = sk$.

3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input pk'_r, sk'_r, and $param'$. If $\mathcal{B}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}_1$ makes a similar query to its own extraction

oracle and returns the timestamp TS_t to $\mathcal{B}_1$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.

4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger chooses $b \in_R \{0,1\}$ and computes the challenge TRE-PC encryption C^*, where $(C^*, V_C^*) = \mathsf{Enc}(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^* and the state information $state$.
2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, state)$. If $\mathcal{B}_2$ makes an extraction oracle query for a time $t < t^*$, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.
3. $\mathcal{A}_2$ terminates by outputting the bit b'.

It is clear to see that $\mathcal{A}$ provides perfect simulation for the oracles that $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attacker, and $\mathcal{A}$'s advantage equals δ. Since E is an IND-TR-CPA$_{\mathrm{IS}}$ TRE-PC scheme, then δ is negligible, and the lemma now follows. □

Lemma 23 *If E' is obtained from a TRE-PC scheme E using Construction 4, then E' is not IND-TR-CPA$_{OS}$ secure.*

Proof. It is straightforward to verify that an IND-TR-CPA$_{\mathrm{OS}}$ attacker can identify the random bit chosen by the challenger with probability 1, because the pre-open key always contains the plaintext. As a result, the lemma follows. □

Theorem 24 *If the set of IND-TR-CPA$_{TS}$ secure TRE-PC schemes is non-empty, then IND-TR-CPA$_{IS}$ $\not\rightarrow$ IND-TR-CPA$_{OS}$.*

Proof. Suppose E is an IND-TR-CPA$_{\mathrm{IS}}$ secure TRE-PC scheme, and generate E' from E using Construction 3. Then E' is IND-TR-CPA$_{\mathrm{IS}}$ secure (by Lemma 22) but is not IND-TR-CPA$_{\mathrm{OS}}$ secure (by Lemma 23). The result follows. □

8.4.4 Relationship between IND-TR-CCA$_{\mathsf{TS}}$ and Binding

We prove that IND-TR-CCA$_{\mathrm{TS}}$ does not imply binding for TRE-PC schemes.

Construction 5 *Suppose that* $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ *is a TRE-PC scheme. We can construct a TRE-PC scheme* E'*, where the algorithms are defined as follows.*

1. *The algorithms* $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}'$ *are defined in the same way as in* E*.*
2. *The algorithm* $\mathsf{Dec}'_{\mathsf{RK}}$ *is defined in the same way as* $\mathsf{Dec_{RK}}$*, except that it returns a random message from the plaintext space when* $\mathsf{Dec_{RK}}$ *returns* $\perp$*.*
3. *The algorithm* $\mathsf{Dec}'_{\mathsf{PK}}$ *is defined in the same way as* $\mathsf{Dec_{PK}}$*, except that it returns a random message from the plaintext space when* $\mathsf{Dec_{PK}}$ *returns* $\perp$*.*

Lemma 25 *If* E' *is obtained from an IND-TR-CCA*$_{TS}$ *secure TRE-PC scheme* E *using Construction 5, then* E' *is IND-TR-CCA*$_{TS}$ *secure.*

Proof. Suppose an IND-TR-CCA$_{\mathrm{TS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has advantage δ in attacking E'. We show that there exists an IND-TR-CCA$_{\mathrm{TS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we will be able to conclude that δ is negligible since E is IND-TR-CCA$_{\mathrm{TS}}$ secure. The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the master key mk.

2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $mk' = mk$.

3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ on the input mk', pk'_r, and $param'$.

   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext C and the pre-open key V_C, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C, V_C). Suppose it receives m' from the oracle. If $m' = \perp$ then $\mathcal{A}_1$ returns a random message from the plaintext space to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m' to $\mathcal{B}_1$.
   - If $\mathcal{B}_1$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with ciphertext C and the time t, then $\mathcal{A}_1$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C, t). Suppose it receives m' from the oracle. If $m' = \perp$ then $\mathcal{A}_1$ returns a random message from the plaintext space to $\mathcal{B}_1$. Otherwise $\mathcal{A}_1$ returns the message m' to $\mathcal{B}_1$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.

4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger chooses $b \in_R \{0, 1\}$ and computes the challenge $(C^*, V_C^*) = \mathsf{Enc}'(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^*, the challenge pre-open key V_{C^*} and the state information $state$.

2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, V_{C^*}, state)$.

   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{RK}}$ oracle on (C, V_C). Suppose $\mathcal{A}_2$ receives m' as the response. If $m' = \perp$ then $\mathcal{A}_2$ returns a random message from the plaintext space to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m' to $\mathcal{B}_2$.
   - If $\mathcal{B}_2$ queries the $\mathsf{Dec}'_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}_2$ queries its $\mathsf{Dec}_{\mathsf{PK}}$ oracle on (C, t). Suppose $\mathcal{A}_2$ receives m' as the response. If $m' = \perp$ then $\mathcal{A}_2$ returns a random message from the plaintext space to $\mathcal{B}_2$. Otherwise $\mathcal{A}_2$ returns the message m' to $\mathcal{B}_2$.

   $\mathcal{B}_2$ eventually terminates by outputting a bit b'.

3. $\mathcal{A}_2$ terminates by outputting the bit b'.

$\mathcal{A}$ provides perfect simulation for the oracles that $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CCA$_{\mathrm{TS}}$ attacker, and $\mathcal{A}$'s advantage equals δ. Since E is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme, δ is negligible; the lemma now follows. □

Lemma 26 *If E' is obtained from a TRE-PC scheme E using Construction 5, then E' is not binding.*

Proof. We first construct a two-stage algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, and then construct a binding attacker $\mathcal{A}'$ for E'. The sub-algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ are defined as follows:

1. $\mathcal{A}_1$ takes $(pk_r, param)$ as input, where pk_r is a string from the same space as pk'_r and $param$ is a string from the same space as $param'$, and returns (m_0, m_1, t^*), where m_0 and m_1 are two equal length messages randomly chosen from the plaintext space of E' and $t^* = 2$.
2. $\mathcal{A}_2$ takes (m_0, m_1, C, V_C) as input, where C is from the ciphertext space of E', V_C is from the pre-open key space of E'. $\mathcal{A}_2$ sets $t^\dagger = 1$ and makes a $\mathsf{Dec}'_{\mathsf{PK}}$ query with input $(C, t^\dagger)$. If $\mathcal{A}_2$ receives $m^\dagger$ as the response, then it terminates by outputting a bit b' which is defined as follows: $b' = i$ if $m^\dagger = m_i$ $(0 \leq i \leq 1)$; otherwise, b' is set randomly.

The attacker $\mathcal{A}'$ is defined as follows. $\mathcal{A}'$ receives $(pk'_r, param')$ from the challenger, and then

1. runs $\mathcal{A}_1$ on the input of $(pk'_r, param')$ and get the output (m_0, m_1, t^*),
2. selects $b \in_R \{0, 1\}$ and computes $(C^*, V_{C^*}) = \mathsf{Enc}(m_b, t^*, pk'_r)$,
3. runs $\mathcal{A}_2$ with input (m_0, m_1, C^*, V_{C^*}),
4. on receiving $\mathcal{A}_2$'s $\mathsf{Dec}'_{\mathsf{PK}}$ query, makes a $\mathsf{Dec}'_{\mathsf{PK}}$ query to its own oracle with input $(C^*, t^\dagger)$, returns the output to $\mathcal{A}_2$,

5. gets b' as output from $\mathcal{A}_2$ and terminates by outputting $(C^*, V_{C^*}, t^\dagger)$.

From the above description, it follows that $\mathcal{A}'$ is a legitimate binding attacker, $\mathcal{A}'$'s advantage is $\delta = Pr[m^\dagger \neq m_b]$, and $|\Pr[b = b'] - \frac{1}{2}|$ is negligible because E' is also IND-TR-CCA$_{\mathrm{TS}}$ secure.

Let E_1 and E_2 be the events that $m^\dagger = m_b$ and $m^\dagger = m_{\bar{b}}$ respectively, where $\bar{b} = |b - 1|$. Let E_3 be the event that neither E_1 nor E_2 occurs. We then have

$$\begin{aligned} \Pr[b = b'] &= \sum_{i=1}^{3} \Pr[E_i] \Pr[b = b' | E_i] && (8.1) \\ &= \Pr[E_1] + \frac{1}{2} \Pr[E_3] && (8.2) \\ &= 1 - \Pr[E_2] - \frac{1}{2} \Pr[E_3] && (8.3) \end{aligned}$$

Note that the reduction from (8.1) to (8.2) is based on the fact that $\Pr[b = b'|E_1] = 1$, $\Pr[b = b'|E_2] = 0$, and $\Pr[b = b'|E_3] = \frac{1}{2}$, while the reduction from (8.2) to (8.3) is based on the fact that $\sum_{i=1}^{3} \Pr[E_i] = 1$.

As a result, $|\Pr[b = b'] - \frac{1}{2}|$ is negligible implies that $|\frac{1}{2} - \delta'|$, where $\delta' = \Pr[E_2] + \frac{1}{2}\Pr[E_3]$, is also negligible, so that δ' is non-negligible. Since $\delta \geq \delta'$, then it is clear that δ is also non-negligible, and the lemma now follows. □

Theorem 27 *If the set of IND-TR-CCA$_{TS}$ secure TRE-PC schemes is non-empty, then IND-TR-CCA$_{TS}$ $\not\rightarrow$ binding.*

Proof. Suppose E is an IND-TR-CCA$_{\mathrm{TS}}$ secure TRE-PC scheme, and generate E' from E using Construction 5. Then E' is IND-TR-CCA$_{\mathrm{TS}}$ secure (by Lemma 25) but is not binding (by Lemma 26). The result follows. □

8.4.5 Relationship between Binding and IND-TR-CPA$_{\mathrm{OS}}$

We prove that binding does not imply IND-TR-CPA$_{\mathrm{OS}}$ for TRE-PC schemes.

Construction 6 *Let E be a TRE-PC scheme with component algorithms as follows.*

1. *The* Setup *algorithm takes a security parameter ℓ as input, and sets the master key $mk = 1$ and the public parameters $param = 1$.*
2. *The* $\mathsf{Gen_U}$ *algorithm takes ℓ as input, and selects a public/private key pair (pk_r, sk_r), where $pk_r = sk_r = 1$. The plaintext and ciphertext space is $\{0,1\}^k$.*
3. *The* $\mathsf{Ext_{TS}}$ *algorithm take mk and a release time t as input, and returns $TS_t = 1$.*
4. *The* Enc *algorithm takes a message m, a release time t and the receiver's public key pk_r as input, and returns a ciphertext $C = m$ and a pre-open key $V_C = 1$.*
5. *The* $\mathsf{Dec_{RK}}$ *algorithm takes a ciphertext C, a pre-open key V_C, and a private key sk_r as input, and returns $m = C$.*
6. *The* $\mathsf{Dec_{PK}}$ *algorithm takes as input a ciphertext C, a timestamp TS_t, and a private key sk_r, and returns $m = C$.*

From the description, it is straightforward to verify that E is binding but not IND-TR-CPA$_{\mathrm{OS}}$ secure.

Theorem 28 *For TRE-PC schemes, binding $\not\rightarrow$ IND-TR-CPA$_{OS}$.*

Moreover, it is clear that, apart from binding, E is not secure in the sense of any of the other defined security notions.

8.4.6 Relationship between IND-TR-CPA$_{\mathsf{IS}}$ and Binding

We prove that IND-TR-CPA$_{\mathrm{IS}}$ does not imply binding for TRE-PC schemes. Note that we have already shown that binding $\not\rightarrow$ IND-TR-CPA$_{\mathrm{IS}}$.

Construction 7 *Suppose that $E = (\mathsf{Setup}, \mathsf{Gen_U}, \mathsf{Ext_{TS}}, \mathsf{Enc}, \mathsf{Dec_{RK}}, \mathsf{Dec_{PK}})$ is a TRE-PC scheme. We construct a TRE-PC scheme E', where the algorithms are defined as follows.*

1. *The algorithms* $\mathsf{Setup}', \mathsf{Gen}'_{\mathsf{U}}, \mathsf{Ext}'_{\mathsf{TS}}, \mathsf{Enc}'$ *are defined in the same way as in* E.
2. *The algorithm* $\mathsf{Dec}'_{\mathsf{RK}}$ *is defined in the same way as* $\mathsf{Dec}_{\mathsf{RK}}$*, except that it returns a random message from the plaintext space when* $\mathsf{Dec}_{\mathsf{RK}}$ *returns* $\perp$.
3. *The algorithm* $\mathsf{Dec}'_{\mathsf{PK}}$ *is defined in the same way as* $\mathsf{Dec}_{\mathsf{PK}}$*, except that it returns a random message from the plaintext space when* $\mathsf{Dec}_{\mathsf{PK}}$ *returns* $\perp$.

Lemma 29 *If* E' *is obtained from an IND-TR-CPA*$_{IS}$ *secure TRE-PC scheme* E *using Construction 7, then* E' *is IND-TR-CPA*$_{IS}$ *secure.*

Proof. Suppose that an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has advantage δ in attacking E'. We show that there exists an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for E, which makes use of $\mathcal{B}$ as a subroutine, that also has advantage δ. Hence, we can conclude that δ is negligible, since E is IND-TR-CPA$_{\mathrm{IS}}$ secure. The attacker $\mathcal{A}_1$ is defined as follows:

1. $\mathcal{A}_1$ receives the public parameters $param$, the public key pk_r, and the private key sk_r.
2. $\mathcal{A}_1$ sets $param' = param$, $pk'_r = pk_r$, and $sk'_r = sk_r$.
3. $\mathcal{A}_1$ executes $\mathcal{B}_1$ with input pk'_r, sk'_r, and $param'$. If $\mathcal{B}_1$ makes a extraction oracle query for time t, then $\mathcal{A}_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_1$. $\mathcal{B}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state'$.
4. $\mathcal{A}_1$ terminates by outputting the messages m_0 and m_1, a release time t^*, and the state information $state = state'$.

The challenger chooses $b \in_R \{0, 1\}$ and compute the challenge TRE-PC encryption C^*, where $(C^*, V_C^*) = \mathsf{Enc}(m_b, t^*, pk_r)$. The attacker $\mathcal{A}_2$ is defined as follows:

1. $\mathcal{A}_2$ receives the challenge ciphertext C^* and the state information $state$.

2. $\mathcal{A}_2$ executes $\mathcal{B}_2$ on the input $(C^*, state)$. If $\mathcal{B}_2$ makes a extraction oracle query for a time $t < t^*$, then $\mathcal{A}_2$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{B}_2$. $\mathcal{B}_2$ eventually terminates by outputting a bit b'.

3. $\mathcal{A}_2$ terminates by outputting the bit b'.

$\mathcal{A}$ provides perfect simulation for the oracles that $\mathcal{B}$ may query, $\mathcal{A}$ is a legitimate IND-TR-CPA$_{\text{IS}}$ attacker, and $\mathcal{A}$'s advantage equals δ. Since E is an IND-TR-CPA$_{\text{IS}}$ secure TRE-PC scheme, δ is negligible, and the lemma follows. □

Lemma 30 *If E' is obtained from a TRE-PC scheme E using Construction 7, then E' is not binding.*

Proof. The proof of this lemma is based on the fact that, since E' is IND-TR-CPA$_{\text{IS}}$ secure (by Lemma 29), a polynomial-time attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ has only a negligible advantage in the following game.

1. Game setup: The challenger runs Setup' to generate the time server's master key mk' and the public system parameters $param'$. The challenger also runs $\mathsf{Gen}'_{\mathsf{U}}$ to generate a public/private key pair (pk'_r, sk'_r).

2. Phase 1: The attacker runs $\mathcal{B}_1$ on the input $(pk'_r, param')$. $\mathcal{B}_1$ has access to the following types of oracles:

   - An oracle for $\mathsf{Ext}'_{\mathsf{TS}}$, which, on receiving a query for time t, returns $\mathsf{Ext}'_{\mathsf{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec}'_{\mathsf{RK}}$, which, on receiving a query for (C, V_C), returns $\mathsf{Dec}'_{\mathsf{RK}}(C, V_C, sk'_r)$.
   - An oracle for $\mathsf{Dec}'_{\mathsf{PK}}$, which, on receiving a query for (C, t), returns $\mathsf{Dec}'_{\mathsf{PK}}(C, TS_t, sk'_r)$.

   $\mathcal{B}_1$ terminates by outputting two equal length messages m_0, m_1 and a release time t^* which is larger than all the inputs to the $\mathsf{Ext}'_{\mathsf{TS}}$ oracle. In addition, $\mathcal{B}_1$ also outputs some state information *state*.

3. Challenge: The challenger picks $d \in_R \{0,1\}$, computes $(C^*, V_{C^*}) = \mathsf{Enc}(m_d, t^*, pk'_r)$, and returns C^*.

4. Phase 2: The attacker runs $\mathcal{B}_2$ on the input $(C^*, state)$. $\mathcal{B}_2$ has access to the same types of oracles as $\mathcal{B}_1$. However, $\mathcal{B}_2$ may not make a $\mathsf{Ext}'_{\mathsf{TS}}$ query on a time $t \geq t^*$ and a $\mathsf{Dec}'_{\mathsf{PK}}$ query on the input (C, t), where $t \geq t^*$. $\mathcal{B}_2$ terminates by outputting a guess bit $d' \in \{0,1\}$.

In this attack game, instead of taking the private key sk'_r as an input as required by a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attack game, the attacker is granted access to the decryption oracles. Since E' is IND-TR-CPA$_{\mathrm{IS}}$ secure, it follows that the attacker's advantage in the above game (i.e. $|\Pr[d' = d] - \frac{1}{2}|$) is negligible.

We can now construct a binding attacker $\mathcal{A}'$ and an algorithm $\mathcal{A}$ identical to those in the proof of Lemma 26. The proof is also exactly the same as that of Lemma 26, except that in this case the probability $|\Pr[b = b'] - \frac{1}{2}|$ is negligible because $\mathcal{B}$ has only a negligible advantage, and the lemma now follows. □

Theorem 31 *If the set of IND-TR-CPA$_{IS}$ secure TRE-PC schemes is non-empty, then IND-TR-CPA$_{IS}$ $\not\rightarrow$ binding.*

Proof. Suppose E is an IND-TR-CPA$_{\mathrm{IS}}$ secure TRE-PC scheme, and generate E' from E using Construction 7. Then E' is IND-TR-CPA$_{\mathrm{IS}}$ secure (by Lemma 29) but is not binding (by Lemma 30). The result follows. □

8.5 Conclusions

In this chapter we have refined the Hwang-Yum-Lee model for TRE-PC schemes. We have also established the relationships between the security notions defined in the new model.

Chapter 9

Hybrid construction of TRE-PC Schemes

Contents

In this chapter we define a novel type of KEM, which we call a TRE-PC KEM, and propose an implementation of this new primitive. We then propose a hybrid construction for TRE-PC schemes based on a TRE-PC KEM and a DEM.

9.1 Motivation

The use of a symmetric encryption scheme as a subroutine of an asymmetric encryption scheme has long been known as a useful technique for improving the efficiency of asymmetric encryption. Cramer and Shoup [84] formalised one approach, namely the KEM-DEM paradigm (KEM and DEM are defined in Section 2.2.5), for producing such hybrid asymmetric encryption schemes. In this paradigm, the KEM is used to encrypt/decrypt a symmetric key, and the DEM is used to encrypt/decrypt data using the symmetric key. This KEM-DEM approach has subsequently been

applied to various other branches of asymmetric cryptography (see, for example, [39, 41, 89]).

In this chapter we propose a similar paradigm to construct TRE-PC schemes, where the main difference is that the KEM is replaced with a new primitive, which we call a TRE-PC KEM, which is a special type of KEM with timed-release and pre-open capabilities.

The rest of this chapter is organised as follows. In Section 9.2 we define the new TRE-PC KEM primitive, and also propose an instantiation of this primitive. In Section 9.3 we propose a hybrid paradigm and prove relevant security claims. In the last section we conclude the chapter.

9.2 Definition and Instantiation of TRE-PC KEM

We define a TRE-PC KEM using the same types of principals as those used in the definition of TRE-PC schemes (defined in 8.3.1). The following two kinds of entities are involved in a TRE-PC scheme:

- The users, each of which may act as both a sender and a receiver.
- A trusted time server, which is required to publish timestamps periodically. We assume that the time server acts correctly in generating its parameters and publishing the timestamps. However, when discussing semantic security, we take into account the fact that the time server may be curious, i.e. it may try to decapsulate the ciphertext (where decapsulation is formally defined below). Apart from this, the time server will do nothing else malicious.

In addition, we consider the following four types of attacker:

- An outside attacker which does not know the master key of the time server. In the rest of this paper, the term *outside attacker* refers to this kind of attacker,

while the term *curious time server* (see below) refers to the special kind of outside attacker which knows the master key of the time server.

- A curious time server which knows the master key of the time server.
- Authorised but curious receivers which try to decapsulate the ciphertext before the release time without the pre-open key.
- Authorised but malicious senders which try to make the receiver decapsulate a message different from that which was originally sent.

A TRE-PC KEM consists of the following polynomial-time algorithms:

- TRE-PC-KEM.Setup: Run by the time server, the setup algorithm takes a security parameter ℓ as input, and generates a secret master-key mk and the public parameters $param$.
- TRE-PC-KEM.$\mathsf{Ext}_{\mathsf{TS}}$: Run by the time server, the timestamp extraction algorithm takes mk and a time t as input, and generates a timestamp TS_t.
- TRE-PC-KEM.Gen: Run by a user, the key generation algorithm takes a security parameter ℓ as input, and outputs a public/private key pair (pk_r, sk_r).
- TRE-PC-KEM.Encap: Run by the message sender, the key encapsulation algorithm takes a release time t and a public key pk_r as input, and outputs (K, C, V_C), where K is a symmetric key, C is ciphertext, and V_C is the pre-open key for C. We assume that the bit length of K is a polynomial function of ℓ, $\mathsf{KenLen}(\ell)$ say.
- TRE-PC-KEM.$\mathsf{Decap}_{\mathsf{RK}}$: Run by the receiver, the (pre-open) decapsulation algorithm takes ciphertext C, a pre-open key V_C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.
- TRE-PC-KEM.$\mathsf{Decap}_{\mathsf{PK}}$: Run by the receiver, the (standard) decapsulation algorithm takes ciphertext C, a timestamp TS_t which is determined by the release time associated with C, and the receiver's private key sk_r as input, and returns either the encapsulated key K or an error message $\perp$.

Note that *param* is an implicit input to all the algorithms except for TRE-PC-KEM.Setup.

9.2.1 Security Definitions

For a TRE-PC KEM, we consider the same types of attackers as for a TRE-PC scheme, and, analogously, we have the following security definitions.

9.2.1.1 Soundness of a TRE-PC KEM

Informally, the decapsulation algorithms of a sound TRE-PC KEM should always "undo" the output of the encapsulation algorithm. Formally, soundness is defined as follows.

Definition 44 *A TRE-PC KEM is sound if, for any time t and (K, C, V_C) where*

$$(K, C, V_C) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t, pk_r),$$

the following two requirements are satisfied

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}(C, V_C, sk_r),$$

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}(C, TS_t, sk_r).$$

9.2.1.2 Binding of a TRE-PC KEM

Analogously to the definition given in 8.3.2.2, we give the following definition of the *binding* property.

Definition 45 *A TRE-PC KEM is binding if any polynomial-time attacker $\mathcal{A}$ has only a negligible probability of winning the following game.*

1. Game setup: The challenger runs TRE-PC-KEM.Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs TRE-PC-KEM.Gen to generate a public/private key pair (pk_r, sk_r).

2. Challenge: The attacker $\mathcal{A}$ executes with input $(pk_r, param)$. At some point, $\mathcal{A}$ generates a ciphertext C^* for release at time t^* and a pre-open key V_{C^*}, and then terminates by outputting (C^*, t^*, V_{C^*}). During its execution, $\mathcal{A}$ has access to the following oracles:

   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}(mk, t)$.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$, which, on receiving a query for (C, t'), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_{t'}, sk_r)$. Note that t' may not be the release time for C.

In this game, $\mathcal{A}$ wins if $O_1 \neq \perp$, $O_2 \neq \perp$, and $O_1 \neq O_2$, where

$$O_1 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C^*, V_{C^*}, sk_r),$$

$$O_2 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C^*, TS_{t^*}, sk_r).$$

9.2.1.3 Security against malicious outsiders

We define semantic security for a TRE-PC KEM against outside attackers which do not know the time server's master key. Specifically, we define semantic security under an adaptive chosen ciphertext attack (IND-TR-KEM-CCA$_{\mathrm{OS}}$ security) and semantic security under an adaptive chosen plaintext attack (IND-TR-KEM-CPA$_{\mathrm{OS}}$ security).

Definition 46 *A TRE-PC KEM is IND-TR-KEM-CCA$_{OS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.

   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}(mk, t)$.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$, which, on receiving a query for (C, V'_C), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$, which, on receiving a query for (C, t'), returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_{t'}, sk_r)$. Note that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information $state$.

3. Challenge: The challenger generates the challenge as follows:

   (a) The challenger computes $(K_0, C^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$.

   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, where $\mathsf{KeyLen}(\ell)$ is a polynomial function of ℓ.

   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (K_b, C^*, V_{C^*}).

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracle as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not permitted to make a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input (C^*, t^*) or a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

In this game the attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 47 *A TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure if it is IND-TR-KEM-CCA$_{OS}$ secure against attackers that make no decryption queries.*

9.2.1.4 Security against a curious time server

In this subsection we define semantic security against a curious time server for a TRE-PC KEM. Specifically, we define semantic security under an adaptive chosen ciphertext attack (IND-TR-KEM-CCA$_{\mathrm{TS}}$ security) and semantic security under an adaptive chosen plaintext attack (IND-TR-KEM-CPA$_{\mathrm{TS}}$ security).

Definition 48 *A TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure, if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs TRE-PC-KEM.Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs TRE-PC-KEM.Gen to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles.
   - An oracle for TRE-PC-KEM.Decap$_{\mathsf{RK}}$, which, on receiving a query for (C, V'_C), returns TRE-PC-KEM.Decap$_{\mathsf{RK}}(C, V'_C, sk_r)$. Note that C and V'_C may have no relationship with each other, i.e. V'_C may not be the pre-open key for C.
   - An oracle for TRE-PC-KEM.Decap$_{\mathsf{PK}}$, which, on receiving a query for (C, t'), returns TRE-PC-KEM.Decap$_{\mathsf{PK}}(C, TS_{t'}, sk_r)$. Note that t' need not be the legitimate release time for C.

   $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information $state$.
3. Challenge: The challenger generates the challenge as follows:
   (a) The challenger computes $(K_0, C^*, V_{C^*}) =$ TRE-PC-KEM.Encap(t^*, pk_r).
   (b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.
   (c) The challenger randomly selects a bit $b \in \{0,1\}$, and returns (K_b, C^*, V_{C^*}).

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracle as $\mathcal{A}_1$. However, $\mathcal{A}_2$ is not permitted to make a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input (C^*, t^*), or a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputting a guess bit b'.

In this game the attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

Definition 49 *A TRE-PC KEM is IND-TR-KEM-CPA$_{TS}$ secure if it is IND-TR-KEM-CCA$_{TS}$ secure against attackers that make no decryption queries.*

9.2.1.5 Security against a malicious receiver

Semantic security against a malicious receiver, i.e. IND-TR-KEM-CPA$_{\mathrm{IS}}$ security, is defined as follows.

Definition 50 *A TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure if any two-stage polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has only a negligible advantage in the following game.*

1. Game setup: The challenger runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to an oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$, which, on receiving a query for time t, returns $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting a release time t^* which is larger than all the inputs to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ oracle, and some state information $state$.
3. Challenge: The challenger generates the challenge as follows:
   (a) The challenger computes $(K_0, C^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$.

(b) The challenger randomly selects $K_1 \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$.

(c) The challenger randomly selects a bit b, and returns (K_b, C^*).

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(K_b, C^*, state)$. $\mathcal{A}_2$ has access to an oracle for $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$ for any input $t < t^*$. $\mathcal{A}_2$ eventually terminates by outputting a guess bit b'.

In this game the attacker wins if $b' = b$, and its advantage is defined to be $|\Pr[b = b'] - \frac{1}{2}|$.

9.2.2 An Instantiation for TRE-PC KEM

The polynomial-time algorithms of the proposed TRE-PC KEM are defined as follows.

- $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$: The algorithm takes a security parameter ℓ as input, and generates the following parameters:
  - an additive group $\mathbb{G}_1$ of prime order q, a generator P of $\mathbb{G}_1$, and a multiplicative group $\mathbb{G}_2$ of the same order as $\mathbb{G}_1$,
  - a polynomial-time computable bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$,
  - three cryptographic hash functions $\mathsf{H_1} : \{0,1\}^* \to \mathbb{G}_1$, $\mathsf{H_2} : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^{\ell}$, and $\mathsf{H_3} : \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^{\mathsf{KeyLen}(\ell)}$,
  - a public/private key pair (S, s), where $S = sP$ and s is randomly chosen from $\mathbb{Z}_q$,

  where the master secret is $mk = s$ and the public parameters are

$$param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H_1}, \mathsf{H_2}, \mathsf{H_3}).$$

- $\mathsf{TRE\text{-}PC\text{-}KEM.Ext_{TS}}$: The algorithm takes the master secret mk and a time t as input and returns $TS_t = s\mathsf{H_1}(t)$.
- $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$: The algorithm takes ℓ as input, randomly chooses sk_r from $\mathbb{Z}_q$, and generates a public/private key pair (pk_r, sk_r) where $pk_r = sk_r P$.

- TRE-PC-KEM.Encap: The algorithm takes a release time t and the receiver's public key pk_r as input, and returns (K, C, V_C), which are computed as follows:

  1. Randomly choose r and v from $\mathbb{Z}_q$, and compute $Q_t = \mathsf{H}_1(t)$, $C_1 = rP$, $C_2 = vP$, $X_1 = r \cdot pk_r$, and $X_2 = \hat{e}(vS, Q_t)$,
  2. Compute $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ and $K = \mathsf{H}_3(X_1, X_2)$,
  3. Set $V_C = vQ_t$ and $C = (C_1, C_2, C_3)$.

- TRE-PC-KEM.$\mathsf{Dec}_{\mathsf{RK}}$: The algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the pre-open key $V_C = vQ_t$, and the private key sk_r as input, and runs as follows:

  1. Compute $X_1 = sk_r C_1$ and $X_2 = \hat{e}(S, V_C)$, and check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  2. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return an error message $\perp$.

- TRE-PC-KEM.$\mathsf{Dec}_{\mathsf{PK}}$: This algorithm takes a ciphertext $C = (C_1, C_2, C_3)$, the timestamp TS_t, and the private key sk_r as input, and runs as follows:

  1. Compute $X_1 = sk_r C_1$ and $X_2 = \hat{e}(C_2, TS_t)$, and check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds,
  2. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return an error message $\perp$.

9.2.3 Security Results

It is straightforward to verify that the TRE-PC KEM is sound. We next prove that the TRE-PC KEM is secure in the sense of binding, IND-TR-KEM-CCA$_{\mathrm{TS}}$, and IND-TR-KEM-CPA$_{\mathrm{IS}}$. Then from the security definitions, it is straightforward to verify that the proposed scheme is also secure in the sense of the other security notions.

Theorem 32 *If* H_2 *is collision-resistant, then the TRE-PC KEM is binding.*

Proof. Without loss of generality, suppose that at the end of the legitimate binding attack game the attacker outputs (C^*, t^*, V_{C^*}), where $C^* = (C_1^*, C_2^*, C_3^*)$. Recalling the definitions of $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}$ and $\mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}$ from the previous section, the attacker wins the game only if $O_1 \neq O_2$, where

$$O_1 = \mathsf{H_3}(X_1, X_2'),\ O_2 = \mathsf{H_3}(X_1, X_2''),\ X_1 = sk_r C_1^*,\ X_2' = \hat{e}(S, V_{C^*}),$$

$$X_2'' = \hat{e}(C_2^*, TS_{t^*}),\ C_3^* = \mathsf{H_2}(C_2^*, X_1, X_2'),\ \text{and}\ C_3^* = \mathsf{H_2}(C_2^*, X_1, X_2'').$$

If the attacker wins, then it is staightforward to verify that $X_2' \neq X_2''$; otherwise $O_1 = O_2$. Hence, if the attacker wins the game then this implies that the attacker can find a collision for $\mathsf{H_2}$, where the two inputs are (C_2^*, X_1, X_2') and (C_2^*, X_1, X_2''). Under the assumption that $\mathsf{H_2}$ is collision-resistant, it follows that the attacker can only win the game with a negligible probability. □

Theorem 33 *The TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure in the random oracle model under the CDH assumption.*

Proof. We prove the theorem using a sequence of games.

$\mathsf{Game_0}$: this game is a legitimate IND-TR-KEM-CCA$_{\mathrm{TS}}$ attack game, where the hash functions are random oracles.

1. Game setup: The challenger first runs $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ to generate $mk = s$ and $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H_1}, \mathsf{H_2}, \mathsf{H_3})$, and then runs $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ to generate a public/private key pair (pk_r, sk_r). The challenger simulates the random oracle $\mathsf{H_1}$ as follows. The challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}_1$ (the hash-code for this message), and an element of $\mathbb{Z}_q$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}_1$; otherwise, the challenger returns yP, where y is a randomly chosen element of $\mathbb{Z}_q$, and stores a new vector in the list. $\mathsf{H_2}$ and $\mathsf{H_3}$ are simulated in a similar way.

2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(mk, pk_r, param)$. $\mathcal{A}_1$ has access to the following decapsulation oracles:

   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ oracle on the input (C, V_C), where $C = (C_1, C_2, C_3)$. The challenger returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V_C, sk_r)$.
   - $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ oracle on the input (C, t), where $C = (C_1, C_2, C_3)$. Since the challenger knows s, it computes $TS_t = s\mathsf{H}_1(t)$ and returns $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, TS_t, sk_r)$.

   At some point, $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information $state$.

3. Challenge: The challenger generates the challenge as follows:

   (a) Randomly choose r^* and v^* from $\mathbb{Z}_q$, and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,

   (b) Compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_1^* = r^*P$, $C_2^* = v^*P$, $X_1^* = r^* \cdot pk_r$, $X_2^* = \hat{e}(v^*S, Q_{t^*})$, $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$, and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$,

   (c) Return (K_b, C^*, V_{C^*}), where b is randomly chosen from $\{0,1\}$, $C^* = (C_1^*, C_2^*, C_3^*)$, and $V_{C^*} = v^*Q_{t^*}$.

4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(K_b, C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracle as $\mathcal{A}_1$, but is not permitted to make a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query on the input (C^*, t^*) and a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C^*, V_{C^*}). $\mathcal{A}_2$ terminates by outputting a guessing bit b'.

Game_1: this game is identical to Game_0 except that the decapsulation queries are answered as follows.

- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V_C)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

  1. Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(S, V_C)$,
  2. If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds.
  3. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, t)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

  1. Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(C_2, TS_t)$,
  2. If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds.
  3. If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

Let E_0 and E_1 denote the events that $b = b'$ at the end of Game_0 and Game_1, respectively. We next show that $|\Pr[E_0] - \Pr[E_1]|$ is negligible in the random oracle model.

Let F_1 be the event that the challenger answers a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query on the input (C, V_C) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(S, V_C)$. Recall that, for a valid query, either $C \neq C^*$ or $V_C \neq V_{C^*}$ should hold; therefore, at least one of the following inequalities should hold: $C_1 \neq C_1^*$, $C_2 \neq C_2^*$, $C_3 \neq C_3^*$, and $V_C \neq V_{C^*}$. Given a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ query with input of (C, V_C), where the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(S, V_C)$, and given the assumption that H_2 is a random oracle, we can deduce the following:

1. If $C_1 \neq C_1^*$, then $X_1 \neq X_1^*$ and $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with probability $\frac{1}{2^\ell}$.
2. If $C_2 \neq C_2^*$, then $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with probability $\frac{1}{2^\ell}$.
3. If $V_C \neq V_{C^*}$, then $X_2 \neq X_2^*$ and $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with probability $\frac{1}{2^\ell}$.
4. If $C_3 \neq C_3^*$, then $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ occurs with probability at most $\frac{1}{2^\ell}$.

As a result, in the presence of a polynomial-time attacker, F_1 occurs with a negligible probability in Game_0, i.e. $\Pr[F_1]$ is negligible.

Let F_2 be the event that the challenger answers a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ query with input of (C, t) with a valid message, where $C = (C_1, C_2, C_3)$ and the attacker makes no H_2 query on the input (z_1, z_2, z_3) satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(C_2, TS_t)$. For similar reasons, $\Pr[F_2]$ is negligible in Game_0.

It is clear that Game_0 and Game_1 perform identically unless one of the events F_1 or F_2 occurs; therefore, $\Pr[E_0|\neg(F_1 \vee F_2)] = \Pr[E_1|\neg(F_1 \vee F_2)]$. We thus have $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1 \vee F_2]$. Since $\Pr[F_1]$ and $\Pr[F_2]$ are both negligible, $|\Pr[E_0] - \Pr[E_1]|$ is also negligible.

Game_2: this game is identical to Game_1, except that the challenger randomly selects C_3^* from $\{0,1\}^\ell$ and K_0 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$ instead of computing $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$ and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$. Let E_2 be the event that $b = b'$ at the end of Game_2.

We next show that $|\Pr[E_1] - \Pr[E_2]|$ is negligible in the random oracle model under the CDH assumption..

It is clear that Game_2 and Game_1 perform identically unless H_2 is queried with input $(C_2^*, r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$ or H_3 is queried with input $(r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$. Let F_3 denote that the event that either of these events occurs. We now construct an algorithm $\mathcal{A}'$, which makes use of an IND-TR-KEM-CCA$_{\mathrm{TS}}$ attacker $\mathcal{A}$ as a subroutine, that solves the CDH problem with a non-negligible probability if $\Pr[F_3]$ is non-negligible.

The attacker $\mathcal{A}'$ plays the same role as the challenger plays in Game_2:

1. $\mathcal{A}'$ receives the parameters $(\mathbb{G}_1, \mathbb{G}_2, q, \hat{e})$ and a v-CDH challenge (P, aP, bP), and generates $(\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, a public/private key pair (S, s), and sets $pk_r = aP$. $\mathcal{A}'$ sets $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and simulates the random oracles in the same way as the challenger in Game_2.
2. $\mathcal{A}'$ runs $\mathcal{A}_1$ on the input of $(mk, pk_r, param)$ and answers $\mathcal{A}_1$'s decapsulation queries as follows.
   - On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C, V_C)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

(a) Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(S, V_C)$,

(b) If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds.

(c) If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

- On receiving a $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C, t)$ query, where $C = (C_1, C_2, C_3)$, the challenger computes the response as follows.

  (a) Check whether there is an input (z_1, z_2, z_3) to H_2 satisfying $z_1 = C_2$, $\hat{e}(C_1, pk_r) = \hat{e}(z_2, P)$, and $z_3 = \hat{e}(C_2, TS_t)$,

  (b) If the check fails, return $\perp$; otherwise, set $X_1 = z_2, X_2 = z_3$ and continue to check whether $C_3 = \mathsf{H}_2(C_2, X_1, X_2)$ holds.

  (c) If the check succeeds, return $K = \mathsf{H}_3(X_1, X_2)$; otherwise, return $\perp$.

3. Suppose $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information *state*. $\mathcal{A}'$ computes the challenge as follows:

   (a) Randomly choose v^* from $\mathbb{Z}_q$, C_3^* from $\{0,1\}^\ell$, and K_0 and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$.

   (b) Set $C_1^* = bP$ and compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_2^* = v^*P$, and $X_2^* = \hat{e}(v^*S, Q_{t^*})$.

   (c) Return (K_b, C^*, V_{C^*}), where b is randomly chosen from $\{0,1\}$, $C^* = (C_1^*, C_2^*, C_3^*)$, and $V_{C^*} = vQ_{t^*}$.

4. $\mathcal{A}'$ runs $\mathcal{A}_2$ on the input of $(K_b, C^*, V_{C^*}, state)$, and answers $\mathcal{A}_2$'s decapsulation queries in the same way as in step 2.

5. After $\mathcal{A}_2$ terminates, $\mathcal{A}'$ first randomly selects an input from the input set composed of the following two types of inputs: the inputs to H_2 in the form $(C_2^*, ?, \hat{e}(v^*S, Q_{t^*}))$, and the inputs to H_3 in the form $(?, \hat{e}(v^*S, Q_{t^*}))$, where ? can be any element from $\mathbb{G}_1$. If an input to H_2, say $(C_2^*, w_1', \hat{e}(v^*S, Q_{t^*}))$, is chosen, $\mathcal{A}'$ sets $\lambda = w_1'$; otherwise, if an input to H_3, say $(w_2', \hat{e}(v^*S, Q_{t^*}))$ is chosen, then $\lambda = w_2'$. $\mathcal{A}'$ terminates by outputting λ.

It is straightforward to verify that the algorithm $\mathcal{A}'$ faithfully plays the role that the challenger will play in Game_2. Suppose n_1 oracle queries have been made to H_2 and n_2 oracle queries have been made to H_3, where the queries are in the form specified in step 5 of $\mathcal{A}'$. Note that the queries to H_3 are made either directly by $\mathcal{A}$ or by $\mathcal{A}'$

in simulating the decapsulation oracles; however, the queries to H_2 are all made by $\mathcal{A}$. It follows that $\Pr[\lambda = abP] = \frac{1}{n_1+n_2}\Pr[F_3]$, and hence $\Pr[F_3]$ is negligible from the CDH assumption. Next observe that Game_2 and Game_1 will perform identically unless the event F_3 occurs. We then have $|\Pr[E_1] - \Pr[E_2]| \leq \Pr[F_3]$, and hence $|\Pr[E_1] - \Pr[E_2]|$ is negligible.

It is straightforward to verify that $|\Pr[E_2] - \frac{1}{2}| = 0$ in Game_2, because both K_0 and K_1 are randomly chosen. Therefore, it follows that $|\Pr[E_0] - \Pr[E_1]|$, $|\Pr[E_1] - \Pr[E_2]|$, and $|\Pr[E_2] - \frac{1}{2}|$ are all negligible. As a result, $|\Pr[E_0] - \frac{1}{2}|$ is also negligible, and the theorem follows. □

Theorem 34 *The TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure in the random oracle model under the BDH assumption.*

Proof. We prove the theorem using a sequence of games.

Game_0: this game is a legitimate IND-TR-KEM-CPA$_{\mathrm{IS}}$ attack game, where hash functions are modelled as random oracles.

1. Game setup: The challenger first runs TRE-PC-KEM.Setup to generate $mk = s$ and $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, \hat{e}, S, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and then runs TRE-PC-KEM.Gen to generate a public/private key pair (pk_r, sk_r). The challenger simulates the random oracle H_1 as follows: The challenger maintains a list of vectors, each of them containing a request message, an element of $\mathbb{G}_1$ (the hash-code for this message), and an element of $\mathbb{Z}_q$. After receiving a request message, the challenger first checks its list to see whether the request message is already in the list. If the check succeeds, the challenger returns the stored element of $\mathbb{G}_1$; otherwise, the challenger returns yP, where y a randomly chosen element of $\mathbb{Z}_q$, and stores the new vector in the list. H_2 and H_3 are simulated in a similar way.

2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. If $\mathcal{A}_1$ makes an extraction oracle query for a time t, the challenger returns the timestamp

TS_t to $\mathcal{A}_1$. At some point, $\mathcal{A}_1$ terminates by outputting a release time t^*, which is larger than all the inputs to the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ oracle, and some state information *state*.

3. Challenge: The challenger generates the challenge as follows.
   (a) Randomly choose r^* and v^* from $\mathbb{Z}_q$, and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$,
   (b) Compute $Q_{t^*} = \mathsf{H}_1(t^*)$, $C_1^* = r^*P$, $C_2^* = v^*P$, $X_1^* = r^* \cdot pk_r$, $X_2^* = \hat{e}(v^*S, Q_{t^*})$, $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$, and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$,
   (c) Return (K_b, C^*), where b is randomly chosen from $\{0,1\}$ and $C^* = (C_1^*, C_2^*, C_3^*)$.
4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(K_b, C^*, state)$. If $\mathcal{A}_2$ makes an extraction oracle query for a time $t < t^*$, the challenger returns the timestamp TS_t to $\mathcal{A}_2$. $\mathcal{A}_2$ eventually terminates by outputting a bit b'.

Let E_0 be the event that $b = b'$ at the end of the game.

Game_1: this game is identical to Game_0, except that the challenger randomly selects C_3^* from $\{0,1\}^\ell$, and K_0 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$ instead of computing $C_3^* = \mathsf{H}_2(C_2^*, X_1^*, X_2^*)$ and $K_0 = \mathsf{H}_3(X_1^*, X_2^*)$. Let E_1 be the event that $b = b'$ at the end of Game_1. We next show that $|\Pr[E_0] - \Pr[E_1]|$ is negligible in the random oracle model under the BDH assumption.

It is clear that Game_1 and Game_0 perform identically unless H_2 is queried on the input of $(C_2^*, r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$ or H_3 is queried on the input of $(r^* \cdot pk_r, \hat{e}(v^*S, Q_{t^*}))$. Let F_1 denote the event that either of the these events occurs. We now construct an algorithm $\mathcal{A}'$, which makes use of an IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, that solves the BDH problem with a non-negligible probability if $\Pr[F_1]$ is non-negligible.

Without loss of generality, we assume that the total number of H_1 queries $\mathcal{A}_1$ may make is bounded by n_1 ($n_1 \geq 1$)[1]. Note that these queries do not include those which are indirectly caused by the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ queries. The attacker $\mathcal{A}'$

[1]For simplicity of description, it is reasonable to require that $\mathcal{A}_1$ query H_1 only once with the same input.

is implemented to play the same role as the challenger in Game_1 and is defined as follows:

1. $\mathcal{A}'$ receives the parameters $(\mathbb{G}_1, \mathbb{G}_2, q, \hat{e})$ and a BDH challenge (P, aP, bP, cP), and generates $(\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, a public/private key pair (S, s) and sets $S = aP$. $\mathcal{A}'$ sets $param = (\mathbb{G}_1, \mathbb{G}_2, q, P, S, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$, and simulates H_1, H_2, and H_3 in the same way as the challenger does in Game_1. In addition, $\mathcal{A}'$ randomly selects $j \in \{1, 2, \cdots, n_1 + 1\}$.

2. $\mathcal{A}'$ runs $\mathcal{A}_1$ on the input of $(pk_r, sk_r, param)$. If $1 \leq j \leq n_1$, $\mathcal{A}'$ answers $\mathcal{A}_1$'s j-th query $\mathsf{H}_1(t')$ with cP, and stores $(t', cP, null)$. If $\mathcal{A}_1$ makes a $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ query for a time t, $\mathcal{A}'$ first checks whether $t \neq t'$. If the check succeeds, $\mathcal{A}'$ computes and returns the timestamp TS_t to $\mathcal{A}_1$; otherwise, $\mathcal{A}'$ terminates as a failure. Note that if $t \neq t'$ then $\mathsf{H}_1(t) = yP$ for some y, where y is known to $\mathcal{A}'$. Therefore, $\mathcal{A}'$ can compute $TS_t = yaP$, although a is unknown.

3. Suppose $\mathcal{A}_1$ terminates by outputting a release time t^* and some state information $state$. If one of the following events occurs, $\mathcal{A}_1$ terminates as a failure.

   (a) t^* has been queried to H_1 as the i-th query and $i \neq j$,

   (b) t^* has not been queried to H_1 and $1 \leq j \leq n_1$.

   If $j = n_1 + 1$ and t^* has not been queried to H_1, $\mathcal{A}'$ sets $\mathsf{H}_1(t^*) = cP$. $\mathcal{A}'$ then computes the challenge as follows:

   (a) Randomly choose r^* from $\mathbb{Z}_q$, C_3^* from $\{0,1\}^{\ell}$, and K_0 and K_1 from $\{0,1\}^{\mathsf{KeyLen}(\ell)}$.

   (b) Compute $C_1^* = r^*P$ and set $C_2^* = bP$.

   (c) Return (K_b, C^*), where b is randomly chosen from $\{0,1\}$ and $C^* = (C_1^*, C_2^*, C_3^*)$.

4. $\mathcal{A}'$ runs $\mathcal{A}_2$ on the input of $(K_b, C^*, state)$. If $\mathcal{A}_2$ makes a $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ query on the input t, $\mathcal{A}'$ computes and returns the timestamp TS_t.

5. After $\mathcal{A}_2$ terminates, $\mathcal{A}'$ first randomly selects an input from the input set which is composed of the following two types of inputs: the inputs to H_2 in the form $(C_2^*, r^* \cdot pk_r, ?)$ and the inputs to H_3 in the form $(r^* \cdot pk_r, ?)$, where ? can be any element from $\mathbb{G}_2$. If an input to H_2, say $(C_2^*, r^* \cdot pk_r, w_1')$, is chosen,

$\mathcal{A}'$ sets $\lambda = w_1'$; otherwise, if an input to H_3, say $(r^* \cdot pk_r, w_2')$ is chosen then $\lambda = w_2'$. $\mathcal{A}'$ terminates by outputting λ.

It is straightforward to verify that the probability that $\mathcal{A}'$ successfully ends is $\frac{1}{n_1+1}$, i.e. the probability that $\mathcal{A}'$ does not terminate in step 3 is $\frac{1}{n_1+1}$. If $\mathcal{A}'$ successfully ends, then it faithfully plays the role that the challenger will play in Game_1. Suppose n_1 oracle queries have been made to H_2 and n_2 oracle queries have been made to H_3, where the queries are in the form specified in step 5 of $\mathcal{A}'$. Note that these queries are all made by $\mathcal{A}$. Therefore, if $\mathcal{A}'$ successfully ends, the probability that $\lambda = \hat{e}(P,P)^{abc}$ holds is $\frac{\Pr[F_1]}{n_2+n_3}$, so that the probability that $\mathcal{A}'$ can compute $\hat{e}(P,P)^{abc}$ is $\frac{\Pr[F_1]}{(n_1+1)(n_2+n_3)}$. From the BDH assumption, we can deduce that $\Pr[F_1]$ is negligible. Recall that Game_1 and Game_0 perform identically unless the event F_1 occurs. We then have $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1]$, and $|\Pr[E_0] - \Pr[E_1]|$ is negligible.

It is straightforward to verify that $|\Pr[E_1] - \frac{1}{2}| = 0$ in Game_1 because both K_0 and K_1 are randomly chosen. Therefore, it follows that $|\Pr[E_0] - \Pr[E_1]|$ and $|\Pr[E_1] - \frac{1}{2}|$ are both negligible. Therefore, $|\Pr[E_0] - \frac{1}{2}|$ is also negligible, and the theorem follows. □

9.3 Hybrid Construction of TRE-PC Schemes

In this section we first propose a hybrid method to build a TRE-PC scheme using a TRE-PC KEM and a DEM, and then discuss the security of the resulting TRE-PC scheme.

9.3.1 The Construction

The polynomial-time algorithms of the TRE-PC scheme are as follows.

- The Setup algorithm is the same as the $\mathsf{TRE\text{-}PC\text{-}KEM.Setup}$ algorithm.
- The $\mathsf{Gen_U}$ algorithm is the same as the $\mathsf{TRE\text{-}PC\text{-}KEM.Gen}$ algorithm.

- The $\mathsf{Ext}_{\mathsf{TS}}$ algorithm is the same as the $\mathsf{TRE\text{-}PC\text{-}KEM.Ext}_{\mathsf{TS}}$ algorithm.
- The Enc algorithm: Taking a message m, a release time t, and the receiver's public key pk_r as input, this algorithm returns a ciphertext $C = (C_1, C_2)$ and a pre-open key V_C, where

$$(K, C_1, V_C) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t, pk_r), \text{ and } C_2 = \mathsf{DEM.Enc}(m, K).$$

- The $\mathsf{Dec}_{\mathsf{RK}}$ algorithm: Taking a ciphertext $C = (C_1, C_2)$, a pre-open key V_C, and the private key sk_r as input, this algorithm first computes K, where

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C_1, V_C, sk_r).$$

  If $K = \perp$, the algorithm returns $\perp$. Otherwise, it returns m, where

$$m = \mathsf{DEM.Dec}(C_2, K),$$

- The $\mathsf{Dec}_{\mathsf{PK}}$ algorithm: Taking a ciphertext $C = (C_1, C_2)$, the timestamp TS_t, and the private key sk_r as input, this algorithm first computes K, where

$$K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C_1, TS_t, sk_r).$$

  If $K = \perp$, the algorithm returns $\perp$. Otherwise, it returns m, where

$$m = \mathsf{DEM.Dec}(C_2, K),$$

9.3.2 Security Results

It is straightforward to verify that if the TRE-PC KEM and DEM are sound then the TRE-PC scheme is also sound. In the following security analysis, we make the implicit assumption that the TRE-PC KEM and DEM are sound.

Theorem 35 *If the TRE-PC KEM is binding, then the TRE-PC scheme is binding.*

Proof. Suppose $\mathcal{A}$ is a binding attacker for the TRE-PC scheme. We construct a binding attacker $\mathcal{A}'$ for the TRE-PC KEM which makes use of $\mathcal{A}$ as a subroutine.

The attacker $\mathcal{A}'$ is defined as follows:

1. $\mathcal{A}'$ receives the public key pk_r and the public parameters $param$ as input, and runs $\mathcal{A}$ on the input $(pk_r, param)$.

   - If $\mathcal{A}$ makes an extraction oracle query for a time t, then $\mathcal{A}'$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the $\mathsf{Dec}_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}'$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ oracle on (C_1, V_C). If it receives $\perp$, $\mathcal{A}'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}$.
   - If $\mathcal{A}$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}'$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ oracle on (C_1, t). If it receives $\perp$, $\mathcal{A}'$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}$.

   $\mathcal{A}$ terminates by outputting (C^*, V_{C^*}, t^*), where $C^* = (C_1^*, C_2^*)$.

2. $\mathcal{A}'$ terminates by outputting (C_1^*, V_{C^*}, t^*).

It is straightforward to verify that $\mathcal{A}'$ is a legitimate binding attacker for the TRE-PC KEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. Let E_1 be the event that, given (C^*, V_{C^*}, t^*), the decryption algorithms of the TRE-PC scheme produce different non-error messages, i.e. $O_1 \neq \perp$, $O_2 \neq \perp$, and $O_1 \neq O_2$, where

$$O_1 = \mathsf{Dec}_{\mathsf{RK}}(C^*, V_{C^*}, sk_r) \text{ and } O_2 = \mathsf{Dec}_{\mathsf{PK}}(C^*, TS_{t^*}, sk_r).$$

Let E_2 be the event that given (C_1^*, V_{C^*}, t^*) the decapsulation algorithms of the TRE-PC KEM produce different non-error messages, i.e. $K_1 \neq \perp$, $K_2 \neq \perp$, and $K_1 \neq K_2$, where

$$K_1 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}(C_1^*, V_{C^*}, sk_r),$$

$$K_2 = \mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}(C_1^*, TS_{t^*}, sk_r).$$

From the definition of the TRE-PC scheme, it is clear that $\Pr[E_1] \leq \Pr[E_2]$. Since the TRE-PC KEM is binding, $\Pr[E_2]$ is negligible. As result, $\Pr[E_1]$ is also negligible and the theorem follows. □

Theorem 36 *If the TRE-PC KEM is IND-TR-KEM-CCA$_{OS}$ secure and the DEM is IND-CCA2 secure, then the TRE-PC scheme is IND-TR-CCA$_{OS}$ secure.*

Proof. We prove the theorem using a sequence of games.

Game_0: this game is a legitimate IND-TR-CCA$_{\mathrm{OS}}$ attack game, defined as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, param)$. $\mathcal{A}_1$ has access to the following oracles:
   - An oracle for $\mathsf{Ext_{TS}}$, which takes as input a release time t, and returns $\mathsf{Ext_{TS}}(mk, t)$.
   - An oracle for $\mathsf{Dec_{RK}}$, which takes as input a ciphertext C and a pre-open key V_C, and returns $\mathsf{Dec_{RK}}(C, V_C, sk_r)$.
   - An oracle for $\mathsf{Dec_{PK}}$, which takes as input a ciphertext C and a release time t, and returns $\mathsf{Dec_{PK}}(C, TS_t, sk_r)$.

   $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^* and some state information $state$.
3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed in two steps:
   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) Compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.
4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ has access to the same types of oracle as $\mathcal{A}_1$. However, $\mathcal{A}_2$ may not query the $\mathsf{Dec_{RK}}$ oracle on the input (C^*, V_{C^*}) or the $\mathsf{Dec_{PK}}$ oracle on the input (C^*, t^*). $\mathcal{A}_2$ terminates by outputting a guess bit b'.

Let E_0 be the event that $b = b'$ at the end of the game.

Game_1: this game is identical to Game_0, except for the following.

1. In the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed as follows:
   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) Randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, and compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.
2. In Phase 2, on receiving a $\mathsf{Dec_{RK}}$ query on the input (C, V_C), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $V_C = V_{C^*}$, the challenger returns $\mathsf{DEM.Dec}(C_2, K^\dagger)$.
3. In Phase 2, on receiving a $\mathsf{Dec_{PK}}$ query on the input (C, t), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $t = t^*$, the challenger returns $\mathsf{DEM.Dec}(C_2, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of Game_1. We next prove the following claims.

Claim 7 $|\Pr[E_0] - \Pr[E_1]|$ *is negligible if the TRE-PC KEM is IND-TR-KEM-CCA*$_{OS}$ *secure.*

Proof. We construct an IND-TR-KEM-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the TRE-PC KEM, which makes use of an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage $\frac{1}{2}|\Pr[E_0] - \Pr[E_1]|$. As a result we can conclude that $|\Pr[E_0] - \Pr[E_1]|$ is negligible, since the TRE-PC KEM is IND-TR-KEM-CCA$_{\mathrm{OS}}$ secure.

$\mathcal{A}'$ takes $(pk_r, param)$, which is generated by the challenger, as input and consists of the following sub-algorithms:

1. The sub-algorithm $\mathcal{A}'_1$ executes $\mathcal{A}_1$ with input $(pk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries as follows.
   - If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}'_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t it receives from the extraction oracle to $\mathcal{A}_1$.

- If $\mathcal{A}_1$ queries the $\mathsf{Dec}_{\mathsf{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}'_1$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{RK}}$ oracle on (C_1, V_C). If it receives $\perp$, $\mathcal{A}'_1$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'_1$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.
- If $\mathcal{A}_1$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and for the time t, then $\mathcal{A}'_1$ queries its $\mathsf{TRE\text{-}PC\text{-}KEM.Decap}_{\mathsf{PK}}$ oracle on (C_1, t). If it receives $\perp$, $\mathcal{A}'_1$ returns $\perp$. Otherwise, if it receives K, $\mathcal{A}'_1$ returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.

Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}'_1$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(K_b, C_1^*, V_{C^*}, state')$ as input, where (K_b, C_1^*, V_{C^*}) is the challenge for the TRE-PC KEM. $\mathcal{A}'_2$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0, 1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$ except for the following two cases.
   (a) If $\mathcal{A}_2$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle on the input (C, V_C), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $V_C = V_{C^*}$, then $\mathcal{A}'_2$ returns $\mathsf{DEM.Dec}(C_2, K_b)$.
   (b) If $\mathcal{A}_2$ queries the $\mathsf{Dec}_{\mathsf{PK}}$ oracle on the input (C, t), where $C = (C_1^*, C_2)$, $C_2 \neq C_2^*$, and $t = t^*$, then $\mathcal{A}'_2$ returns $\mathsf{DEM.Dec}(C_2, K_b)$.

   If $\mathcal{A}_2$ eventually terminates by outputting a guess bit d', then $\mathcal{A}'_2$ terminates by outputting a guess bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CCA$_{\mathrm{OS}}$ attacker, and its advantage is $\frac{1}{2}|\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]|$. If $b = 0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CCA$_{\mathrm{OS}}$ attacker may query in Game_0; otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $\Pr[b' = 1|b = 0] = \Pr[E_0]$, $\Pr[b' = 1|b = 1] = \Pr[E_1]$, and the claim follows. □

Claim 8 $|\Pr[E_1] - \frac{1}{2}|$ *is negligible if the DEM is IND-CCA2 secure.*

Proof. We construct an IND-CCA2 attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the DEM, which makes use of an IND-TR-CCA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage $|\Pr[E_1] - \frac{1}{2}|$. As a result, we can conclude that $|\Pr[E_1] - \frac{1}{2}|$ is negligible, since the DEM is IND-CCA2 secure. $\mathcal{A}'$ takes ℓ as input, and consists of the following sub-algorithms.

1. The sub-algorithm $\mathcal{A}'_1$ takes ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen_U}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries as follows.

   - If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}'_1$ computes and returns the timestamp TS_t to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec_{RK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the pre-open key V_C, then $\mathcal{A}'_1$ computes $K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{RK}}(C_1, V_C)$. If $K = \perp$, $\mathcal{A}'_1$ returns $\perp$; otherwise, it returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.
   - If $\mathcal{A}_1$ queries the $\mathsf{Dec_{PK}}$ oracle with the ciphertext $C = (C_1, C_2)$ and the time t, then $\mathcal{A}'_1$ computes $K = \mathsf{TRE\text{-}PC\text{-}KEM.Decap_{PK}}(C_1, t)$. If $K = \perp$, $\mathcal{A}'_1$ returns $\perp$; otherwise, it returns $\mathsf{DEM.Dec}(C_2, K)$ to $\mathcal{A}_1$.

   Suppose $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$. $\mathcal{A}'_1$ terminates by outputting m_0 and m_1, and the state information $state' = (state, t^*)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(C^*_2, state')$ as input, where C^*_2 is the challenge for the DEM. $\mathcal{A}'_2$ computes $(K^*, C^*_1, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^* = (C^*_1, C^*_2)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$, except for the following two cases.

   (a) If $\mathcal{A}_2$ queries the $\mathsf{Dec_{PK}}$ oracle on the input (C, V_C), where $C = (C^*_1, C_2)$, $C_2 \neq C^*_2$, and $V_C = V_{C^*}$, $\mathcal{A}'_2$ queries its $\mathsf{DEM.Dec}$ oracle on the input C_2 and sends the result to $\mathcal{A}_2$.

   (b) If $\mathcal{A}_2$ queries the $\mathsf{Dec_{PK}}$ oracle on the input (C, t), where $C = (C^*_1, C_2)$, $C_2 \neq C^*_2$, and $t = t^*$, $\mathcal{A}'_2$ queries its $\mathsf{DEM.Dec}$ oracle on the input C_2 and sends the result to $\mathcal{A}_2$.

   If $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}'_2$ terminates by outputting b'.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-CCA2 attacker, and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage is equal to $\mathcal{A}$'s advantage in the game Game_1, i.e. $|\Pr[E_1] - \frac{1}{2}|$. Since the DEM is IND-CCA2 secure, $|\Pr[E_1] - \frac{1}{2}|$ is negligible, and the claim follows. □

From claims 4 and 5 we have that both $|\Pr[E_0] - \Pr[E_1]|$ and $|\Pr[E_1] - \frac{1}{2}|$ are negligible. As a result, $|\Pr[E_0] - \frac{1}{2}|$ is also negligible, and the theorem follows. □

Using exactly the same techniques as those used in proving the previous theorem, we can prove the following theorem regarding IND-TR-CCA$_{\mathrm{TS}}$ security.

Theorem 37 *If the TRE-PC KEM is IND-TR-KEM-CCA$_{TS}$ secure and the DEM is IND-CCA2 secure, then the TRE-PC scheme is IND-TR-CCA$_{TS}$ secure.*

We next prove corresponding results regarding the IND-TR-CPA$_{\mathrm{OS}}$ security of the TRE-PC KEM.

Theorem 38 *If the TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{OS}$ secure.*

Proof. We prove the theorem using a sequence of games.

Game_0: this game is a legitimate IND-TR-CPA$_{\mathrm{OS}}$ attack game, defined as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker $\mathcal{A}_1$ executes with input $(pk_r, param)$. $\mathcal{A}_1$ has access to an oracle for $\mathsf{Ext_{TS}}$, which takes as input a release time t, and returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^*, and some state information $state$.

3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed in two steps:
   (a) Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) Compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.
4. Phase 2: The attacker $\mathcal{A}_2$ executes with input $(C^*, V_{C^*}, state)$. $\mathcal{A}_2$ also has access to the $\mathsf{Ext_{TS}}$ oracle. $\mathcal{A}_2$ eventually terminates by outputting a guess bit b'.

Let E_0 be the event that $b = b'$ at the end of the game.

$\mathsf{Game_1}$: this game is identical to $\mathsf{Game_0}$, except that, in the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ and a pre-open key V_{C^*}, which are computed as follows:

1. Compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
2. Randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, and compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of $\mathsf{Game_1}$. We next prove the following claims.

Claim 9 $|\Pr[E_0] - \Pr[E_1]|$ *is negligible if the TRE-PC KEM is IND-TR-KEM-CPA$_{OS}$ secure.*

Proof. We construct an IND-TR-KEM-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the TRE-PC KEM, which makes use of an IND-TR-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage $\frac{1}{2}|\Pr[E_0] - \Pr[E_1]|$. We can thus conclude that $|\Pr[E_0] - \Pr[E_1]|$ is negligible, since the TRE-PC KEM is IND-TR-KEM-CPA$_{\mathrm{OS}}$ secure.

$\mathcal{A}'$ takes $(pk_r, param)$ as input and consists of the following sub-algorithms:

1. The sub-algorithm $\mathcal{A}'_1$ executes $\mathcal{A}_1$ with input $(pk_r, param)$ and answers $\mathcal{A}_1$'s extraction oracle query. On receiving an extraction query for a time t, $\mathcal{A}'_1$

makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}_1$. If $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$, then $\mathcal{A}'_1$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(K_b, C_1^*, V_{C^*}, state')$ as input, where (K_b, C_1^*, V_{C^*}) is the challenge for the TRE-PC KEM. $\mathcal{A}'_2$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0, 1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$. If $\mathcal{A}_2$ eventually terminates by outputting a guessing bit d', then $\mathcal{A}'_2$ terminates by outputting a guessing bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CPA$_{\mathrm{OS}}$ attacker and its advantage is $\frac{1}{2}|\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]|$. If $b = 0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CPA$_{\mathrm{OS}}$ attacker may query in Game_0; otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $\Pr[b' = 1|b = 0] = \Pr[E_0]$, $\Pr[b' = 1|b = 1] = \Pr[E_1]$, and the claim follows. □

Claim 10 $|\Pr[E_1] - \frac{1}{2}|$ *is negligible if the DEM is IND-CPA secure.*

Proof. We construct an IND-CPA attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the DEM, which makes use of an IND-TR-CPA$_{\mathrm{OS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage $|\Pr[E_1] - \frac{1}{2}|$. We can thus conclude that $|\Pr[E_1] - \frac{1}{2}|$ is negligible, since the DEM is IND-CPA secure.

$\mathcal{A}'$ takes ℓ as input and consists of the following sub-algorithms:

1. The sub-algorithm $\mathcal{A}'_1$ takes ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen_U}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, param)$. If $\mathcal{A}_1$ makes an extraction oracle query for a time t, then $\mathcal{A}'_1$ computes and returns the timestamp TS_t to $\mathcal{A}_1$. If $\mathcal{A}_1$

terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$, then $\mathcal{A}'_1$ terminates by outputting m_0 and m_1, and the state information $state' = (state, t^*)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(C^*_2, state')$ as input, where C^*_2 is the challenge for the DEM. $\mathcal{A}'_2$ computes $(K^*, C^*_1, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^* = (C^*_1, C^*_2)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, V_{C^*}, state)$, and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$. If $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}'_2$ terminates by outputting b'.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-CPA attacker for the DEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage is equal to $\mathcal{A}$'s advantage in Game_1, i.e. $|\Pr[E_1] - \frac{1}{2}|$. Since the DEM is IND-CPA secure, $|\Pr[E_1] - \frac{1}{2}|$ is negligible, and the claim follows. □

From claims 6 and 7 we have that both $|\Pr[E_0] - \Pr[E_1]|$ and $|\Pr[E_1] - \frac{1}{2}|$ are negligible. As a result, $|\Pr[E_0] - \frac{1}{2}|$ is also negligible and the theorem follows. □

Using exactly the same techniques as those used in proving the previous theorem, we can prove the the following theorem regarding IND-TR-CPA$_{\mathrm{TS}}$ security.

Theorem 39 *If the TRE-PC KEM is IND-TR-KEM-CPA$_{TS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{TS}$ secure.*

We next prove corresponding results regarding the IND-TR-CPA$_{\mathrm{IS}}$ security of the TRE-PC-KEM.

Theorem 40 *If the TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure and the DEM is IND-CPA secure, then the TRE-PC scheme is IND-TR-CPA$_{IS}$ secure.*

Proof. We prove the theorem using a sequence of games.

Game_0: this game is a legitimate IND-TR-CPA$_{\mathrm{IS}}$ attack game, defined as follows.

1. Game Setup: The challenger runs Setup to generate the time server's master key mk and the public parameters $param$. The challenger also runs $\mathsf{Gen_U}$ to generate a public/private key pair (pk_r, sk_r).
2. Phase 1: The attacker executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$. $\mathcal{A}_1$ has access to the $\mathsf{Ext_{TS}}$ oracle, which takes as input a release time t, and returns $\mathsf{Ext_{TS}}(mk, t)$. $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, a release time t^*, which is larger than the the inputs to the $\mathsf{Ext_{TS}}$ oracle, and some state information $state$.
3. Challenge: The challenger returns $C^* = (C_1^*, C_2^*)$ which is computed as follows:
   (a) compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
   (b) compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^*)$.
4. Phase 2: The attacker executes $\mathcal{A}_2$ on the input $(C^*, state)$. $\mathcal{A}_2$ has access to the $\mathsf{Ext_{TS}}$ oracle on any input $t < t^*$. $\mathcal{A}_2$ eventually terminates by outputting a guessing bit b'.

Let E_0 be the event that $b = b'$ at the end of the game.

Game_1: this game is identical to Game_0, except that, in the Challenge phase, the challenger returns $C^* = (C_1^*, C_2^*)$ which is computed as follows:

1. compute $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$,
2. randomly select $K^\dagger \in \{0,1\}^{\mathsf{KeyLen}(\ell)}$, compute $C_2^* = \mathsf{DEM.Enc}(m_b, K^\dagger)$.

Let E_1 be the event that $b = b'$ at the end of Game_1. We next prove the following claims.

Claim 11 $|\Pr[E_0] - \Pr[E_1]|$ *is negligible if the TRE-PC KEM is IND-TR-KEM-CPA$_{IS}$ secure.*

Proof. We construct an IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the TRE-PC KEM, which makes use of an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage $\frac{1}{2}|\Pr[E_0] - \Pr[E_1]|$. We thus can conclude that $|\Pr[E_0] - \Pr[E_1]|$ is negligible as the TRE-PC KEM is IND-TR-KEM-CPA$_{\mathrm{IS}}$ secure.

$\mathcal{A}'$ takes $(pk_r, sk_r, param)$ as input and consists of the following sub-algorithms:

1. The sub-algorithm $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$ and answers $\mathcal{A}_1$'s extraction oracle query. On receiving an extraction query for a time t, $\mathcal{A}'_1$ makes a similar query to its own extraction oracle and returns the timestamp TS_t to $\mathcal{A}_1$.

   If $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$, then $\mathcal{A}'_1$ terminates by outputting t^*, and the state information $state' = (state, m_0, m_1)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(K_b, C_1^*, state')$ as input, where (K_b, C_1^*) is the challenge for the TRE-PC KEM. $\mathcal{A}'_2$ computes $C_2^* = \mathsf{DEM.Enc}(m_d, K_b)$ where d is randomly chosen from $\{0, 1\}$, and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as $\mathcal{A}'_1$. If $\mathcal{A}_2$ eventually terminates by outputting a guessing bit d', then $\mathcal{A}'_2$ terminates by outputting a guessing bit b', where $b' = 1$ if $d' = d$ and $b' = 0$ if $d' \neq d$.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-TR-KEM-CPA$_{\mathrm{IS}}$ attacker and its advantage is $\frac{1}{2}|\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]|$. If $b = 0$, $\mathcal{A}'$ provides perfect simulation for the oracles that an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ may query in Game_0; otherwise $\mathcal{A}'$ provides perfect simulation for the oracles that such an attacker may query in Game_1. Therefore, $\Pr[b' = 1|b = 0] = \Pr[E_0]$, $\Pr[b' = 1|b = 1] = \Pr[E_1]$, and the claim follows. □

Claim 12 $|\Pr[E_1] - \frac{1}{2}|$ *is negligible if the DEM is IND-CPA secure.*

Proof. We construct an IND-CPA attacker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for the DEM, which makes use of an IND-TR-CPA$_{\mathrm{IS}}$ attacker $\mathcal{A}$ as a subroutine, and has advantage

$|\Pr[E_1] - \frac{1}{2}|$. We thus can conclude that $|\Pr[E_1] - \frac{1}{2}|$ is negligible, since the DEM is IND-CPA secure.

$\mathcal{A}'$ takes ℓ as input and consists of the following sub-algorithms:

1. The sub-algorithm $\mathcal{A}'_1$ takes ℓ as input, runs Setup to generate mk and $param$, and runs $\mathsf{Gen_U}$ to generate the public/private key pair (pk_r, sk_r). $\mathcal{A}'_1$ executes $\mathcal{A}_1$ on the input $(pk_r, sk_r, param)$ and answers $\mathcal{A}_1$'s oracle queries in the same way as the challenger will in $\mathsf{Game_1}$. If $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, a release time t^*, and some state information $state$, then $\mathcal{A}'_1$ terminates by outputting m_0 and m_1, and the state information $state' = (state, t^*)$.

2. The sub-algorithm $\mathcal{A}'_2$ takes $(C_2^*, state')$ as input, where C_2^* is the challenge for the DEM. $\mathcal{A}'_2$ computes $(K^*, C_1^*, V_{C^*}) = \mathsf{TRE\text{-}PC\text{-}KEM.Encap}(t^*, pk_r)$ and sets $C^* = (C_1^*, C_2^*)$. $\mathcal{A}'_2$ executes $\mathcal{A}_2$ on the input $(C^*, state)$ and answers $\mathcal{A}_2$'s oracle queries in the same way as the challenger will in $\mathsf{Game_1}$. Suppose $\mathcal{A}_2$ eventually terminates by outputting a bit b', then $\mathcal{A}'_2$ terminates by outputting b'.

It is straightforward to verify that $\mathcal{A}'$ is a legitimate IND-CPA attacker for the DEM and provides perfect simulation for the oracles that $\mathcal{A}$ may query. $\mathcal{A}'$'s advantage is equal to $\mathcal{A}$'s advantage in the game $\mathsf{Game_1}$, i.e. $|\Pr[E_1] - \frac{1}{2}|$. Since the DEM is IND-CPA secure, $|\Pr[E_1] - \frac{1}{2}|$ is negligible, and the claim follows. □

From claims 8 and 9 we have that both $|\Pr[E_0] - \Pr[E_1]|$ and $|\Pr[E_1] - \frac{1}{2}|$ are negligible. As a result, $|\Pr[E_0] - \frac{1}{2}|$ is also negligible, and the theorem follows. □

9.4 Conclusions

In this chapter we have proposed a new primitive known as a TRE-PC KEM, and a hybrid paradigm using a TRE-PC KEM and a DEM for to construct TRE-PC

schemes. We have also proposed an efficient instantiation of a TRE-PC KEM, which shares some similarities with the Hwang-Yum-Lee schemes [124].

Chapter 10

Conclusions

Contents

In this chapter we provide a brief summary of what we have achieved in this thesis. We also outline some directions for further research.

10.1 Summary of Contributions

In the first part of the thesis we studied a number of aspects of key establishment protocols, including security properties, security models, existing key establishment protocols, and proposals for a new security model and protocols. This study has revealed security vulnerabilities in a number of existing key establishment protocols, some of which have been proved secure in well-known security models. This shows that the existing security models do not cover all the potentially desirable security properties. As a result, we have proposed a new security model that covers all the security properties (we have described). We also proposed two compilers which are more efficient than existing schemes, and a number of novel protocols.

In the second part of this thesis, we refined the Hwang-Yum-Lee model for TRE-PC schemes. The refined security model covers a new security property, namely the binding property, which we believe is a desirable property for TRE-PC schemes. In addition, the refined model provides more appropriate definitions for semantic security against four different type of attackers, and establishes the relationships

between security notions. We have also proposed a hybrid paradigm for constructing TRE-PC schemes based on a TRE-PC KEM and a DEM, and we have moreover proposed an instantiation of a TRE-PC KEM. Although sharing some similarities with the scheme of Hwang, Yum and Lee, our instantiation is more efficient than this latter scheme.

10.2 Future Research Directions

Although much work has been done on key establishment, there are still many interesting open research questions. One is to understand the security properties that may be required of key establishment protocols, especially in a group environment when a dishonest partner exists. Another possible direction is to find new ways to construct key establishment protocols. In order to achieve forward secrecy, most existing protocols employ a Diffie-Hellman style key agreement process; it would be interesting to investigate other structures that can be used to achieve this desirable property.

Since TRE-PC schemes can achieve a number of desirable properties such as binding, it would be interesting to investigate the application of such schemes in constructing large security systems.

Bibliography

[1] GB 15629.11-2003. Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium access control (MAC) and Physical Layer(PHY) Specifications, 2003.

[2] GB 15629.11-2003-XG1-2006. Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11:Wireless LAN Medium Access Control(MAC) and Physical Layer(PHY) specifications Amendment 1, 2006.

[3] GB 15629.1102-2003. Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer(PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, 2003.

[4] M. Abadi, B. Blanchet, and C. Fournet. Just fast keying in the pi calculus. In D. A. Schmidt, editor, *Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004*, volume 2986 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2004.

[5] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *ACM Conference on Computer and Communications Security*, pages 36–47, 1997.

[6] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[7] M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval. Password-based group key exchange in a constant number of rounds. In M. Yung, editor, *Proceedings of the 9th International Workshop on Practice and Theory in Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 427–442. Springer, 2006.

[8] M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based encrypted key exchange. In V. Serge, editor, *Proceedings of the 8th International Workshop on Theory and Practice in Public Key*, volume 3386 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2005.

[9] M. Abdalla, P.-A. Fouque, and D. Pointcheval. Password-based authenticated key exchange in the three-party setting. In V. Serge, editor, *Proceedings of the 8th International Workshop on Theory and Practice in Public Key*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2005.

[10] M. Abdalla and D. Pointcheval. Simple password-based encrypted key exchange protocols. In A. Menezes, editor, *Topics in Cryptology — CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2005.

[11] G. Agnew, R. Mullin, and S. Vanstone. An interactive data exchange protocol based on discrete exponentiation. In *Advances in Cryptology — EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 159–166. Springer, 1988.

[12] L. Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.

[13] W. Aiello, S. M. Bellovin, M. B., R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Just fast keying: Key agreement in a hostile internet. *ACM Trans. Inf. Syst. Secur.*, 7(2):242–273, 2004.

[14] WAPI Alliance. WAPI Implementation Plan. http://www.wapia.org, 2003.

[15] R. Anderson and T. Lomas. Fortifying key negotiation schemes with poorly chosen passwords. *Electronics Letters*, 30(13):1040–1041, 1994.

[16] G. Ateniese, M. Steiner, and G. Tsudik. Authenticated group key agreement and friends. In *Proceedings of the 5th ACM conference on Computer and communications security*, pages 17–26. ACM Press, 1998.

[17] M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. In *2005 IEEE Symposium on Security and Privacy*, pages 171–182. IEEE Computer Society, 2005.

[18] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. On password-based authenticated key exchange using collisionful hash functions. In J. Pieprzyk and J. Seberry, editors, *Information Security and Privacy, First Australasian Conference, ACISP'96*, volume 1172 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 1996.

[19] G. Barthe, J. Cederquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In *Automated Reasoning — Second International Joint Conference, IJCAR 2004*, volume 3097 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2004.

[20] R. Bauer, T. Berson, and R. Feiertag. A key distribution protocol using event markers. *ACM Trans. Comput. Syst.*, 1(3):249–255, 1983.

[21] J. A. Beachy and W. D. Blair. *Abstract Algebra.* Waveland Press, Inc., 3 edition, 2005.

[22] M. Bellare, A. Boldyreva, and A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology — EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2004.

[23] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in Cryptology — CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[24] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428, 1998.

[25] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[26] M. Bellare and S. Goldwasser. Encapsulated key-escrow. Technical Report MIT/LCS/TR-688, MIT LCS, 1996.

[27] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.

[28] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, 2000.

[29] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology — CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 1993.

[30] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM Press, 1993.

[31] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.

[32] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive: Report 2004/331, 2004.

[33] M. Beller, L. Chang, and Y. Yacobi. Security for personal communications services: public-key vs. private key approaches. In *Proceedings of the 3rd International Symposium on personal, indoor and mobile radio communications*, pages 26–31. IEEE Press, 1992.

[34] M. Beller, L. Chang, and Y. Yacobi. Privacy and authentication on a portable communications system. *IEEE Journal on Selected Areas in Communications*, 11(6):821–829, 1993.

[35] M. Beller and Y. Yacobi. Batch Diffie-Hellman key agreement systems and their application to portable communications. In R. Rueppel, editor, *Advances in Cryptology — EUROCRYPT 1992*, volume 658 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 1992.

[36] M. Beller and Y. Yacobi. Fully-fledged two-way public key authentication and key agreementfor low-cost terminals. *Electronics Letters*, 29(11):999–1001, 1993.

[37] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society, 1992.

[38] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[39] K. Bentahar, P. Farshim, J. Malone-Lee, and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive: Report 2005/058, 2005.

[40] R. Bird, I. S. Gopal, A. Herzberg, P. A. Janson, S. Kutten, R. Molva, and M. Yung. Systematic design of two-party authentication protocols. In *Advances in Cryptology — CRYPTO 1991*, pages 44–61. Springer, 1992.

[41] T. E. Bjørstad and A. W. Dent. Building better signcryption schemes with tag-kems. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 491–507. Springer, 2006.

[42] J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.

[43] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Proceedings of Cryptography and Coding, 6th IMA International Conference*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 1997.

[44] S. Blake-Wilson and A. Menezes. Entity authentication and authenticated key transport protocols employing asymmetric techniques. In B. Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Proceedings of Security Protocols, 5th International Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 137–158. Springer, 1997.

[45] S. Blake-Wilson and A. Menezes. Authenticated diffie-hellman key agreement protocols. In S. E. Tavares and H. Meijer, editors, *Proceedings of the Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 339–361. Springer, 1999.

[46] S. Blake-Wilson and A. Menezes. Unknown key-share attacks on the Station-to-Station (STS) protocol. In H. Imai and Y. Zheng, editors, *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 154–170. Springer, 1999.

[47] B. Blanchet. A computationally sound mechanized prover for security protocols. In *2006 IEEE Symposium on Security and Privacy*, pages 140–154. IEEE Computer Society, 2006.

[48] B. Blanchet and D. Pointcheval. Provably secure threshold password-authenticated key exchange. In C. Dwork, editor, *Advances in Cryptology — CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 2006.

[49] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[50] D. Boneh and M. Naor. Timed commitments. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, pages 236–254. Springer, 2000.

[51] C. Boyd. Towards a classification of key agreement protocols. In *proceedings of the Eighth IEEE Computer Security Foundations Workshop*, pages 38–43. IEEE Computer Society, 1995.

[52] C. Boyd. A class of flexible and efficient key management protocols. In *proceedings of the Ninth IEEE Computer Security Foundations Workshop*, pages 2–8. IEEE Computer Society, 1996.

[53] C. Boyd. On key agreement and conference key agreement. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *Information Security and Privacy, Second Australasian Conference, ACISP'97*, volume 1270 of *Lecture Notes in Computer Science*, pages 294–302. Springer, 1997.

[54] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2004.

[55] C. Boyd and D. Park. Public key protocols for wireless communications. In *Proceedings of the 1st International Conference on Information Security and Cryptology*, pages 47–57. Korea Institute of Information Security and Cryptology (KIISC), 1998.

[56] V. Boyko, P. D. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange using diffie-hellman. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer, 2000.

[57] E. Bresson and D. Catalano. Constant round authenticated group key agreement via distributed computation. In F. Bao, R. H. Deng, and J. Zhou, editors, *Proceedings of the 7th International Workshop on Practice and Theory in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2004.

[58] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2002.

[59] E. Bresson, O. Chevassut, and D. Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Y. Zheng, editor, *Advances in*

Cryptology — ASIACRYPT 2002, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514. Springer, 2002.

[60] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 255–264. ACM Press, 2001.

[61] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. D. Santis, editor, *Advances in Cryptology— EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, 1994.

[62] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In A. D. Santis, editor, *Pre–Proceedings of EUROCRYPT 1994*, pages 279–290, 1994.

[63] M. Burmester and Y. Desmedt. A secure and scalable group key exchange system. *Inf. Process. Lett.*, 94(3):137–143, 2005.

[64] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Proceedings of the Twelfth ACM Symposium on Operating System Principles*, pages 1–13, 1989.

[65] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[66] J. Byun and D. Lee. N-Party encrypted Diffie-Hellman key exchange using different passwords. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 2005.

[67] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. D. MacKenzie. Universally composable password-based key exchange. In R. Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, 2005.

[68] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology*

— *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, 2001.

[69] U. Carlsen. Optimal privacy and authentication on a portable communications system. *Operating Systems Review*, 28(3):16–23, 1994.

[70] J. Cathalo, B. Libert, and J.-J. Quisquater. Efficient and non-interactive timed-release encryption. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Proceedings of the 7th International Conference on Information and Communications Security*, volume 3783 of *Lecture Notes in Computer Science*, pages 291–303. Springer, 2005.

[71] A. C. F. Chan and I. F. Blake. Scalable, server-passive, user-anonymous timed release cryptography. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 504–513. IEEE Computer Society, 2005.

[72] D. Chaum and T. P. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO 1993*, pages 89–105. Springer, 1993.

[73] L. Chen, Z. Cheng, and N. Smart. Identity-based key agreement protocols from pairings. Cryptology ePrint Archive: Report 2006/199, 2006.

[74] L. Chen, D. Gollmann, and C. J. Mitchell. Key distribution without individual trusted authentification servers. In *Proceedings of the Eighth IEEE Computer Security Foundations Workshop*, pages 30–36. IEEE Computer Society, 1995.

[75] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *Proc. of the 16th IEEE Computer Security Foundations Workshop*, pages 219–233. IEEE Computer Society Press, 2003.

[76] L. Chen and Q. Tang. Bilateral unknown key-share attacks in key agreement protocols. Cryptology ePrint Archive: Report 2007/209, 2007.

[77] J. Cheon and B. Jun. A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem. In D. Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 212–225. Springer, 2003.

[78] O. Chevassut, P. Fouque, P. Gaudry, and D. Pointcheval. The twist-augmented technique for key exchange. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 410–426. Springer, 2006.

[79] K. Choi, J. Hwang, D. Lee, and I. Seo. Id-based authenticated key agreement for low-power mobile devices. In C. Boyd and J. M. González Nieto, editors, *Information Security and Privacy, 10th Australasian Conference, Proceedings*, volume 3574 of *Lecture Notes in Computer Science*, pages 494–505. Springer, 2005.

[80] K. Y. Choi, J. Y. Hwang, and D. H. Lee. Efficient ID-based group key agreement with bilinear maps. In F. Bao, R. Deng, and J. Y. Zhou, editors, *Proceedings of the 2004 International Workshop on Practice and Theory in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 130–144. Springer, 2004.

[81] K. R. Choo, C. Boyd, and Y. Hitchcock. Errors in computational complexity proofs for protocols. In B. Roy, editor, *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 624–643. Springer, 2005.

[82] K. R. Choo, C. Boyd, and Y. Hitchcock. Examining indistinguishability-based proof models for key establishment protocols. In B. Roy, editor, *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 585–604. Springer, 2005.

[83] K. R. Choo and Y. Hitchcock. Security requirements for key establishment proof models: Revisiting bellare-rogaway and Jeong-Katz-Lee protocols. In C. Boyd and J. Nieto, editors, *Information Security and Privacy, 10th Australasian Conference, Proceedings*, volume 3574 of *Lecture Notes in Computer Science*, pages 429–442. Springer, 2005.

[84] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[85] G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 1999.

[86] J. Daemen and V. Rijmen. *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer, 2002.

[87] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

[88] D. Denning and G. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.

[89] A. W. Dent. Hybrid signcryption schemes with outsider security. In J. Zhou, J. Lopez, R. H. Deng, and F. Bao, editors, *Proceedings of the 8th International Information Security Conference*, volume 3650 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2005.

[90] A. W. Dent and Q. Tang. Revisiting the security model for timed-release public-key encryption with pre-open capability. Cryptology ePrint Archive: Report 2006/306, 2006.

[91] A. W. Dent and Q. Tang. Revisiting the security model for timed-release encryption with pre-open capability. In J. A. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, editors, *Information Security, 10th International Conference, ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 158–174. Springer, 2007.

[92] A. Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2000.

[93] D.Harkins and D. Carrel. The Internet Key Exchange (IKEv2) Protocol. IETF RFC 2409, 1998.

[94] T. Dierks and C. Allen. The TLS protocol version 1.0. IETF RFC 2246, 1999.

[95] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[96] W. Diffie, P. Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[97] X. J. Du, Y. Wang, J. H. Ge, and Y. M. Wang. ID-based authenticated two round multiparty key agreement. Cryptology ePrint Archive: Report 2003/247, 2003.

[98] X. J. Du, Y. Wang, J. H. Ge, and Y. M. Wang. An improved ID-based authenticated group key agreement scheme. Cryptology ePrint Archive, Report 2003/260, 2003.

[99] R. Dutta and R. Barua. Password-based Encrypted Group Key Agreement. *International Journal of Network Security*, 3(1):30–41, 2006.

[100] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[101] S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT 1991*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1993.

[102] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, 2003.

[103] M. Girault. Self-certified public keys. In D. Davies, editor, *Advances in Cryptology — EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer, 1991.

[104] O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[105] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 408–432. Springer, 2001.

[106] S. Goldwasser and Y. T. Kalai. On the (in)security of the Fiat-Shamir paradigm. In *Proceedings of the 44th Symposium on Foundations of Computer Science*, pages 102–115. IEEE Computer Society, 2003.

[107] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365–377. ACM, 1982.

[108] L. Gong. Using one-way functions for authentication. *SIGCOMM Comput. Commun. Rev.*, 19(5):8–11, 1989.

[109] L. Gong. Increasing availability and security of an authentication service. *IEEE Journal on Selected Areas in Communications*, 11(5):657–662, 1993.

[110] L. Gong. Lower bounds on messages and rounds for network authentication protocols. In *ACM Conference on Computer and Communications Security*, pages 26–37, 1993.

[111] L. Gong, T. Lomas, R. Needham, and J. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[112] C. Günther. An identity-based key-exchange protocol. In J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — EUROCRYPT 1989*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer, 1990.

[113] Y. Lee H. Lee, H. Lee. An authenticated group key agreement protocol on braid groups. Cryptology ePrint Archive: Report 2003/018, 2005.

[114] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In *ACM Conference on Computer and Communications Security*, pages 122–131, 1998.

[115] S. Halevi and H. Krawczyk. A plausible approach to computer-aided cryptographic proofs. Cryptology ePrint Archive, Report 2005/181, 2005.

[116] L. Harn and H. Y. Lin. Authenticated key agreement without using oneway hash functions. *Electronics Letters*, 37(10):1429–1431, 2001.

[117] J. Herranz and J. L. Villar. An Unbalanced Protocol for Group Key Exchange. In *TrustBus 2004*, volume 3184 of *Lecture Notes in Computer Science*, pages 172–180. Springer, 2004.

[118] S. Hirose and S. Yoshida. An authenticated diffie-hellman key agreement protocol secure against active attacks. In H. Imai and Y. Zheng, editors, *Proceedings of the first International Workshop on Practice and Theory in Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 1998.

[119] Y. Hitchcock, C. Boyd, and J. Nieto. Tripartite key exchange in the canetti-krawczyk proof model. In A. Canteaut and K. Viswanathan, editors, *Advances in Cryptology — INDOCRYPT 2003*, volume 3348 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2004.

[120] Y. Hitchcock, C. Boyd, and J. Manuel González Nieto. Modular proofs for key exchange: rigorous optimizations in the canetti-krawczyk model. *Appl. Algebra Eng. Commun. Comput.*, 16(6):405–438, 2006.

[121] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

[122] G. Horn, K. M. Martin, and C. J. Mitchell. Authentication protocols for mobile network environment value-added services. *IEEE Transactions on Vehicular Technology*, 51(2):383–392, 2002.

[123] G. Horn and B. Preneel. Authentication and payment in future mobile systems. *Journal of Computer Security*, 8(2/3), 2000.

[124] M. S. Hwang, J. W. Lo, and S. C. Lin. An efficient user identification scheme based on ID-based cryptosystem. *Computer Standards & Interfaces*, 26:565–569, 2004.

[125] Y. Hwang, D. Yum, and P. Lee. Timed-release encryption with pre-open capability and its application to certified e-mail system. In J. Zhou, J. Lopez, R. Deng, and F. Bao, editors, *Proceedings of the 8th International Information Security Conference*, volume 3650 of *Lecture Notes in Computer Science*, pages 344–358. Springer, 2005.

[126] I. Ingemarsson, D. Tang, and C. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5):714–720, 1982.

[127] Institute of Electrical and Electronics Engineers, Inc. *IEEE P1363.2 draft D20, Standard Specifications for Password-Based Public-Key Cryptographic Techniques*, March 2005.

[128] International Organization for Standardization. *ISO/IEC 9797–2, Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*, 1999.

[129] International Organization for Standardization. *ISO/IEC 11770–4, Information technology — Security techniques — Key management — Part 4: Mechanisms based on weak secrets*, 2006.

[130] D. P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review*, 26(5):5–26, 1996.

[131] D. P. Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings of the 1997 Workshop on Enterprise Security*, pages 248–255, 1997.

[132] M. Jakobsson and D. Pointcheval. Mutual authentication for low-power mobile devices. In P. F. Syverson, editor, *Financial Cryptography, 5th International Conference*, volume 2339 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 2001.

[133] P. Janson and G. Tsudik. Secure and minimal protocols for authenticated key distribution. *Computer Communications*, 18(9):645–653, 1995.

[134] É. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption, 9th International Workshop*, volume 2365 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2002.

[135] I. Jeong, J. Katz, and D. Lee. One-round protocols for two-party authenticated key exchange. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security, Second International Conference*, volume 3089 of *Lecture Notes in Computer Science*, pages 220–232. Springer, 2004.

[136] J. E. Johnson, D. E. Langworthy, L. Lamport, and F. H. Vogt. Formal specification of a web services protocol. *Electr. Notes Theor. Comput. Sci.*, 105:147–158, 2004.

[137] A. M. Johnston and P. Gemmell. Authenticated key exchange provably secure against the man-in-the-middle attack. *Journal of Cryptology*, 15(2):139–148, 2002.

[138] A. Joux. A one round protocol for tripartite diffie-hellman. *Journal of Cryptology*, 17(4):263–276, 2004.

[139] M. J. Jacobson Jr., R. Scheidler, and H. C. Williams. An improved real-quadratic-field-based key exchange procedure. *Journal of Cryptology*, 19(2):211–239, 2006.

[140] A. Juels and J. Brainard. Client puzzles: A cryptographic defense against connection depletion. *Networks and Distributed Security Systems (NDSS 1999)*, pages 1168–1177, 1999.

[141] M. Just and S. Vaudenay. Authenticated multi-party key agreement. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 36–49. Springer, 1996.

[142] J. Katz, P. D. MacKenzie, G. Taban, and V. D. Gligor. Two-server password-only authenticated key exchange. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, Proceedings*, volume 3531 of *Lecture Notes in Computer Science*, pages 1–16, 2005.

[143] J. Katz, R. Ostrovsky, and M. Yung. Efficient password-authenticated key exchange using human-memorable passwords. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, 2001.

[144] J. Katz and J. Shin. Modeling insider attacks on group key-exchange protocols. Cryptology ePrint Archive: Report 2005/163, 2005.

[145] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, 2003.

[146] S. Kent. IP Authentication Header. IETF RFC 4302, 2005.

[147] S. Kent. IP Encapsulating Security Payload (ESP). IETF RFC 4303, 2005.

[148] J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, 2001.

[149] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, and C. Park. New public-key cryptosystem using braid groups. In M. Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2000.

[150] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–207, 1987.

[151] K. Koyama. Secure conference key distribution schemes for conspiracy attack. In R. Rueppel, editor, *Advances in Cryptology — EUROCRYPT 1992*, volume 330 of *Lecture Notes in Computer Science*, pages 449–453. Springer, 1992.

[152] K. Koyama and K. Ohta. Identity-based conference key distribution systems. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO 1987*, volume 283 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 1988.

[153] K. Koyama and K. Ohta. Security of improved identity-based conference key distribution systems. In C. Günther, editor, *Advances in Cryptology — EUROCRYPT 1988*, volume 330 of *Lecture Notes in Computer Science*, pages 11–19. Springer, 1988.

[154] H. Krawczyk. Skeme: a versatile secure key exchange mechanism for internet. In *Proceedings of the 1996 Symposium on Network and Distributed System Security*, pages 114–127. IEEE Computer Society, 1996.

[155] H. Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In Victor Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[156] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. IETF RFC 2104, 1997.

[157] C. Kudla and K. G. Paterson. Modular security proofs for key agreement protocols. In B. K. Roy, editor, *Advances in Cryptology — ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 549–565. Springer, 2005.

[158] T. Kwon. Practical authenticated key agreement using passwords. In K. Zhang and Y. Zheng, editors, *Proceedings of the 7th Information Security Conference*, volume 3225 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[159] T. Kwon and J. Song. Efficient and secure password-based authentication protocols against guessing attacks. *Computer Communications*, 21(9):853–861, 1998.

[160] T. Kwon and J. Song. Efficient key exchange and authentication protocols protecting weak secrets. *IEICE Transactions Fundamentals*, E81-A(1):156–163, 1998.

[161] T. Kwon and J. Song. Secure agreement scheme for g^{xy} via password authentication. *Electronics Letters*, 35(11):892–893, 1999.

[162] C. S. Laih, L. Ding, and Y. M. Huang. Password-only authenticated key establishment protocol without public key cryptography. *Electronics Letters*, 41(4):185–186, 2005.

[163] P. Laud. Handling encryption in an analysis for secure information flow. In P. Degano, editor, *Proceedings of the 12th European Symposium on Programming*, volume 2618 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2003.

[164] P. Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *2004 IEEE Symposium on Security and Privacy*, pages 71–85. IEEE Computer Society, 2004.

[165] K. Lauter and A. Mityagin. Security analysis of KEA authenticated key exchange protocol. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Proceedings of the 9th International Conference on Theory and Practice of*

Public-Key Cryptography, volume 3958 of *Lecture Notes in Computer Science*, pages 378–394. Springer, 2006.

[166] H. Lee, K. Ha, and K. Ku. ID-based multi-party authenticated key agreement protocols from multilinear forms. In J. Zhou, J. Lopez, R. H. Deng, and F. Bao, editors, *Information Security, 8th International Conference Proceedings*, volume 3650 of *Lecture Notes in Computer Science*, pages 104–117. Springer, 2005.

[167] H. Lee, D. Won, K. Sohn, and H. Yang. Efficient 3-pass password-based key exchange protocol with low computational cost for client. In J. Song, editor, *Information Security and Cryptology — ICISC 1999*, volume 1787 of *Lecture Notes in Computer Science*, pages 147–155. Springer, 1999.

[168] W. B. Lee and K. C. Liao. Constructing identity-based cryptosystems for discrete logarithm based cryptosystems. *Journal of Network and Computer Applications*, 27:191–199, 2004.

[169] X. Li, S. Moon, and J. Ma. On the security of the authentication module of chinese WLAN standard implementation plan. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 340–348, 2006.

[170] C. Lim and P. Lee. Several practical protocols for authentication and key exchange. *Inf. Process. Lett.*, 53(2):91–96, 1995.

[171] C. Lin, H. Sun, and T. Hwang. Three-party encrypted key exchange: Attacks and a solution. *Operating Systems Review*, 34(4):12–20, 2000.

[172] T. Lomas, L. Gong, J. Saltzer, and R. Needham. Reducing risks from poorly chosen keys. *ACM SIGOPS Operating Systems Review*, 23(5):14–18, 1989.

[173] T. Lomas, L. Gong, J. Saltzer, and R. Needham. Reducing risks from poorly chosen keys. In *Proceedings of the Twelfth ACM Symposium on Operating System Principles*, pages 14–18, 1989.

[174] G. Lowe. An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.*, 56(3):131–133, 1995.

[175] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 1996.

[176] G. Lowe. Some new attacks upon security protocols. In *Ninth IEEE Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society, 1996.

[177] S. Lu and S. A. Smolka. Model checking the secure electronic transaction (SET) protocol. In *Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 358–365. IEEE Computer Society, 1999.

[178] S. Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In B. Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Proceedings of the 5th International Workshop on Security Protocols*, pages 79–90. Springer, 1997.

[179] P. MacKenzie. More efficient password-authenticated key exchange. In D. Naccache, editor, *Topics in Cryptology — CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 361–377. Springer, 2001.

[180] P. MacKenzie. On the security of the SPEKE password-authenticated key exchange protocol. http://eprint.iacr.org/2001/057, 2001.

[181] P. MacKenzie, S. Patel, and R. Swaminathan. Password-authenticated key exchange based on RSA. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 599–613. Springer, 2000.

[182] P. D. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. *Journal of Cryptology*, 19(1):27–66, 2006.

[183] M. Mambo and H. Shizuya. A note on the complexity of breaking okamoto-tanaka ID-based key exchange scheme. In *Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, pages 258–262. Springer, 1998.

[184] M. Mambo and H. Shizuya. A note on the complexity of breaking okamoto-tanaka ID-based key exchange scheme. *IEICE Trans. Commun.*, E82-A(61):77–80, 2000.

[185] K. Matsuura and H. Imai. Modification of internet key exchange resistant against denial-of-service. In *Pre-Proc. of Internet Workshop 2000*, pages 167–174, 2000.

[186] T. May. *Time-release crypto*, 1993.

[187] A. Mayer and M. Yung. Secure protocol transformation via "expansion": from two-party to groups. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 83–92. ACM Press, 1999.

[188] C. Meadows and P. F. Syverson. A formal specification of requirements for payment transactions in the SET protocol. In R. Hirschfeld, editor, *Financial Cryptography, Second International Conference, Proceedings*, volume 1465 of *Lecture Notes in Computer Science*, pages 122–140. Springer, 1998.

[189] A. J. Menezes, M. Qu, and S. A. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Workshop on selected areas in cryptography (SAC 1995)*, pages 22–32, 1995.

[190] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[191] R. C. Merkle. Secure communications over insecure channels. *Commun. ACM*, 21(4):294–299, 1978.

[192] R. C. Merkle. One way hash functions and DES. In G. Brassard, editor, *Advances in Cryptology — CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1990.

[193] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In M. Naor, editor, *Proceedings of the First Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 133–151. Springer, 2004.

[194] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology—CRYPTO 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1986.

[195] C. J. Mitchell, M. Ward, and P. Wilson. On key control in key agreement protocols. *Electronics Letters*, 34:980–981, 1998.

[196] M.Tatebayashi, N. Matsuzaki, and D. Newman Jr. Key distribution protocol for digital mobile communication systems. In G. Brassard, editor, *Advances in Cryptology — CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 324–334. Springer, 1989.

[197] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.

[198] M. H. Nguyen and S. P. Vadhan. Simpler session-key generation from short random passwords. In M. Noar, editor, *Proceedings of the First Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2004.

[199] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2002.

[200] E. Okamoto. Key distribution systems based on identification information. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO 1987*, volume 283 of *Lecture Notes in Computer Science*, pages 194–202. Springer, 1988.

[201] E. Okamoto and K. Tanaka. Key distribution system based on identification information. *IEEE Journal on Selected Areas in Communications*, 8(1):4–11, 1990.

[202] H. Orman. The Oakley key determination protocol. IETF RFC 2412, 1998.

[203] B. Klein M. Otten and T. Beth. Conference key distribution protocols in distributed systems. In P. G. Farrell, editor, *Cryptography and Coding, 4th IMA International Conference, Proceedings*, pages 225–241. Springer, 1995.

[204] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, 1987.

[205] C. Park. On certificate-based security protocols for the wireless mobile communication systems. *IEEE Networks*, 11(9):50–55, 1997.

[206] S. Pasini and S. Vaudenay. SAS-Based authenticated key agreement. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 395–409. Springer, 2006.

[207] S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 236–247. IEEE Computer Society, 1997.

[208] O. Pereira and J. J. Quisquater. A security analysis of the cliques protocols suites. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, pages 73–81. IEEE Computer Society, 2001.

[209] J. Pieprzyk and C.-H. Li. Multiparty key agreement protocols. *IEE Proceedings - Computers and Digital Techniques*, 147(4):229–236, 2000.

[210] G. Price. A general attack model on hash-based client puzzles. In K. G. Paterson, editor, *Cryptography and Coding, 9th IMA International Conference, Proceedings*, volume 2898 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2003.

[211] M. Raimondo and R. Gennaro. Provably secure threshold password-authenticated key exchange. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2003.

[212] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT LCS, 1996.

[213] S. Saeednia. Identity-based and self-certified key-exchange protocols. In *Proceedings of the Second Australasian Conference on Information Security and Privacy*, pages 303–313. Springer, 1997.

[214] S. Saeednia. Improvement of Günther's identity-based key exchange protocol. *Electronics Letters*, 36(18):1535–1536, 2000.

[215] M. Satyanarayanan. Integrating security in a large distributed system. *ACM Trans. Comput. Syst.*, 7(3):247–280, 1989.

[216] S. Shin, K. Kobara, and H. Imai. Efficient and leakage-resilient authenticated key transport protocol based on RSA. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005*, volume 3531 of *Lecture Notes in Computer Science*, pages 269–284. Springer, 2005.

[217] V. Shoup. On formal models for secure key exchange. Technical report, IBM Research Report RZ 3120, 1998.

[218] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. http://shoup.net/papers/, 2006.

[219] The SET standard Specification. SET Secure Electronic Transaction LLC. http://www.setco.org, 1997.

[220] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconference system. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 1998*, volume 403 of *Lecture Notes in Computer Science*, pages 520–528. Springer, 1998.

[221] M. Steiner, G. Tsudik, and M. Waidner. Refinement and extension of encrypted key exchange. *Operating Systems Review*, 29(3):22–30, 1995.

[222] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM conference on Computer and communications security*, pages 31–37. ACM Press, 1996.

[223] D. Stinson. *Cryptography Theory and Practice*. CRC Press, Inc., second edition, 2002.

[224] M. Strangio. Efficient diffie-hellmann two-party key agreement protocols based on elliptic curves. In H. Haddad, L. Liebrock, A. Omicini, and R. Wainwright, editors, *Proceedings of the 2005 ACM Symposium on Applied Computing*, pages 324–331. ACM, 2005.

[225] M. Strangio. On the resilience of key agreement protocols to key compromise impersonation. In A. Atzeni and A. Lioy, editors, *Public Key Infrastructure, Third European PKI Workshop: Theory and Practice, Proceedings*, volume 4043 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2006.

[226] M. Strangio. An optimal round two-party password-authenticated key agreement protocol. In *Proceedings of the the First International Conference on Availability, Reliability and Security*, pages 216–223. IEEE Computer Society, 2006.

[227] Q. Tang. On the security of three versions of the WAI protocol in Chinese WLAN implementation plan. Cryptology ePrint Archive: Report 2007/122, 2007.

[228] Q. Tang and L. Chen. Weaknesses in two group Diffie-Hellman key exchange protocols. Cryptology ePrint Archive: Report 2005/197, 2005.

[229] Q. Tang and K. R. Choo. Secure password-based authenticated group key agreement for data-sharing peer-to-peer networks. In J. Zhou, M. Yung, and F. Bao, editors, *Applied Cryptography and Network Security, 4th International Conference*, volume 3989 of *Lecture Notes in Computer Science*, pages 162–177, 2006.

[230] Q. Tang and C. J. Mitchell. Rethinking the security of some authenticated group key agreement schemes. Cryptology ePrint Archive: Report 2004/363, 2004.

[231] Q. Tang and C. J. Mitchell. Efficient compilers for authenticated group key exchange. In Y. Hao, J. Liu, Y. Wang, Y. Cheung, H. Yin, L. Jiao, J. Ma, and Y. Jiao, editors, *Computational Intelligence and Security, International Conference*, volume 3802 of *Lecture Notes in Computer Science*, pages 192–197. Springer, 2005.

[232] Q. Tang and C. J. Mitchell. Enhanced password-based key establishment protocol. Cryptology ePrint Archive: Report 2005/141, 2005.

[233] Q. Tang and C. J. Mitchell. On the security of some password-based key agreement schemes. Cryptology ePrint Archive: Report 2005/156, 2005.

[234] Q. Tang and C. J. Mitchell. On the security of some password-based key agreement schemes. In Y. Hao, J. Liu, Y. Wang, Y. Cheung, H. Yin, L. Jiao, J. Ma, and Y. Jiao, editors, *Computational Intelligence and Security, International Conference*, volume 3802 of *Lecture Notes in Computer Science*, pages 149–154. Springer, 2005.

[235] Q. Tang and C. J. Mitchell. Security properties of two authenticated conference key agreement protocols. In S. Qing, W. Mao, J. Lopez, and G. Wang, editors, *Information and Communications Security, 7th International Conference*, volume 3783 of *Lecture Notes in Computer Science*, pages 304–314. Springer, 2005.

[236] Q. Tang and C. J. Mitchell. Weaknesses in a leakage-resilient authenticated key transport protocol. Cryptology ePrint Archive: Report 2005/173, 2005.

[237] Q. Tang and C. J. Mitchell. Cryptanalysis of a hybrid authentication protocol for large mobile networks. *Journal of Systems and Software*, 79(4):496–501, 2006.

[238] Y. Tseng. A secure authenticated group key agreement protocol for resource-limited mobile devices. *Comput. J.*, 50(1):41–52, 2007.

[239] G. Tsudik and E. Herreweghen. Some remarks on protecting weak keys and poorly-chosen secrets from guessing attacks. In *Symposium on Reliable Distributed Systems*, pages 136–142, 1993.

[240] W. Tzeng. A practical and secure-fault-tolerant conferenc-key agreement protocol. In H. Imai and Y. Zheng, editors, *Proceedings of the third International Workshop on Practice and Theory in Public Key Cryptosystems*, pages 1–13. Springer, 2000.

[241] W. Tzeng. A secure fault-tolerant conference-key agreement protocol. *IEEE Transactions on Computers*, 51(4):373–379, 2002.

[242] W. Tzeng and Z. Tzeng. Round-efficient conference key agreement protocols with provable security. In *Advances in Cryptology — ASIACRYPT 2000*, pages 614–628. Springer, 2000.

[243] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.

[244] X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In V. Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[245] X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *Advances in Cryptology — EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[246] X. Wang, H. Yu, and Y. L. Yin. Efficient collision search attacks on SHA-0. In V. Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[247] R. Winternitz. A secure one-way hash function built from DES. In *Proceedings of the IEEE Symposium on Information Security and Privacy*, pages 88–90. IEEE Press, 1984.

[248] D. S. Wong and A. H. Chan. Efficient and mutually authenticated key exchange for low power computing devices. In C. Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 272–289. Springer, 2001.

[249] T. Wu. The secure remote password protocol. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 1998*, pages 97–111, 1998.

[250] Y. Yacobi. A key distribution "paradox". In A. Menezes and S. Vanstone, editors, *Advances in Cryptology — CRYPTO 1990*, volume 537 of *Lecture Notes in Computer Science*, pages 268–273. Springer, 1990.

[251] Y. Yacobi and M. Beller. Batch Diffie-Hellman key agreement systems. *Journal of Cryptology*, 10(2):89–96, 1997.

[252] S. Yen and M. Liu. High performance nounce-based authentication and key distribution protocols against password guessing attacks. *IEICE Transactions Fundamentals*, E80-A(11):2209–2217, 1997.

[253] F. G. Zhang and X. F. Chen. Attacks on two ID-based authenticated group key agreement schemes. Cryptology ePrint Archive, Report 2003/259, 2003.

[254] H. Zhou, L. Fan, and J. Li. Remarks on unknown key-share attack on authenticated multiple-key agreement protocol. *Electronics Letters*, 39(17):1248–1249, 2003.