國立交通大學

資訊工程學系

博 士 論 文

可證明安全的公開金鑰密碼系統與通行碼驗證金鑰交換

Provably Secure Public Key Cryptosystems and Password Authenticated Key Exchange Protocols

研 究 生：張庭毅

指導教授：楊維邦　教授

黃明祥　教授

中 華 民 國 九 十 五 年 十 二 月

可證明安全的公開金鑰密碼系統與通行碼驗證金鑰交換

Provably Secure Public Key Cryptosystems and Password Authenticated Key Exchange Protocols

研 究 生：張庭毅　　Student：Ting-Yi Chang

指導教授：楊維邦 博士　　Advisor：Dr. Wei-Pang Yang

黃明祥 博士　　Dr. Min-Shiang Hwang

國 立 交 通 大 學

資 訊 工 程 學 系

博 士 論 文

A Dissertation Submitted to

Department of Computer Science

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

December 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年十二月

可證明安全的公開金鑰密碼系統與通行碼驗證金鑰交換

學生：張庭毅　　指導教授：楊維邦 博士
黃明祥 博士

國立交通大學資訊科學與工程研究所　博士班

摘要

在本篇論文中，我們探討兩個主題：公開金鑰密碼系統與通行碼驗證金鑰交換。

公開金鑰密碼系統：在ElGamal加密系統中，當要加密的明文大於它的模數p時，明文必須被切割成數個片段，使得每一個片段都必須小於p，並針對每一個片段進行ElGamal加密。Hwang等人提出一個新的方法稱為ElGamal-like加密系統，目的是在加密較大明文時，具有效率。我們將指出，不論該系統是否運算在二次剩餘下，ElGamal-like加密系統並不符合IND-CPA且在加解密過程中會有機率導致失敗。

為了達到加密較大明文時能較有效率，我們提出一個轉換方式，將符合IND-CPA的ElGamal加密系統轉換成在隨機神諭模式下，符合IND-CCA2，稱為ElGamal-extended加密系統。為了證明不論明文長度為何，在加密過程只產生兩個亂數值是安全的，我們定義一個新的安全符號，稱為$\text{IND-CPA}_{\text{PAIR}}$。ElGamal-extended加密系統在加解密過程中，運算複雜度及所需的資料傳輸都比其他的加密系統有效率。

通行碼驗證金鑰交換：允許兩端（如一端為客戶端，另一端為伺服器端）經由人類可記憶的通行碼在非安全的通道上建立會議金鑰，藉由該

會議金鑰來建立安全認證的通道。我們首先指出部份提出的通行碼驗證金鑰交換方法，遭受到偽造認證攻擊、離線通行碼猜測攻擊及無法提供完整性順向機密。

我們進一步提出一個簡易的通行碼驗證金鑰交換，其對稱加密是經由一個遮罩產生函式，也就是將要傳遞的訊息乘上通行碼的雜湊結果。此方法的安全證明在Bellare-Pointcheval-Rogaway安全模式底下並假設計算的Diffie-Hellman問題是困難的且雜湊函式為隨機神諭。同時，我們進一步提出一個新的保護通行碼更換協定，在這方法中，允許使用者任意更換其通行碼。

關鍵字：適應性的選擇密文攻擊，認證，選擇明文攻擊，Diffie-Hellman問題，選擇密文攻擊，偽造認證攻擊，不可辨識，金鑰交換，不可延展，離線通行碼猜測，單向，通行碼，可證明安全，公開金鑰密碼系統，隨機神諭。

Provably Secure Public Key Cryptosystems and Password Authenticated Key Exchange Protocols

Student: Ting-Yi Chang Advisor: Dr. Wei-Pang Yang
Dr. Min-Shiang Hwang

Institute of Computer Science and Engineering
National Chiao Tung University

ABSTRACT

In this thesis, we focus on two topics: public key cryptosystems and password authenticated key exchange protocols.

Public Key Cryptosystems. In the ElGamal cryptosystem, when the plaintext is lager than the modulus p, it should be divided into several pieces which are smaller than p and then each piece is applied to ElGamal cryptosystem one by one. Hwang et al. proposed an ElGamal-like cryptosystem for encrypting a large plaintext efficiently. However, we show that their scheme is insecure against IND-CPA whether the cryptosystem is operated in the quadratic residue modulo p or not. Moreover, the encryption and / or decryption in their scheme have the probability to be failed.

In order to encrypt a large plaintext efficiently, we present an efficient conversion from IND-CPA secure ElGamal encryption scheme to a IND-CCA2 secure extension of the ElGamal encryption scheme in the random oracle model, called the ElGamal-extension cryptosystem. To demonstrate that the ElGamal-Extension cryptosystem is secure using only two random numbers no matter what the length of a plaintext, a new security notation IND-CPA_{PAIR} is constructed. The proposed scheme is more efficient than other

cryptosystems in terms of computational complexity and the amount of data transformation.

Password Authenticated Key Exchange Protocols. A password authenticated key exchange (PAKE) protocol allows two parties (a client and a server) to establish a session key when the secret key used for authentication is a human-memorable password. We show some (PAKE) schemes are vulnerable to the forged authenticator attacks, off-line password guessing attacks, and do not provide perfect forward secrecy.

We present a simple PAKE protocol which was conjectured secure when the symmetric-encryption primitive is instantiated via a mask generation function that is the product of the message with a hash of the password. This protocol is secure in the Ballare-Poincheval-Rogaway security model under the assumption that the computational Diffie-Hellman problem is hard and that the hash functions closely behaves like a random oracle. At the same time, we propose a new protected password change (PPC) protocol. The PPC protocol offers users the freedom of changing passwords at will.

Keywords: Adaptive chosen-ciphertext attack, authentication, chosen-plaintext attack, Diffie-Hellman problem, chosen-ciphertext attack, forged authenticator attack, indistinguishable, key exchange, non-malleability, off-line password guessing attack, one-wayness, password, provably secure, public key cryptosystem, random oracle.

誌　謝

論文得以順利完成，要特別感謝我的兩位指導教授楊維邦教授及黃明祥教授，不論是研究上或為人處事之道，對學生的悉心指導與鼓勵。在兩位指導教授的引領下，讓學生了解如何作好研究，提供很多寶貴經驗，使學生能夠順利取得學位及學到作研究的態度及方法，並在學生求學的過程中，感到充實，在此，向兩位老師致上最深切的感謝。

感謝口試委員楊朝成教授、吳宗成教授、官大智教授、洪國寶、葉義雄教授及袁賢銘教授對於本論文不吝指教及提供許多寶貴的建議，在此，由衷感謝。

最後，我要特別感謝我母親、姐姐，感謝家人多年來的栽培與支持，使我能很平順的完成學業；使我得以專注於我的研究上，此成果與我的家人分享。

I would like to thank Dr. Wei-Pang Yang and Dr. Min-Shiang Hwang, my supervisors, for their many suggestions and constant support during this research.

In addition, I wish to thank the following: Professor C. C. Yang, Professor T. C. Wu, Professor D. J. Guan, Professor G. Horng, Professor Y. S. Yeh and Professor S. M. Yuan. They gave me many suggestions that make a better perspective on my own results.

Of course, I am grateful to my mom and sister for their patience and *love*. Without them this work would never have come into existence (literally). In particular, my wife keep my family will, it makes me more concentrate my mind in this research.

台灣，新竹　　張庭毅

Hsinchu, Taiwan　　Ting-Yi Chang

December, 2006

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Cryptographic Schemes and Their Security

Numerous cryptographic schemes have been proposed to meet various desirable security and performance requirements. It is very important to design robust and versatile cryptographic schemes which can serve as sound security primitives. However, many schemes were subsequently found to be flawed, and were then either modified to withstand the new attacks or totally abandoned. The designers do all they can to think a lot of attacks and fight for those attacks one by one. Baek [6] gives a good example is that Bleichenbacher's attack [17] on the RSA [69] encryption standard PKCS #1 which was implemented in the widely-used Secure Socket Layer (SSL) protocol. It seems the attacked PKCS #1 had been constructed through intuitive heuristics rather than a rigorous analysis. Via that intuitive heuristics analysis, we call those schemes are *attack-response secure*. Consequently, Bleichen-

bacher's attack suggests that the heuristic approach to design cryptographic schemes can be risky and the security of cryptographic schemes should be evaluated very carefully before they are deployed.

How do we know that a cryptographic scheme be secure? A sound approach to evaluating the security of cryptographic schemes or protocols already exists. This approach is called "provable security" and stems from Goldwasser and Micali's [35] pioneering work on public key encryption schemes that hides all partial information about plaintext. According to Stinson [79], the provable security approach can be described as follows:

> "This approach is to provide evidence of security by reducing the security of the cryptosystem to some well-studied problem that is thought to be difficult. For example, it may be able to prove a statement of the type 'a given cryptosystem is secure if a given integer N cannot be factored.' Cryptosystems of this type are sometimes called provably secure."

A reductionist approach to evaluate the security of cryptographic schemes is popular. One shows with mathematical rigor that any attacker that can break the cryptographic scheme can be transformed into an efficient algorithm to solve the underlying well-studied problem, e.g., integer factorization problem, discrete logarithm problem, that is widely believed to be very hard. We call such assumption is the existence *primitive cryptographic assumptions*. Are those assumptions are correct? The answer will give at least until the famous P$\stackrel{?}{=}$NP question is resolved. Unfortunately, the current state of the art in the theory of computational complexity is such that one cannot hope to prove the truth of these computational hardness assumptions [33].

Turning this logic around: Via the reduction, we have constructed an algorithm that solves the underlying well-studied problem. However, we know that the problem is difficult. We give an concept in Figure 1.1 as follows.

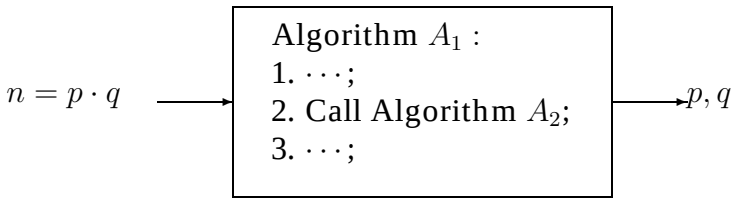


Figure 1.1: Reduction approach

Algorithm A_1: an algorithm for solving integer factorization problem,

Algorithm A_2: an algorithm for braking cryptographic scheme.

A security proof in this style, in addition to the name *reduction to contradiction*, is also called a *reduction proof*. Via that analysis, we call the scheme is *provably secure in the standard model* [35, 79]. We give the notation in the theory of computational complexity [56] as follows. $A \leq_p B$: problem A is reducible to B in a polynomial-time if (1) Given an instance $\mathcal{I}_A$ of A we can construct an instance $\mathcal{I}_B$ of B such that the solution $B(\mathcal{I}_B)$ can be converted into the solution $A(\mathcal{I}_A)$, and (2) Both the construction of the instance and the conversion of the solutions can be done in polynomial time.

If $A \leq_p B$, then problem A is no harder to solve than B. If $A \leq_p B$ and $B \leq_p A$, two problems are equal, denoted as $A \equiv_p B$. Back to the cryptographic schemes, the notation in the theory of computational complexity is as follows.

[***Theorem***]

primitive cryptographic assumptions $\leq_p$ breaking cryptographic scheme

[***Proof***]

If there is an algorithm A_2 can break the cryptographic scheme, then we can construct an algorithm A_1 to break (solve) the primitive cryptographic assumption (hardness problem).

To design a cryptographic scheme and prove it secure in this way is no easy task, especially if one wants to have a *practical* cryptographic scheme. The provable security approach was beginning to be applied to the analysis of practical schemes, it was found that a major stumbling block to providing security proofs for these schemes involved the modelling of *one-way cryptographic hash functions*. Such functions were used in many schemes, i.e. for the purpose of collision-resistant compression (hashing) of information before the application of a digital signature, for producing authentication, for checking validity of chiphertexts, and other cryptographic values without leaking information on the secret hashed value via the hash function output. To make this task more manageable, an *ideal hash function* [12] named *random oracle* is proposed. It is a powerful and imaginary function, which is deterministic and efficient and has uniform output values. This result of this approach is a reduction proof in the above sense, but the proof is only valid in a *parallel universe* where *random oracle* exists. We call the scheme via the above analysis is *provably secure in the random oracle model.*

[***Theorem***]

primitive cryptographic assumptions $\wedge$ random oracle

$\leq_p$ breaking cryptographic scheme

[***Proof***]

If there is an algorithm A_2 can break the cryptographic scheme in the random oracle model, then we can construct an algorithm A_1 to break (solve) the primitive cryptographic assumption (hardness problem).

The existence of *random oracle* is not a hardness assumption like integer factorization problem. In the real word, random oracles are replaced by hash functions (or pseudo-random functions) using in schemes which are provably secure in the random oracle model. However, an random oracle model-based technique for security proof is a useful test-bed and often gives a better performance [7]; cryptographic schemes which do not perform well on the test-bed should be dumped.

The following are two important properties for *reduction to contradiction* [55].

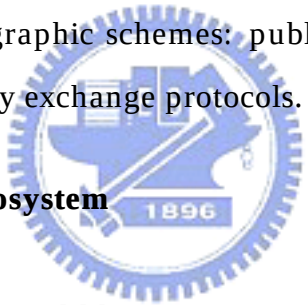
- The reduction should be efficient (in a polynomial-time). For example, an algorithm A_2 breaking the cryptographic scheme in 10^{-6} seconds and the reduction is required 2^3 of the security parameter 1024-bit to construct an algorithm A_1 solving the hard problem. Then the time complexity for the reduction is at the level of $1024^8 = 2^{80}$. The time complexity of A_1 for solving the hard problem is 38 billion years.
- The assumptions which are required for a cryptographic scheme should be as weak as possible. Weaker assumptions are easier to satisfy using more practical and available cryptographic constructions, which provide a higher security confidence than those using stronger assumptions.

Since Goldwasser and Micali's work, there have been a number of research

works taking this approach and it has become a paradigm of cryptographic research. As a consequence, and possibly affected by the negative results on the security of the past cryptographic standards, e.g, [1], [40], and [17], today's standard organizations such as ISO-IEC [73], P1363 [63], and NESSIE [29] strongly recommend that a precise security analysis based on the provable security approach should be included in a proposal of new cryptographic schemes or protocols.

1.1.2 Scope of This Thesis

We have discuss the provable security approach above. It is not only important to analyze the security of a designed cryptographic scheme but also helps to design new ones, with high level of security guarantee. In this thesis, we aim at two cryptographic schemes: public key cryptosystems and password authenticated key exchange protocols.



PART I: Public Key Cryptosystem

1. We show that the ElGamal-like cryptosystem [44] for encrypting large messages is insecure, which is attack-response secure. However, their motivation is good, since the ElGamal cryptosystem [28] encrypts large messages is inefficient.

2. In order to encrypt a large message efficiently in the ElGamal cryptosystem, we propose a new cryptosystem, called ElGamal-extension and prove its security in the random oracle model.

PART II: Password Authenticated Key Exchange Protocol

1. Many password authenticated key exchange PAKE protocols are proposed. Most of them belong to attack-response secure. We show that Tseng [81], Ku and Wang [49], Tseng, Jan and Chien [82], Hwang and Yeh [43] schemes are insecure.

2. We present a simple PAKE protocol which was conjectured secure when the symmetric-encryption primitive is instantiated via a mask generation function that is the product of the message with a hash of the password. This protocol is secure in the Ballare-Poincheval-Rogaway security model [10] under the assumption that the computational Diffie-Hellman problem is hard and that the hash functions closely behaves like a random oracle. At the same time, we propose a new protected password change (PPC) protocol. The PPC protocol offers users the freedom of changing passwords at will.

1.2 Overview of Chapters

1.2.1 Chapter 2: Background

In Chapter 2, we survey the background theory on which the subject matter of the rest of the public key cryptosystem is based. First, we review the basic of public key cryptosystem and some computational primitives such as integer factorization, discrete logarithm, various Diffie-Hellman problems. We then study the important security notations for public key cryptosystems by using the pair GOAL={OW, IND, CPA} and ATK={CPA, CCA1, CCA2} [8]. We then study the random oracle model [12], which is somewhat con-

troversial but is an important ingredient of the practice-oriented probable security paradigm in which one can design efficient probably-secure cryptographic schemes [7]. A special security notation PA is introduced, which is proposed in the random oracle model. PA has some properties, for example, a cryptosystem meets PA and IND-CPA implies it meets IND-CCA2.

1.2.2 Chapter 3: Security Analysis of ElGamal-Like Cryptosystem

In Chapter 3, we first give a brief review of the famous ElGamal cryptosystem [28]. It has been proven to meet IND-CPA in the quadratic residue under the Diffie-Hellman assumption [83]. In order to state our results clearly and precisely in breaking the the ElGamal cryptosystem [44] in Section 3.3. We first show that the ElGamal cryptosystem is insecure in the IND-CPA sense if the operations are not in the quadratic residue in Section 3.2. In order to efficiently encrypt a large message in the ElGamal cryptosystem, Hwang, Chang, and Hwang [44] proposed an ElGamal-like cryptosytem and declare that their scheme is secure in chosen-plaintext attacks. However, In Section 3.4, we separately show that the ElGmal-like cryptosystem is insecure in the IND-CPA sense no matter in the quadratic residue and not in the quadratic residue.

1.2.3 Chapter 4: An ElGamal-Extension Cryptosystem

In Chapter 4, we propose a new ElGmal-extension cryptosystem and prove it is secure in the IND-CCA2 sense in the random oracle model, which has the following advantages:

- It is only necessary to generate two random numbers. The total number of modular exponentiations is 4/2 in the encryption/decryption, which is not be increased by the number of plaintexts. Only some low computational complexity operations such as random function operations and modular multiplications are needed.
- The size of ciphtertext is smaller when the plaintext is large enough.
- It is secure in the IND-CCA2 sense, which provides a higher security level than that of IND-CPA achieved by the ElGamal encryption scheme.

We then design a special security notation $\text{IND-CPA}_{\text{PAIR}}$ and show the proposed scheme can achieve it. Then, we compare the computational complexity and ciphertext size of our scheme with those of some cryptosystems, which achieve the same security level IND-CCA2.

1.2.4 Chapter 5: Password Authenticated Key Exchange Protocols

In Chapter 5, we survey the background theory on which the subject matter of the rest of the password authenticated key exchange protocol is based.

We study the important security model [10] proposed by Bellare, Pointcheval, and Rogaway for password authenticated key exchange protocols, which defines the adversary's capabilities such as passive attacks, active attacks, known-key attacks, password guessing attacks, etc. and the goals such as mutual authentication, authenticated key exchange semantic security.

we show that Tseng [81], Ku and Wang [49], Tseng, Jan and Chien [82], Hwang and Yeh [43] password authenticated key exchange protocols are insecure.

1.2.5 Chapter 6: Simple Password Authenticated Key Exchange and Protected Password Change Protocols

In Section 5.4, we shall present a simple password authenticated key exchange protocol by modifying the Yeh-Sun scheme [85]. At the same time, we shall also present a new protected password change protocol which unlike the previously proposed schemes [47, 49, 53, 70, 81, 85] where the parties cannot arbitrarily change their own passwords, offers users the freedom of changing passwords at will.

1.2.6 Chapter 7: Conclusions

Finally, the concluding remarks of this thesis will be made in Chapter 6.

Chapter 2

Background

2.1 Introduction

Cryptosystems are classified as symmertic cryptosystems and asymmertic (public key) cryptosystems. In symmertic cryptosystems, , such as DES [76] and Rijndael [23, 24, 25], use the common secret key to encrypt plaintext and to decrypt ciphertext. This brings two difficulties as follows.

- To privately distribute the secret keys.
- To management a large number of secret keys. For example, if there are n users who want exchange confident data, then $\frac{n(n-1)}{2}$ secret keys are needed. This number increases rapidly as the number of user grows.

In public key (asymmertic) cryptosystems, each user creates a pair of keys, one of which is published in a public directory while the one is to be kept secret. The publicized key, referred to as *public key*, is used as encryption key,

but the secret key, referred to as *private (secret) key*, is used as decryption key. As a result, there is no key distribution problem and key sharing problem as in symmertic key cryptosystem. However, it is time consuming when encrypting large messages with asymmertic cryptosystems.

In this chapter, we first give an overview of public key cryptosystems and its security notations in an informal way in and the following sections give the definitions will be revisited in a formal way.

2.1.1 Public Key Cryptosystems

Here, we see that how to run public key cryptosystems, which are often divided into three phases as follows.

- Key generation phase:
  The receiver Bob creates his secret key sk_B and public key pk_B.
- Encryption phase:
  Anyone who wants to encrypt a confident message (plaintext x) to Bob by using pk_B.
- Decryption phase:
  Upon receiving the corresponding ciphertext y, Bob uses her sk_B to recover the plaintext x.

Figure 2.1 illustrates a schematic outline of a public key cryptosystem.

To reveal Bob's secret key sk_B from the public key pk_B is difficult. This property is guaranteed by the trapdoor one-way function as follows [55]:

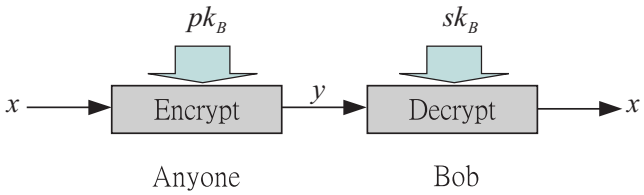


Figure 2.1: Encryption and decryption in a public key cryptosystem

Trapdoor one-way function $f_t(x) : \mathcal{X} \rightarrow \mathcal{Y}$: It is easy to compute $f_t(x)$ for all $x \in \mathcal{X}$ but difficult to invert for almost all values in $\mathcal{Y}$. If the trapdoor information t is used, then for all values $y \in \mathcal{Y}$ are easy to compute $x \in \mathcal{X}$ such that $y = f_t(x)$.

Diffie and Hellman [27] constructed a trapdoor one-way function based on modulo exponentiation. This function made it possible for distributing a common session key to be shared between two users to establish a secret communication over an channel. This protocol is the famous "Diffie-Hellman key agreement protocol". However, the first practical realization of public cryptosystem was accomplished by Revist, Shamir, and Adleman, called RSA cryptosystem [69]. Later, ElGamal [28] constructed a new public key cryptosystem based on Diffie-Hellman trapdoor one-way function.

Two typical primitives of the trapdoor one-way function are RSA [69] and ElGamal [28] as described before. The difference between ElGamal function and RSA function is that probabilistic, rather than deterministic. In a probabilistic trapdoor one-way function, when encrypting a plaintext x twice, the probability that we regain the same ciphertext y must be negligibly small. In Section 2.1.2 and Section 2.1.3, we review RSA cryptoystem and ElGamal cryptosystem, respectively.

2.1.2 RSA Cryptosystem

- Key generation phase:

  The receiver Bob creates his secret key sk_B and public key pk_B as follows.

  1. Choose two large distinct primes p and q, and compute $N = p \cdot q$
  2. Choose e that is prime to $\varphi(N)$ and $pk_B = (N, e)$, where $\varphi(N) = (p-1) \cdot (q-1)$.
  3. Choose d with $e \cdot d = 1 \bmod \varphi(N)$ and $sk_B = (N, d)$.

- Encryption phase:

  Anyone who wants to encrypt a plaintext $x < N$ to Bob by using pk_B as follow.

$$y = x^e \bmod N.$$

- Decryption phase:

  Upon receiving the corresponding ciphertext y, Bob uses his sk_B to recover the plaintext x as follows.

$$\begin{aligned} y^d &= (x^e)^d \bmod N \\ &= x \bmod N. \end{aligned}$$

The factorization of N can be reduced to an algorithm that computes d from (N, e). However, there is an open question as to whether an efficient algorithm for factoring can be derived from an efficient algorithm with inputs (N, e) and y to invert RSA [26]. Note that d should be greater than $n^{1/4}$; otherwise, a polynomial-time algorithms to compute d can be constructed [84]. For choosing large prime, the readers can refer to [39] for more details.

2.1.3 ElGamal Cryptosystm

- Key generation phase:

  The receiver Bob creates his secret key sk_B and public key pk_B.

  1. Choose a large prime p and a primitive root $g \in \mathbb{Z}_p^*$.
  2. Choose an integer s at random with range $1 \leq s \leq p-2$ and $sk_B = (p, g, s)$.
  3. Compute $Y = g^s \bmod p$ and $pk_B = (p, g, Y)$.

- Encryption phase:

  Anyone who wants to encrypt a plaintext x to Bob by using pk_B.

$$y = (y_1, y_2) = (g^r \bmod p, x \cdot Y^r \bmod p),$$

  where r is chosen at random with range $1 \leq k \leq p-2$.

- Decryption phase:

  Upon receiving the corresponding ciphertext y, Bob uses his sk_B to recover the plaintext x.

$$\begin{aligned} y_2 \cdot ((y_1)^s)^{-1} &= (x \cdot Y^r) \cdot ((g^r)^s)^{-1} \bmod p \\ &= x \bmod p. \end{aligned}$$

The ElGamal cryptosystem has been proposed several years ago and is one of the few probability encryption schemes. Until 1998, Tsiounis and Yung [83] proved the security of the ElGamal cryptosystem (operations in the quadratic residue) is actually equivalent to the decision Diffie-Hellman problem. However, we will further analysis the security for ElGamal cryptosystem when operations are not in the quadratic residue in Chapter 3.

2.2 Computational Primitives

The secure core of a public key cryptosystem is relied on the hardness of a certain computational problem. Though various computational problems have been proposed so far, we only review the integer factorization and discrete logarithm problems, which are the most widely-used computational problems in the conventional cryptographic schemes.

Integer Factorization (IF) Problem: Given $N = p \cdot q$ where p and q are large primes, find p and q.

Many public key cryptosystems based on IF-related problems are proposed, which are RSA [69], Robin [67], Okamoto-Uchiyama [62], Pointcheval [66], Paillier [64], etc.

There are several computational problems related the discrete logarithm problem as follows.

Discrete Logarithm (DL) Problem: Given a finite cyclic $\mathbb{G}$=$\{ g^1, g^2, \ldots, g^{p-1}\}$ where g be a generator of $\mathbb{G}$ and $p = |\mathbb{G}|$ is the order of $\mathbb{G}$, and a random element $r \in \mathbb{G}$, find the unique integer $i \in \mathbb{Z}_p$ such that $r = g^i$.

Computational Diffie-Hellman (CDH) problem: given a finite cyclic $\mathbb{G}$=$\{ g^1, g^2, \ldots, g^{p-1}\}$ where where g be a generator of $\mathbb{G}$ and $p = |\mathbb{G}|$ is the prime order of $\mathbb{G}$, and $g^a, g^b \in \mathbb{G}$ for random integers $a, b \in \mathbb{Z}_p$, compute $g^{ab} \in \mathbb{G}$.

Decisional Diffie-Hellman (DDH) problem: given a finite cyclic $\mathbb{G}$=$\{ g^1, g^2, \ldots, g^{p-1}\}$ where where g be a generator of $\mathbb{G}$ and $p = |\mathbb{G}|$ is the prime order of $\mathbb{G}$, and $g^a, g^b, g^c \in \mathbb{G}$ for random integers $a, b, c \in \mathbb{Z}_p$, decide $c = a \cdot b \in \mathbb{Z}_p$.

Relative to a fixed group $\mathbb{G}$ and generator g for $\mathbb{G}$, it is obvious polynomial-time reductions from the DDH problem to CDH problem, and from CDH problem to DL problem (if one can solve the DL problem then he can solve the CDH problem, and if one can solve the CDH problem then he can solve the DDH problem),

$$\text{DDH problem} \leq_p \text{CDH problem} \leq_p \text{DL problem}$$

but reductions in the reverse direction are not know. In other words, if DDH problem is hard then CDH problem is hard? and if CDH problem is hard then DL problem is hard?

$$\text{DL problem} \leq_p \text{CDH problem} \leq_p \text{DDH problem?}$$

All three problems are widely conjectured to be hard, and have been used as assumptions in proving the security of a variety of cryptographic schemes. See [18, 19, 57] for more detailed on this issue. Shoup [71] gives a heuristic evidence for the hardness of all there problem in a certain model, structured model of computation.

2.3 Security Notations

As discussed in Chapter 1, provable security evaluates the security of a given cryptographic scheme by presenting a reduction between the properly defined security notation for the scheme and the underlying primitive which is known to be secure. When an exact definition of security

was not known at the time, Rabin [67], nevertheless, has given a scheme where an eavesdropper's ability to exact the complete plaintext when given a ciphertext is computationally equivalent to factoring. This was the first scheme in which security problem was reduced to some complexity assumption. This presented a methodology of reducing the security property to a well-defined complexity assumption which, in the absence of lower bound proofs, has become the major one in cryptography.

Previously, what we have to face is that a passive attacker could break a cryptosystem only in the *all-or-nothing* (*one-wayness*) sense. However, this security notation which only deals with the case of passive attackers is not strong enough. On the contrary, the attacker maybe more active rather than passive; that is, she has more powerful capabilities to modify a ciphertext or to calculate a plaintext in some unspecified ways. In many applications, plaintexts may contain information which can be guessed easily such as in a BUY / SELL instruction to a stock broker. The attack only need to know a character (bit) of BUY / SELL instruction without decrypting the whole plaintext BUY / SELL instruction. He has the ability to determine what instruction is sent by the victim.

For example, recall the RSA cryptosystem described in Section 2.1.1. Assume that one encrypts his instruction x to a stock using the stock's public key $pk = (N, e)$ as follows.

$$y = x^e \bmod N.$$

It seems a safe bet that if an eavesdropper sees a ciphertext y corresponding to a random plaintext x, then it will impossible for that eavesdropper to figure out what x is. But when the plaintext is not random such as BUY / SELL

instruction in this example, the eavesdropper can easily to check what the corresponding plaintext is as follows.

$$\begin{cases} (\mathrm{BUY})^e \bmod N \stackrel{?}{=} y, \\ (\mathrm{SELL})^e \bmod N \stackrel{?}{=} y. \end{cases}$$

Since the RSA encryption is deterministic, the eavesdropper only encrypts the guessed plaintexts BUY / SELL and then checks its ciphertext is equal to his interception.

For the same situation, what happens in the ElGamal cryptosystem in Section 2.1.2? Note that there is a random integer r in the encryption phase and this results in the ciphertext is different for the same plaintext. Obviously, the ElGamal cryptosytem can withstand the above attack. In many applications of cryptosystems, it is often the case that user is required, upon receipt of a challenge message, to perform a decryption operation using her private key and send the decryption result back. If we give the power that the adversary can arbitrarily generate the ciphertexts and obtain the corresponding plaintexts. We can see the ElGamal cryptosystem cannot withstand the adversary with this power. For example, the adversary intercepts the ciphertext $y = (y_1, y_2) = (g^r \bmod p, (\mathrm{BUY}) \cdot Y^r \bmod p)$, which encrypts the message BUY. How can the adversary know the plaintext is BUY? He can generate a ciphertext as follows:

$$\begin{aligned} y' &= (y_1, y_2/2) \\ &= (g^r \bmod p, ((\mathrm{BUY}) \cdot Y^r \bmod p)/2) \end{aligned}$$

Then, he obtains the plaintext $\mathrm{BUY}/2$ and it multiplied by 2. Obviously, the corresponding plaintext to the intercepted ciphertext is BUY.

To capture the powerful attackers, the stronger security notations are necessary and have been proposed. Bellare et al. [8] uses the pair *goal* (GOAL) and *adversary models* (ATK) to define the security notations of public key cryptosystems and describe the relations among them.

The goals GOAL={OW, IND, NM} are defined as follows.
One-wayness (OW): given the challenge ciphertext y, the adversary has no ability to decrypt y to obtain the whole plaintext x.
Indistinguishability (IND): given the challenge ciphertext y, the adversary has no ability to obtain any information about the plaintext x.
Non-malleability (NM): given the challenge ciphertext y, the adversary has no ability to decrypt y to get a different ciphertext y' and output a meaningful relation to relate the corresponding plaintexts x and x'.

The adversary models ATK={CPA, CCA1, CCA2} are defined as follows.
Chosen-Plaintext Attack (CPA) [35]: the adversary is only given the public key and she can obtain any ciphertext from any plaintext chosen by her. In the public key cryptosystems, this attack cannot be avoided. It is considered as a basic requirement for most provably secure public key cryptosystems.
Chosen-Ciphertext Attack (CCA1) [59]: not only given the public key, but also the adversary has to access a decryption oracle before being given the challenge ciphertext. It has also been called a *lunch-time* or *midnight attack*.
Adaptive Chosen-Ciphertext Attack (CCA2) [68]: The adversary queries the decryption before and after being challenged; her only restriction here is that she may not feed the decryption oracle with the challenge ciphertext itself. It has also been called a *small-hours* attack.

The following [8, 31] are the relations among those GOAL-ATK, shown in

Figure 2.2

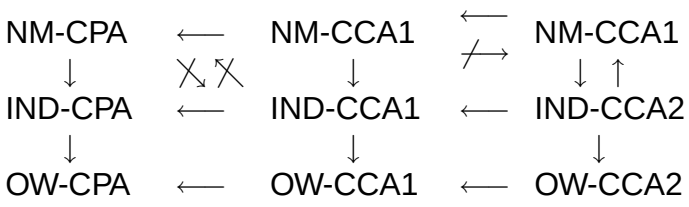


Figure 2.2: Relations among GOAL-ATK

For A, B$\in$GOAL-ATK, "A$\rightarrow$B" denotes A implies B, which means if a cryptosystem is secure in the sense of A, it is also secure in the sense of B. "A$\not\rightarrow$B" denotes A doesn't imply B, which means if a cryptosystem is secure in the sense of A, it is not always secure in the sense of B.

Bellare [7] explains precisely how to achieve provable security, which are summarized in the following steps.

Step 1. Set up a GOAL, e.g. confidentiality via encryption;

Step 2. Construct a ATK and define what it means for a cryptographic scheme be secure;

Step 3. Show by a reduction that the only way to break the security notation GOAL-ATK of cryptographic schemes is to solve computationally hard problems or break other primitives.

primitive cryptographic assumptions $\leq_p$ the proposed cryptosystem is secure in the GOAL-ATK sense

Actually, setting up security goals and constructing relevant attack models, in other words, formulating right definitions for the security of cryptographic schemes is important by itself. In the following, we survey some

important notations widely used in public key cryptography. We begin by describing the IND-ATK scenario, which is usually described in terms of the following game.

Stage 1. The public key pk and secure key sk is generated in the key generation algorithm with inputting a security parameter κ. The adversary obtains pk but not sk.

Stage 2. The adversary makes a number of arbitrary queries to a decryption oracle. Each query is a ciphertext that is decrypted by the *decryption oracle*, making use of sk of the cryptosytem. The corresponding plaintext is given to the adversary. The adversary is free to construct the ciphertexts in an arbitrary way without using the encryption algorithm.

Stage 3. The adversary arbitrarily chooses two plaintexts x_0 and x_1 with the same length $|x_0| = |x_1|$ and gives these to an *encryption oracle*. Upon receiving x_0, x_1, the encryption oracle chooses a coin $b \in \{0, 1\}$ at random, encrypt x_b and give the resulting "challenge" ciphertext y to the adversary.

Stage 4. The adversary continues to submit ciphertexts to the decryption oracle, subject to the restriction that the submitted ciphertexts are not the same as the challenge ciphertext y.

Stage 5. The adversary outputs $b' \in \{0, 1\}$, representing its "guess" of b.

- If Stage 2 and Stage 4 are omitted from the above, then ATK=CPA;
- If Stage 2 is omitted from the above, then ATK=CCA1;
- If no stage is omitted from the above, then ATK=CCA2.

The adversary's advantage in this attack scenario is defined to be

$$|\Pr[b' = b] - \frac{1}{2}|$$

A cryptosystem is defined to be secure in the IND-ATK sense if for any adversary, its advantage is negligible. A concept of IND-CCA2 is present in Figure 2.3 and an exact definition is in Section 2.7.

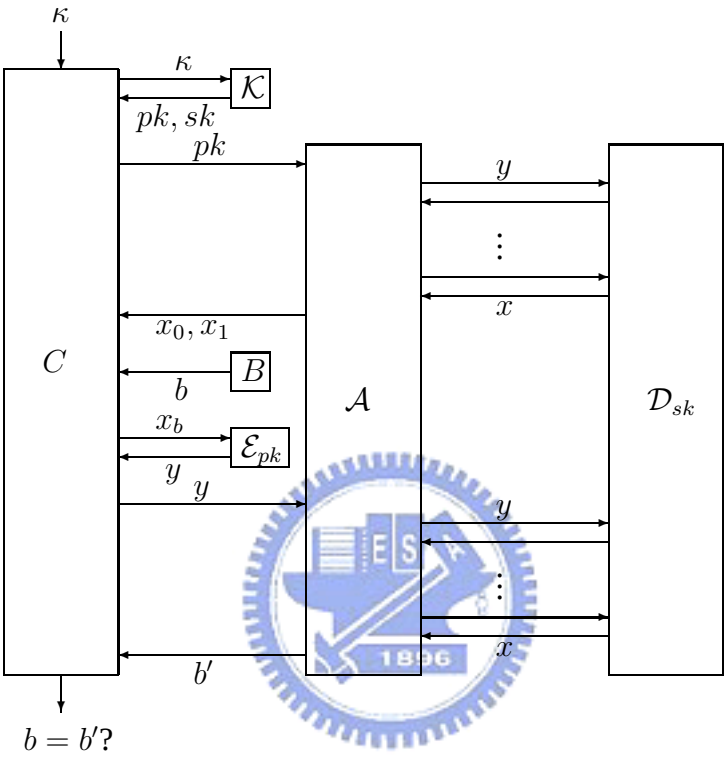


Figure 2.3: A concept of IND-CCA2, C: cryptosystem, $\mathcal{K}$: key generation algorithm, $\mathcal{E}_{pk}$: encryption algorithm, $\mathcal{D}_{sk}$: decryption algorithm, $\mathcal{A}$: adversary, B: flips a coin

Recently, Phan and Pointcheval [65] defined different levels for indistinguishability and non-malleability, and it leads to a stricter and more complex hierarchy for security notations in the public key cryptosystem. That is, an adversary can ask at most i queries before receiving the challenge and at most j queries after receiving the challenge, denotes as (i, j)-level IND / NM.

2.4 The Random Oracle Model

After formulating security notations, we shall show give the reduction from GOAL-ATK to the proposed cryptosystem. However, this is not always easy unless hash functions used in the construction of cryptosystem are assumed to behave as completely random functions. In order to introduce the random oracle model, we first recall the definitions of collision resistant hash functions and universal one-way hash function [79].

Collision Resistant Hash Functions. The definition of a collision resistant hash function is as follows. We say H is (t, ϵ)-collision resistant if for any algorithm A running in time at most t we have:

$$\Pr[(x, y) \leftarrow A : H(x) = H(y) \wedge x \neq y] \leq \epsilon.$$

That is, the algorithm A outputs two distinct values x and y such that $H(x) = H(y)$ with at most ϵ.

Universal One-Way Functions. A different from the collision resistant hash functions is that [58], the algorithm does not get to choose both x and y instead, the algorithm is given a random value x and must find a different value y such that $H(x) = H(y)$. Let $H : \{0,1\}^m \rightarrow \{0,1\}^n$ be a hash function, we say this function is (t, ϵ)-universal one-way if for all algorithms A running in time t we have:

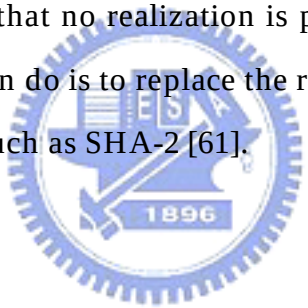
$$\Pr[x \leftarrow \{0,1\}^m; y \leftarrow A(x) : H(x) = H(y) \wedge x \neq y] \leq \epsilon.$$

This makes the adversary's job harder, meaning that the universal one-way functions are weaker than the collision-resistant hash functions. The uni-

versal one-way function families are also called target collision resistant. See [15] for recent results and further discussion.

Random Oracle Model. The random oracle model was first appeared in [30] and popularized by Bellare and Rogaway [12], gives a mathematical model if such ideal hash functions. In this model, a hash function $\mathcal{H}$ is a map from $\{0,1\}^a$ to $\{0,1\}^b$ for some special values a and b. Security proofs in this model treat the hash functions as oracles, that is, one can only query oracles to get the hash results. For each new query, the oracles respond by producing a truly random value. That is, for $x \in \{0,1\}^a$ and $y \in \{0,1\}^b$, we have the probability $\Pr[\mathcal{H}(x) = y] = \frac{1}{|y|}$. For repeated queries, the oracles respond the corresponding answer again.

However, a problem of the random oracle is that the behavior of the random oracles is so ideal so that no realization is possible. In the real world for implementation, one can do is to replace the random oracles by the conventional hash functions such as SHA-2 [61].



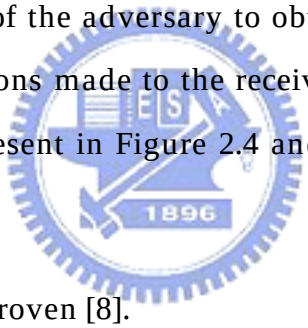
2.5 Plaintext Awareness

In the random oracle model, there is a special notation "plaintext awareness (PA)", that was suggested in [12] (calls it PA-BR) and later enhanced in [8] (calls it PA-BDPR). The idea is that an adversary has no ability to create a ciphertext y without knowing its underlying plaintext x. To capture PA-BDPR, we give a scenario as follows. Let $\mathcal{B}$ be an adversary and $\mathcal{PE}$ be the plaintext (knowledge) exactor.

Stage 1. $\mathcal{B}$ is given a public key pk, access to the random oracle $\mathcal{H}$, and an encryption oracle $\mathcal{E}_{pk}^{\mathcal{H}}$ with pk and its random oracle access. $\mathcal{B}$ outputs a ciphertext y, where y is not in the results of receiving answers by querying $\mathcal{E}_{pk}^{\mathcal{H}}$.

Stage 2. $\mathcal{PE}$ could output the corresponding plaintext x (equal to $\mathcal{D}_{sk}^{\mathcal{H}}$'s output) just by looking at pk, $\mathcal{B}$'s $\mathcal{H}$-queries and the answers to them, and the answers to $\mathcal{B}$'s queries to $\mathcal{E}_{pk}^{\mathcal{H}}$.

The existence of $\mathcal{PE}$ is, intuitively, what it means for the encryption scheme to be plaintext awareness. Obviously, $\mathcal{B}$ can do whatever $\mathcal{PE}$ was doing since, undenyable, she has access to the same things which $\mathcal{PE}$ does. Doing this, $\mathcal{B}$ would know the cleartext for any ciphertext she produces. The different between PA-BDPR and PA-BR is that PA-BR does not provide $\mathcal{E}_{pk}^{\mathcal{H}}$ to $\mathcal{B}$. This resists the ability of the adversary to obtain ciphertexts via eavesdropping on communications made to the receiver. We refer PA-BDPR as PA. A concept of PA is present in Figure 2.4 and an exact definition is in Section 2.7.



The following results are proven [8].

Proposition 1. PA-BR∧IND-CPA→IND-CCA1 *in the random oracle model.*

Proposition 2. PA∧IND-CPA→IND-CCA2 *in the random oracle model.*

Indeed, PA is designed for the random oracle model. We can see that if a scheme does not use the random oracle for which an exactor as above exists then the exactor is essentially a decryption box. [8] leaves an open question to find an analogous but achievable formulation of plaintext awareness for the standard model.

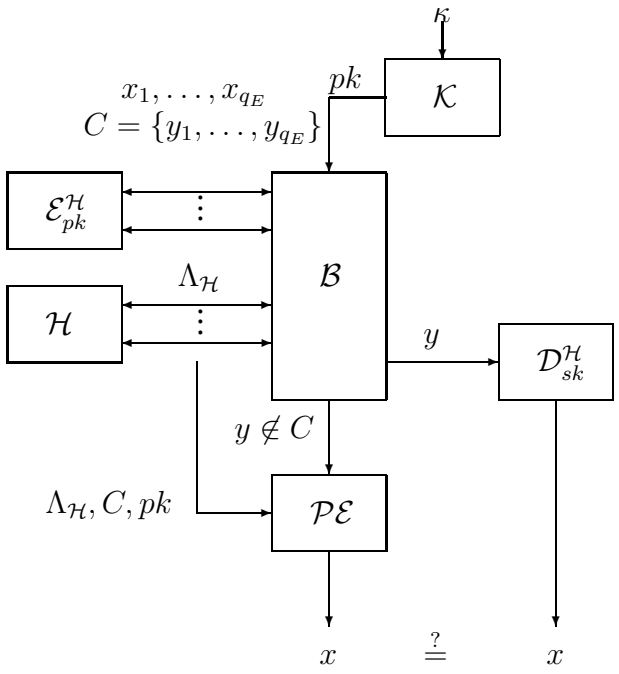


Figure 2.4: A concept of PA, $\mathcal{B}$: adversary, $\mathcal{PE}$: plaintext extractor

Herzog et al. [41] first proposed a PA without the random oracle model via the key registration. Later, Bellare and Palacio [9] removed the burden of key registration. The readers can refer [9, 41] for more details.

2.6 Related Work

Many various public key cryptosystems [64, 66, 74] have been proposed, which aim at to be secure in the stronger notations. The general methodology for formally provable security is to reduce an alleged attack on an encryption scheme to a solution of an intractable problem.

Tsiouns and Yung [83] showed that the IND-CPA of the ElGamal cryptosys-

tem operated in the quadratic residue modulo p is actually equivalent to the Decision Diffie-Hellman (DDH) problem. At the same time, they also proposed an enhanced ElGamal cryptosystem is secure in the IND-CCA2 sense under the Random Oracle (RO) model and the decision Diffie-Hellman assumption. The random oracle is assumed be an *ideally random function* when proving the security and it is replaced by a practical random-like function such as one-way hash function [12].

On the other hand, Cramer and Shoup [22] proposed a new public key cryptosystem based on the ElGamal cryptosystem, which is the first practical IND-CCA2 secure only under decision Diffie-Hellman assumption and the universal one-way hash functions, i.e., in the standard model (without the use of random oracles).

Most schemes are specified, they cannot be adopted by other schemes. There are two major conversions to convert existed trap-door one-way permutations to achieve IND-CCA2. Bellare-Rogaway conversion [13] faces on the deterministic trap-door one-way permutations such as RSA and a comment [72] revealed a flaw in that proof. Later, Fujisaki et al. [31] find a way to rescue Bellare-Rogaway conversion for the trap-door partial-domain one-way permutations being the partial-domain. On the other hand, Fujisaki-Okamoto conversion faces on the probabilistic trap-door one-way functions such as ElGamal. Both conversions are under the random oracle model and trap-door one-way function assumption.

Table 2.1 shows the different assumptions and GOAL-ATK among some related schemes. As we realize it is not pratical to implement the security proof in the RO-based technique since this kind of proof is heuristic only.

Table 2.1: Assumptions and security notations of some related schemes

Schemes	Assumptions	GOAL-ATK
ElGamal in QR_p [83]	DDH problem	IND-CPA
Tsiouns-Yung [83]	DDH problem, RO	IND-CCA2
Shoup-Gennaro [74]	DDH problem, RO	IND-CCA2
Cramer-Shoup [22]	DDH problem, UOWHF	IND-CCA2
Pointcheval [66]	DRSA problem, RO	IND-CCA2
Paillier-Pointcheval [64]	DCR problem, DPDL problem, RO	IND-CCA2
Hwang et al. [44]	DDH problem	IND-CPA
Bellare-Rogaway [13]	deterministic trap-door partial-domain one-way permutations, RO	IND-CCA2
Fujisaki-Okamoto [31]	probabilistic trap-door one-way functions ,RO	IND-CCA2

universal one-way hash function (UOWHF), dependent-RSA (DRSA) problem, decision composite residuosity (DCR) problem, decision partial discrete logarithm (DPDL) problem

However, the RO model usually has better efficiency and is still a useful test-bed to prove the security.

Hwang, Chang, and Hwang [44] consider a situation in the ElGamal cryptosystem. When the plaintext x is larger than the modulus p, it should be divided into several pieces $x_1, x_2, \cdots, x_n$ and the length of each x_i (for $i = 1$ to n) is smaller than p. Then we would need n times to apply ElGamal encryption to obtain n ciphertexts y_i's. According n ciphertexts y_i's, we also need to apply n times ElGamal decryption. This is due to withstand the known-plaintext attacks. In the known-plaintext attacks, the attacker has the ability to obtain plaintext-ciphertext pairs and uses these pair to decrypt a cipher for which she does not have the plaintext. To withstand the reduce the computational complexity and the amount of data transformation as compared to the ElGamal cryptosystem, they proposed an ElGamal-like cryptosystem for encrypting large messages and declared that the resulting scheme is in the IND-CPA sense under decision Diffie-Hellman assumption. However, we will show that their scheme is insecure in the IND-CPA sense in Section 3.3.

2.7 Definitions and Security Models

In this section, we give some definitions using in this thesis as follows.

Definition 1. *Let $x \in \mathbb{Z}_n^*$, x is said to be a quadratic residue modulo n, denoted by* QR_n.

$$\begin{aligned} \mathrm{QR}_n &= \{x \in \mathbb{Z}_n^* | \, There\ is\ a\ y \in \mathbb{Z}_n^*\ with\ x = y^2 \bmod n\}, \\ \mathrm{QNR}_n &= \mathbb{Z}_n^* - \mathrm{QR}_n. \end{aligned}$$

Definition 2. *Let p be a prime > 2, and let $x \in \mathbb{Z}$ be prime to p.*

$$\left(\frac{x}{p}\right) = \begin{cases} +1, if\ [x] \in \mathrm{QR}_p, \\ -1, if\ [x] \in \mathrm{QNR}_p, \end{cases}$$

is called the Legendre symbol of $x \bmod p$.

Definition 3. *A function $\varepsilon(k)$ is negligible if for every positive polynomial $P(k) \in \mathbb{Z}[X]$, there is k_0, such that for every $k \geq k_0, \varepsilon(k) < 1/P(k)$*

Definition 4. *Let $\mathcal{A}$ be a probabilistic algorithm and let $\mathcal{A}(a_1, a_2, \ldots; r)$ be the result of running $\mathcal{A}$ on input $a_1, a_2, \ldots$ and coins r. We let $y \leftarrow \mathcal{A}(a_1, a_2, \ldots)$ denote the experiment of choosing r at random and letting y be $\mathcal{A}(a_1, a_2, \ldots; r)$. If S is a finite set, let $a \leftarrow_R S$ be the operation of choosing a at random and uniformly from S. For probability spaces $S, T, \ldots$, the notation $\Pr[a_1 \leftarrow S; a_2 \leftarrow T; \ldots : p(a_1, a_2, \ldots)]$ denotes the probability that predicate $p(a_1, a_2, \ldots)$ is true after the ordered execution of the algorithms $a_1 \leftarrow S, a_2 \leftarrow T, \ldots$.*

Definition 5 (Computational Diffie-Hellman (CDH) problem). *Let $\mathbb{G}$ be a group of large prime q and g be the generator of $\mathbb{G}$. An algorithm algorithm $\mathcal{A}$ is said to (t, ϵ)-solve the* CDH *problem in group $\mathbb{G}$ if $\mathcal{T}$ runs in no more than time t and furthermore*

$$\Pr[x, y \leftarrow_R \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y) = g^{xy}] \geq \epsilon$$

We say that CDH *problem is hard if there is no such polynomial-time algorithm $\mathcal{A}$.*

Definition 6 (Decisional Diffie-Hellman (DDH) problem). *A distinguishing algorithm $\mathcal{T}$ is said to (t, ϵ)-solve the* DDH *problem in group $\mathbb{G}$ if $\mathcal{T}$ runs in no more than time t and furthermore*

$$|\Pr[r, s, z \leftarrow_R \mathbb{Z}_q : \mathcal{T}(g, g^r, g^s, g^z) = 1] - \Pr[r, s \leftarrow_R \mathbb{Z}_q : \mathcal{T}(g, g^r, g^s, g^{rs}) = 1]| \geq \epsilon$$

We say that DDH *problem is hard if there is no such polynomial-time algorithm $\mathcal{T}$.*

Definition 7 (Probabilistic Public-Key Encryption Scheme). *Let a triple of algorithm $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a probabilistic public key encryption scheme.*

- *The key generation algorithm $\mathcal{K}$, is a probabilistic algorithm which on input 1^k, where k is the security parameter, outputs a pair (pk, sk) of matching public and secret key.*
- *The encryption algorithm $\mathcal{E}$, is a probabilistic algorithm which on input a plaintext x and public key pk, outputs a ciphertext y.*
- *The decryption algorithm $\mathcal{D}$, is a deterministic algorithm which on input ciphertext y and the secret key sk, outputs the plaintext x.*

Definition 8 (Random Oracle Model). *Let Ω be the set of all maps from the set $\{0,1\}^*$ of finite strings to the set $\{0,1\}^\infty$ of infinite strings. $\mathcal{H} \leftarrow \Omega$ denotes as we chose map $\mathcal{H}$ from a set of an appropriate finite length $\{0,1\}^a$ to a set of an appropriate finite length of $\{0,1\}^b$, from Ω at random and uniformly, restricting the domain to $\{0,1\}^a$ and the range to the first b bits of output. By the assumption made in the random oracle model, for fix $x \in \{0,1\}^a$ and $y \in \{0,1\}^b$, then $\Pr[\mathcal{H}(x) = y] = \frac{1}{2^b}$. $\Pi = (\mathcal{K}, \mathcal{E}^{\mathcal{H}}, \mathcal{D}^{\mathcal{H}})$ denotes the $\mathcal{E}$ and $\mathcal{D}$ in public key cryptosystem are allowed to access such identical map $\mathcal{H}$, and we call this encryption scheme is defined in the random oracle model.*

Definition 9 (IND-ATK). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a pair of probabilistic algorithms for adversary, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. For* ATK={CPA,CCA1,CCA2} *and $k \in \mathbb{N}$, denote the success event of $\mathcal{A}$ for Π by*

$$\mathsf{Succ}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k) = [(pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk); b \leftarrow_R \{0,1\};$$
$$y \leftarrow \mathcal{E}_{pk}(x_b) : \mathcal{A}_2^{\mathcal{O}_2}(x_0, x_1, state, y) = b],$$

where the first two components of a triple $(x_0, x_1, state)$ *are the plaintexts with the same length* $|x_0| = |x_1|$*, and the last is state information (including the public key* pk*) and some information to be preserved. Here,* $\mathcal{O}_1(\cdot), \mathcal{O}_2(\cdot)$ *are defined as follows:*

- If ATK=CPA *then* $\mathcal{O}_1(\cdot)$*=null and* $\mathcal{O}_2(\cdot)$*=null;*

- If ATK=CCA1 *then* $\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$ *and* $\mathcal{O}_2(\cdot)$*=null;*

- If ATK=CCA2 *then* $\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$ *and* $\mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)$*.*

In the case of IND-CCA2*,* A_2 *does not ask its oracle to decrypt* y*.*

We denote the advantage of $\mathcal{A}$ *for* Π *as*

$$\mathsf{Adv}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k)] - 1.$$

We say that Π *is secure in the* IND-ATK *sense if for any adversary* $\mathcal{A}$ *being polynomial-time in* k*,* $\mathsf{Adv}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k)$ *is negligible in* k*.*

If we insist that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is allowed to access to a random oracle* $\mathcal{H}$ *in the random oracle model, we rewrite* $\mathsf{Succ}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k)$ *as follows:*

$$\mathsf{Succ}^{\mathsf{IND-ATK}}_{\mathcal{A},\Pi}(k) = [\mathcal{H} \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{H}}(pk);$$
$$b \leftarrow_R \{0,1\}; y \leftarrow \mathcal{E}^{\mathcal{H}}_{pk}(x_b) : \mathcal{A}_2^{\mathcal{O}_2,\mathcal{H}}(x_0, x_1, state, y) = b].$$

On the other hand, when we insist on the random oracle model, we rewrite $\mathcal{D}^{\mathcal{H}}_{sk}$ *instead of* $\mathcal{D}_{sk}$*.*

Definition 10 (NM-ATK). *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a pair of probabilistic algorithms for adversary,* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$*. For* ATK={CPA,CCA1,CCA2} *and* $k \in \mathbb{N}$*, denote the advantage of* $\mathcal{A}$ *for* Π *by*

$$\mathsf{Adv}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k) = |\Pr[\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k)] - \Pr[\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi,\$}(k)]|,$$

where

$$\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k) = [(pk, sk) \leftarrow \mathcal{K}(1^k); (M, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}; x, x' \leftarrow M; y \leftarrow \mathcal{E}_{pk}(x)$$
$$(R, \boldsymbol{y}) \leftarrow A_2^{\mathcal{O}_2}(M, state, y); \boldsymbol{x} \leftarrow \mathcal{D}_{sk}(\boldsymbol{y}) : (y \notin \boldsymbol{y}) \wedge (null \notin \boldsymbol{x}) \wedge R(x, \boldsymbol{x})]$$

and

$$\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi,\$}(k) = [(pk, sk) \leftarrow \mathcal{K}(1^k); (M, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}; x, x' \leftarrow M; y \leftarrow \mathcal{E}_{pk}(x)$$
$$(R, \boldsymbol{y}) \leftarrow A_2^{\mathcal{O}_2}(M, state, y); \boldsymbol{x} \leftarrow \mathcal{D}_{sk}(\boldsymbol{y}) : (y \notin \boldsymbol{y}) \wedge (null \notin \boldsymbol{x}) \wedge R(x', \boldsymbol{x})].$$

Here, $\mathcal{O}_1$, $\mathcal{O}_2$ are defined as before in Definition 9. In the case of IND-CCA2*, $\mathcal{A}_2$ does not ask its oracle to decrypt y.*

We say that M is valid if $|x| = |x'|$ for any x, x' that are given non-zero probability in the message space M.

We say that Π is secure in the NM-ATK *sense if for any adversary $\mathcal{A}$ being polynomial-time in k outputs a valid message space M samplable in polynomial in k and a relation R computable in polynomial in k, then $\mathsf{Adv}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k)$ is negligible in k.*

If we insist that $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is allowed to access to a random oracle $\mathcal{H}$ in the random oracle model, we rewrite $\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k)$ and $\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi,\$}(k)$ as follows:

$$\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi}(k) = [\mathcal{H} \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (M, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{H}}; x, x' \leftarrow M; y \leftarrow \mathcal{E}^{\mathcal{H}}_{pk}(x)$$
$$(R, \boldsymbol{y}) \leftarrow A_2^{\mathcal{O}_2,\mathcal{H}}(M, state, y); \boldsymbol{x} \leftarrow \mathcal{D}_{sk}(\boldsymbol{y}) : (y \notin \boldsymbol{y}) \wedge (null \notin \boldsymbol{x}) \wedge R(x, \boldsymbol{x})]$$

and

$$\mathsf{Succ}^{\mathsf{NM-ATK}}_{\mathcal{A},\Pi,\$}(k) = [\mathcal{H} \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (M, state) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{H}}; x, x' \leftarrow M; y \leftarrow \mathcal{E}^{\mathcal{H}}_{pk}(x)$$
$$(R, \boldsymbol{y}) \leftarrow A_2^{\mathcal{O}_2,\mathcal{H}}(M, state, y); \boldsymbol{x} \leftarrow \mathcal{D}_{sk}(\boldsymbol{y}) : (y \notin \boldsymbol{y}) \wedge (null \notin \boldsymbol{x}) \wedge R(x', \boldsymbol{x})].$$

On the other hand, when we insist on the random oracle model, we rewrite $\mathcal{D}^{\mathcal{H}}_{sk}$ instead of $\mathcal{D}_{sk}$.

Definition 11 (PA). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public key encryption scheme, let B be an adversary, let $\mathcal{PE}$ be an polynomial-time plaintext extractor. For any $k \in \mathbb{N}$*

define

$$\mathsf{Succ}^{\mathsf{PA}}_{\mathcal{PE},\mathcal{B},\Pi}(k) = \Pr[\mathcal{H} \leftarrow \Omega; (pk, sk) \leftarrow \mathcal{K}(1^k); (\Lambda_{\mathcal{H}}, C, y) \leftarrow \text{run } \mathcal{B}^{\mathcal{H},\mathcal{E}^{\mathcal{H}}_{pk}}(pk) :$$
$$\mathcal{PE}(\Lambda_{\mathcal{H}}, C, y, pk) = \mathcal{D}^{\mathcal{H}}_{sk}(y)],$$

where $\Lambda_{\mathcal{H}} = \{(h_1, H_1), \ldots, (h_{q_H}, H_{q_H})\}$, $C = \{y_1, \ldots, y_{q_E}\}$, *and* $y \notin C$.

$(\Lambda_{\mathcal{H}}, C, y) \leftarrow \text{run } \mathcal{B}^{\mathcal{H},\mathcal{E}^{\mathcal{H}}_{pk}}(pk)$ *denotes run* $\mathcal{B}$ *on input* pk, *oracles* $\mathcal{H}$, *and* $\mathcal{E}^{\mathcal{H}}_{pk}$, *recording* $\mathcal{B}$*'s interaction with its oracles.* $\Lambda_{\mathcal{H}}$ *denotes the set of all* $\mathcal{B}$*'s queries and the corresponding answers of* $\mathcal{H}$. C *denotes the set of all answers received as the result of* $\mathcal{E}^{\mathcal{H}}_{pk}$. *Here,* C *does not include the the corresponding queries from* $\mathcal{B}$. *Finally,* $\mathcal{B}$ *outputs* y.

We say that $\mathcal{PE}$ *is a* $(t, \lambda(k))$*-plaintext extractor if* $\mathsf{Succ}^{\mathsf{PA}}_{\mathcal{PE},\mathcal{B},\Pi}(k) \geq \lambda(k)$ *and* $\mathcal{PE}$ *runs within at most running time* t.

We say that Π *is secure in the sense of* PA *if* Π *is secure in the sense of* IND-CPA *and there exists a* $(t, \lambda(k))$*-plaintext extractor* $\mathcal{PE}$ *where* t *is polynomial in* k *and* $\lambda(k)$ *is overwhelming in* k.

To demonstrate that the ElGamal-Extension scheme is secure using only two random numbers, a new pair GOAL and ATK are constructed called $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$. The difference from IND-CPA is that, we also provide the adversary with the knowledge of a pair of plaintext-ciphertext. Intuitively, this pair does not provide any help for the adversary, since the adversary has ability to generate any pair she wants by herself in the public key encryption scheme. This is the refinement presented here and its purpose is explained later in Chapter 4.

Definition 12 ($\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$). *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a pair of probabilistic algo-*

rithms for adversary, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$*. For* $k \in \mathbb{N}$*, denote the success event of* $\mathcal{A}$ *for* Π *by*

$$\mathsf{Succ}^{\mathsf{IND-CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k) = [(pk, sk) \leftarrow \mathcal{K}(1^k); (x^*, x_0, x_1, state) \leftarrow \mathcal{A}_1(pk); b \leftarrow_R \{0,1\};$$
$$y \leftarrow \mathcal{E}_{pk}(x_b); y^* \leftarrow \mathcal{E}_{pk}(x^*) : \mathcal{A}_2(x_0, x_1, x^* \rightleftarrows y^*, state, y) = b].$$

We describe a supplementary explanation: $\mathcal{A}_1$ *outputs* $(x^*, x_0, x_1, state)$ *is defined as before in Definition 9 and an additional plaintext* x^**, where* $|x^*| = |x_0| = |x_1|$*. The encryption oracle encrypts* x_b *to obtain* y *according to a coin flipping* b*. It also encrypts* x^* *to obtain a ciphertext* y^**.* $\mathcal{A}_2$ *additionally has the input* $x^* \rightleftarrows y^*$*, where* $x^* \rightleftarrows y^*$ *denotes* x^* *as the corresponding plaintext of the ciphertext* y^**.*
We denote the advantage of $\mathcal{A}$ *for* Π *as*

$$\mathsf{Adv}^{\mathsf{IND-CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{IND-CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k)] - 1.$$

We say that Π *is secure in the* IND-CPA$_{\mathsf{PAIR}}$ *sense if for any adversary* $\mathcal{A}$ *being polynomial-time in* k*,* $\mathsf{Adv}^{\mathsf{IND-CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k)$ *is negligible in* k*.*

Chapter 3

Security Analysis of ElGamal-Like Cryptosystem

3.1 Review of ElGamal Cryptosystem

Though the ElGamal cryptosystem operated in the quadratic residue modulo p has been showed that is secure in the IND-CPA sense under the Diffie-Hellman assumption [83]. In order to state our results clearly and precisely in breaking the ElGamal-like cryptosystem [44] in Section 3.3, we begin with a review of the ElGamal cryptosystem which is not operated in the quadratic residue modulo p and then show that is insecure against IND-CPA.

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the ElGamal cryptosystem.

- Key generation algorithm $\mathcal{K} : (pk, sk) \leftarrow \mathcal{K}(1^k)$, $pk = (p, g, Y)$ and $sk = (p, g, s)$, where $Y = g^s \bmod p$, $|p| = k$, $1 \leq s \leq p-2$, and $\# < g >= p$. Let $\mathbb{G}_p$ be a group of prime order p of the multiplicative group $\mathbb{Z}_p^*$.

- Encryption algorithm $\mathcal{E}$:

$$(y_1, y_3) = \mathcal{E}_{pk}(x, r) = (g^r \bmod p, x \cdot Y^r \bmod p),$$

where message $x \in \{0,1\}^k$, $x < p$, and $r \leftarrow_R \{0,1\}^k$.

- Decryption algorithm $\mathcal{D}$:

$$x = \mathcal{D}_{sk}(y_1, y_3) = y_3 \cdot (y_1^s)^{-1} \bmod p.$$

3.2 Security Analysis

We can see that g is a primitive root of $\mathbb{G}_p$ by employing the key generation algorithm $\mathcal{K}$ in Seciton 3.1. Below, we first give the following lemmas [26] and then show that encryption scheme is not secure in the IND-CPA sense.

Lemma 1. *Let p be a prime > 2 and g be a primitive root of $\mathbb{Z}_p^*$. Let $[x] \in \mathbb{Z}_p^*$. Then $x \in \mathrm{QR}_p$ if and only if $x = g^\alpha \bmod p$ some even number α, $0 \leq \alpha < p-1$.*

Lemma 2. *The Legendre symbol is multiplicative in x*

$$\left(\frac{xy}{p}\right) = \left(\frac{x}{p}\right)\left(\frac{y}{p}\right)$$

It means $[xy] \in \mathrm{QR}_p$ if and only if either both $[x], [y] \in \mathrm{QR}_p$ or both $[x], [y] \in \mathrm{QNR}_p$.

Theorem 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the ElGamal cryptosystem described in Section 3.1. An adversary $\mathcal{A}$ is a (t', ϵ')-breaker for $\Pi(1^k)$ in IND-CPA if $\mathrm{Adv}^{\mathrm{CPA}}_{\mathcal{A},\Pi}(k) \geq \epsilon'$ and $\mathcal{A}$ runs within at most running time t, where*

$$\epsilon' = 1 \; and \; t' \leq t_1 + 3 \cdot t_{QR}.$$

Proof. We construct a breaking algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ as follows.

Adversary: $\mathcal{A}_1(pk)$
Obtain $\{x_0, x_1\}$, where $x_0 \in$QR$_p$ and $x_1 \in$QNR$_p$
Return $(x_0, x_1, state)$
End.

Encryption oracle: $\mathcal{O}_{EN}(x_0, x_1, pk)$
$r \leftarrow_R \mathbb{Z}_q$
$b \leftarrow_R \{0, 1\}$
$(y_1, y_3) \leftarrow \mathcal{E}_{pk}(x_b, r)$
Return (y_1, y_3)
End.

Adversary: $\mathcal{A}_2(x_0, x_1, state, (y_1, y_3))$
CASE 1: $Y \in$QR$_p$ and $y_1 \in$\{QR$_p$, QNR$_p$\}
If $y_3 \in$QR$_p$, then outputs 0
If $y_3 \in$QNR$_p$, then outputs 1
CASE 2: $y_1 \in$QR$_p$ and $Y \in$QNR$_p$
If $y_3 \in$QR$_p$, then outputs 0
If $y_3 \in$QNR$_p$, then outputs 1
CASE 3: $Y \in$QNR$_p$, $y_1 \in$QNR$_p$
If $y_3 \in$QNR$_p$, then outputs 0
If $y_3 \in$QR$_p$, then outputs 1
End.

We now analyze the successful probability of adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We define the following events.

E$_1$ be the event $(Y \in$QR$_p) \wedge (y_1 \in$\{QR$_p$, QNR$_p$\}),

E$_2$ be the event $(y_1 \in$QR$_p) \wedge (Y \in$QNR$_p$),

E$_3$ be the event $(Y \in$QNR$_p) \wedge (y_1 \in$QNR$_p$).

Let b' be the output of $\mathcal{A}_2$. For CASE 1, $Y = g^s \in$QR$_p$. By Lemma 1, s is even, no matter what $y_1 \in$QR$_p$, or $y_1 \in$QNR$_p$, we know that $Y^r = g^{sr} \in$QR$_p$.

We see that $\mathcal{A}_2$ will output the correct b'=0 (b'=1) if and only if $y_3 \in$QR$_p$ ($y_3 \in$QNR$_p$). This is due to the multiplicative property of Legendre symbol in Lemma 2 as follows.

$$\left(\frac{y_3}{p}\right) = \left(\frac{x_b}{p}\right)\left(\frac{Y^r}{p}\right)$$

Therefore, the condition probability Pr$[b = b'|\mathsf{E}_1]$=1 and the probability Pr[E_1] = 1/2. For the same reason, in CASE 2, the condition probability Pr$[b = b'|\mathsf{E}_2]$=1. Note that $(y_1 \in$QR$_p) \wedge (Y \in$QR$_p)$ is included in the event E_1 and the probability Pr$[\mathsf{E}_1] = 1/4$. For CASE 3, $Y \in$QNR$_p$ and $y_1 \in$QNR$_p$, by Lemma 1, s and r are odd, $Y^r = g^{sr} \in$QNR$_p$. $\mathcal{A}_2$ will output the correct b'=0 (b'=1) if and only if $y_3 \in$QR$_p$ ($y_3 \in$QNR$_p$). Thus, the condition probability Pr$[b = b'|\mathsf{E}_3] = 1$ and the probability Pr$[\mathsf{E}_3] = 1/4$. By the law of total probability,

$$\begin{aligned}
\Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(k)] &= \Pr[b = b'] \\
&= \sum_{i=1}^{3} \Pr[b = b'|\mathsf{E}_i] \cdot \Pr[\mathsf{E}_i] \\
&= 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 1 \cdot \frac{1}{4} \\
&= 1,
\end{aligned}$$

we have $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A},\Pi}(k)] - 1 = 1$.

Thus, we have the ability to distinguish the distinct plaintext x_0 and x_1. To secure against IND-CPA, for security parameter k, primes p and q are chosen such that $p = 2q + 1$ (q is called a Sophie-Germain prime if p is also a prime), where $|p| = k$ and $|q| = k - 1$. Then a unique subgroup $\mathbb{G}_q$ of prime order q of the multiplicative group $\mathbb{Z}_p^*$ and g of $\mathbb{G}_q$ are defined. In other words, the key generation algorithm $\mathcal{K}$ should be modified as $\widehat{\mathcal{K}}$.

- Key generation algorithm $\widehat{\mathcal{K}}$: $(pk, sk) \leftarrow \widehat{\mathcal{K}}(1^k)$, $pk = (p, g, Y)$ and $sk =$

(p, g, s), where $Y = g^s \bmod p$, $|p| = k$, $p = 2q + 1$, $\# < h >= p$, $g = h^2 \bmod p$, $s \in \mathbb{Z}_q$, and $\# < g >= q$.

We can see that g generates all the quadratic residues in $\mathbb{G}_q = \mathrm{QR}_p$. In order to make all the ciphertexts are in QR_p (the algorithm $\mathcal{A}$ cannot determine which message $x_0 \in \mathrm{QR}_p / x_1 \in \mathrm{QNR}_p$ is the corresponding plaintext according the ciphertext), there two are simple methods to achieve it:

Method 1. The messages for encrypting are always in QR_p.

Method 2. If the message x is in QR_p, then we are done. Otherwise, x is replaced by $-x = p - x \in \mathrm{QR}_p$. Since

$$(-1)^{(p-1)/2} = (-1)^q = -1 \bmod p, \text{ where } q \text{ is an odd number},$$

we have

$$(-x)^{(p-1)/2} = (-1)^{(p-1)/2} \cdot (x)^{(p-1)/2} = (-1) \cdot (-1) = 1 \bmod p.$$

A value is determined whether it is in QR_p or not can be computed efficiently by Euler's criterion in a polynomial-time. Let t_{QR} be the time of determining whether a value is in QR_p or not. Let t_1 be the time of choosing two messages $x_0 \in \mathrm{QR}_p$ and $x_1 \in \mathrm{QNR}_p$. Then, from the specification of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it runs within at most 3 times t_{QR} in CASE 2 or CASE 3. Hence, $t \leq t_1 + 3 \cdot t_{\mathrm{QR}}$ and it is in a polynomial-time. □

3.3 Review of ElGamal-Like Cryptosystem

The ElGamal cryptosystem should employ the key generation algorithm $\widehat{\mathcal{K}}$ and choose messages from the subgroup $\mathbb{G}_q$ to ensure that the IND-CPA

sense. The security had been proven [83] is actually equivalent to the DDH problem.

Recall the ElGamal cryptosystem in Section 3.1. If the plaintext x is larger than modulus p, it should be divided into $x_1, x_2, \cdots, x_n$ and the length of each x_i is smaller than $|p|$. After dividing x, each x_i is fed into the encryption algorithm $\mathcal{E}$:

$$\mathcal{E}_{pk}(y_{1,i}, y_{3,i}) = \mathcal{E}_{pk}(x_i, r_i) = (y^{r_i} \bmod p, x \cdot Y^{r_i} \bmod p).$$

Note that $r_i \neq r_j (i \neq j)$, otherwise the cryptosystem is broken by the CPA. For example, the adversary chooses the plaintext x_1, and then she feds x_1 into the encryption algorithm $\mathcal{E}$. With the knowledge of the plaintext-chiphertext $(x_1, (y_{1,1}, y_{3,1}))$, she can obtain $Y^{r_1} \bmod p$ by computing $y_{3,1} \cdot (x_1)^{-1} \bmod p$ without the secret key s. Then, she can easily reveal other plaintexts $x_2, \cdots, x_n$ by computing $x_i = y_{3,i} \cdot (y_{1,1}^s)^{-1} \bmod p$, for $i = 2$ to n. In other words, if $r_i = r_j (i \neq j)$, the cryptosystem is insecure in the OW-CPA sense.

For encrypting a large message with efficient, Hwang et al. [44] modified some parts in the ElGamal cryptosystem, called the ElGamal-like cryptosystem. However, in this section, we will show that even if the ElGamal-like cryptosystem are given the same repair in Section 3.2, it is still insecure in the IND-CPA sense.

Let $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ be the ElGamal-like cryptosystem.

- Key generation algorithm $\mathcal{K}$: $(pk, sk) \leftarrow \mathcal{K}(1^k)$, $pk = (p, g, Y)$ and $sk = (p, g, s)$, where $Y = g^s \bmod p$, $|p| = k$, $s \in \mathbb{Z}_p^*$, and $\# < g >= p$.

- Encryption algorithm $\mathcal{E}'$:

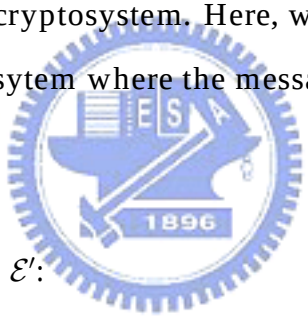
  $$(y_1, y_2, y_{3,i}) = \mathcal{E}'_{pk}(x_i, r_1, r_2) = (g^{r_1} \bmod p, g^{r_2} \bmod p, x_i \cdot (Y^{r_1} \oplus (Y^{r_2})^{2^i}) \bmod p),$$

  where message $x \in \{0,1\}^{>k}$, x is divided into $x_1, x_2, \cdots, x_n$ ($|x_1| = |x_2| = \cdots = |x_{n-1}|$, $n = \lceil |x|/k \rceil$, $|x_n| = |x| \bmod k$, and each $x_i < p$) and $r_1, r_2 \leftarrow_R \mathbb{Z}_q$. The notation $\oplus$ denotes as the bit-wise exclusive-or operation.

- Decryption algorithm $\mathcal{D}'$:

  $$\begin{aligned} x_i &= \mathcal{D}'_{sk}(y_1, y_2, y_{3,i}) = y_{3,i} \cdot (y_1^s \oplus (y_2^s)^{2^i})^{-1} \bmod p, \\ x &= x_1 x_2 \cdots x_n. \end{aligned}$$

This scheme is designed for encrypting large messages, which will more efficient than the ElGamal cryptosystem. Here, we consider the same situation in the ElGamal cryptosytem where the message $x < p$ is for encrypting as follows.



- Encryption algorithm $\mathcal{E}'$:

  $$(y_1, y_2, y_3) = \mathcal{E}'_{pk}(x, r_1, r_2) = (g^{r_1} \bmod p, g^{r_2} \bmod p, x \cdot (Y^{r_1} \oplus (Y^{r_2})^2) \bmod p),$$

  where message $x \in \{0,1\}^k$, $x < p$, and $r_1, r_2 \leftarrow_R \{0,1\}^k$.

- Decryption algorithm $\mathcal{D}'$:

  $$x = \mathcal{D}'_{sk}(y_1, y_2, y_3) = y_3 \cdot (y_1^s \oplus (y_2^s)^2)^{-1} \bmod p.$$

3.4 Security Analysis

In the following theorem, we prove that the ElGamal-like cryptosystem in Section 3.3 is insure in the CPA sense and has the probability to make that cryptosystem failed.

Theorem 2. *Let* $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ *be the ElGamal-like cryptosystem described in Section 3.3. An adversary* $\mathcal{A}'$ *is a* (t', ϵ')*-breaker for* $\Pi'(1^k)$ *in* IND-CPA *if* $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A}',\Pi'}(k) \geq \epsilon'$ *with the event* Fail *it does not occur, and* $\mathcal{A}'$ *runs within at most running time* t'*, where*

$$\epsilon' = 1 \; and \; t' \leq t_1 + 3 \cdot t_{QR}.$$

Proof. We give a simple example and then analyze the results as follows. In the key generation algorithm $\mathcal{K}$, for $p = 7$, we select a generator $g = 5$ of $\mathbb{Z}_p^*$ and thus $\mathrm{QR}_p = \{1, 2, 4\}$ and $\mathrm{QNR}_p = \{3, 5, 6\}$. By Lemma 1, $(Y^{r_2})^2 \bmod p \in \mathrm{QR}_p$. We consider the following situations.

SITUATION 1: $Y^{r_1} \bmod p \in \mathrm{QR}_p$

The values of computing $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p$ are in the set $S_1 = \{1 \oplus 1 \bmod 7, 1 \oplus 2 \bmod 7, 1 \oplus 4 \bmod 7, 2 \oplus 1 \bmod 7, 2 \oplus 2 \bmod 7, 2 \oplus 4 \bmod 7, 4 \oplus 1 \bmod 7, 4 \oplus 2 \bmod 7, 4 \oplus 4 \bmod 7\} = \{0, 3, 5, 3, 0, 6, 5, 6, 0\}$.

SITUATION 2: $Y^{r_1} \bmod p \in \mathrm{QNR}_p$

The values of computing $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p$ are in the set $S_2 = \{1 \oplus 3 \bmod 7, 1 \oplus 5 \bmod 7, 1 \oplus 6 \bmod 7, 2 \oplus 3 \bmod 7, 2 \oplus 5 \bmod 7, 2 \oplus 6 \bmod 7, 4 \oplus 3 \bmod 7, 4 \oplus 5 \bmod 7, 4 \oplus 6 \bmod 7\} = \{2, 4, 0, 1, 0, 4, 0, 1, 2\}$.

We can see that the values of $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p$ has the probability to be 0, no matter what plaintext x is input to encrypt algorithm $\mathcal{E}'$, the value of

$y_3 = x \cdot (Y^{r_1} \oplus (Y^{r_2})^2) \bmod p$ is equal to 0. The encrypt algorithm $\mathcal{E}'$ is failed, together with the decrypt algorithm $\mathcal{D}'$. We first analyze the probability of $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ crashed. Let Fail, $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p = 0$, be the event that $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ crashed. By lemma 1, if the value $Y^{r_1} = g^{sr_1} \bmod p \in \mathrm{QR}_p$, then $s \cdot r_1$ is even; that either r_1 or s are even, which happen with probability $\Pr[Y^{r_1} \in \mathrm{QR}_p] = 3/4$ and the complement $\Pr[Y^{r_1} \in \mathrm{QNR}_p] = 1/4$. We can obtain the probability of Fail as follows.

$$\begin{aligned}
\Pr[\mathsf{Fail}] &= \Pr[\mathsf{Fail}|Y^{r_1} \in \mathrm{QR}_p] \cdot \Pr[Y^{r_1} \in \mathrm{QR}_p] \\
&\quad + \Pr[\mathsf{Fail}|Y^{r_1} \in \mathrm{QNR}_p] \cdot \Pr[Y^{r_1} \in \mathrm{QNR}_p] \\
&= \frac{3}{9} \cdot \frac{3}{4} + \frac{3}{9} \cdot \frac{1}{4} \\
&= \frac{1}{3}.
\end{aligned}$$

If the encryption algorithm $\mathcal{E}'$ chooses r_1 or $r_2 \leftarrow_R \{0,1\}^k$ again to avoid the case $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p = 0$, we still can construct a breaking algorithm $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ in the IND-CPA sense for $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$.

Adversary: $\mathcal{A}'_1(pk)$
Obtain $\{x_0, x_1\}$, where $x_0 \in \mathrm{QR}_p$ and $x_1 \in \mathrm{QNR}_p$
Return $(x_0, x_1, state)$
End.

Encryption oracle: $\mathcal{O}_{EN}(x_0, x_1, pk)$
$r_1, r_2 \leftarrow_R \mathbb{Z}_q$
$b \leftarrow_R \{0,1\}$
$(y_1, y_2, y_3) = \mathcal{E}'_{pk}(x_b, r_1, r_2)$
$= (g^{r_1} \bmod p, g^{r_2} \bmod p, x_b \cdot (Y^{r_1} \oplus (Y^{r_2})^2) \bmod p)$
Return (y_1, y_2, y_3)
End.

Adversary: $\mathcal{A}'_2(x_0, x_1, state, (y_1, y_2, y_3))$
CASE 1: $Y \in \text{QR}_p$ and $y_1 \in \{\text{QR}_p, \text{QNR}_p\}$ // $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p \in \text{QR}_p$
If $y_3 \in \text{QR}_p$, then outputs 0
If $y_3 \in \text{QNR}_p$, then outputs 1
CASE 2: $y_1 \in \text{QR}_p$ and $Y \in \text{QR}_p$
If $y_3 \in \text{QR}_p$, then outputs 0
If $y_3 \in \text{QNR}_p$, then outputs 1
CASE 3: $Y \in \text{QNR}_p$, $y_1 \in \text{QNR}_p$ // $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p \in \text{QNR}_p$
If $y_3 \in \text{QNR}_p$, then outputs 0
If $y_3 \in \text{QR}_p$, then outputs 1
End.

The successful probability of adversary $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ is similar to $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in Section 3.2 if $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ is not crashed, i.e., Fail does not occur. By the multiplicative property of Legendre symbol,

$$\left(\frac{y_3}{p}\right) = \left(\frac{x_b}{p}\right)\left(\frac{Y^{r_1} \oplus (Y^{r_2})^2}{p}\right),$$

the conditional probability $\Pr[\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A}',\Pi'}(k)|\neg\mathsf{Fail}]$ is equal to 1 and $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A}',\Pi'}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A}',\Pi'}(k)|\neg\mathsf{Fail}] - 1 = 1$. For the same reason, from the specification of $\mathcal{A}'$, it runs within at most $t' \leq t_1 + 3 \cdot t_{\text{QR}}$. □

If we attempt to repair this scheme $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ as the same fashion in Section 3.2, the key generation algorithm $\mathcal{K}$ is replaced as $\widehat{\mathcal{K}}$, and then the cryptosystem becomes $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$. The following theorem will show that $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$ is still insecure in the IND-CPA sense.

Theorem 3. *Let $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$ be the ElGamal-like cryptosystem operated in QR_p. An adversary $\mathcal{A}''$ is a (t'', ϵ'')-breaker for $\Pi''(1^k)$ in* IND-CPA *if* $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A}'',\Pi''}(k) \geq \epsilon''$ *with the event* Fail *does not occur, and $\mathcal{A}''$ runs within at most running time t'', where*

$$\epsilon'' = 1 \; and \; t'' \leq t_1 + t_{QR}.$$

Proof. We also give an example for the key generation algorithm $\widehat{\mathcal{K}}$, where $q = 3$, $p = 2q+1 = 7$, $h = 5$, $g = h^2 \bmod p = 4$. Obviously, $g \in \mathrm{QR}_p$, therefore, the group is in QR_p, where $\mathrm{QR}_p = \{1, 2, 4\}$. The value of $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p$ are in the set S_1 as the same as in SITUATION 1 of Theorem 2. $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$ has the probability to fail as follows:

$$\begin{aligned} \Pr[\mathsf{Fail}] &= \Pr[\mathsf{Fail} | Y^{r_1} \in \mathrm{QR}_p] \cdot \Pr[Y^{r_1} \in \mathrm{QR}_p] \\ &= \frac{3}{9} \cdot 1 \\ &= \frac{1}{3}. \end{aligned}$$

A breaking algorithm $\mathcal{A}'' =: (\mathcal{A}''_1, \mathcal{A}''_2)$ in the IND-CPA sense for $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$ is as follows:

Adversary: $\mathcal{A}''_1(pk)$
Obtain $\{x_0, x_1\}$, where $x_0 \in \mathrm{QR}_p$ and $x_1 \in \mathrm{QNR}_p$
Return $(x_0, x_1, state)$
End.

Encryption oracle: $\mathcal{O}_{EN}(x_0, x_1, pk)$
$r_1, r_2 \leftarrow_R \mathbb{Z}_q$
$b \leftarrow_R \{0, 1\}$
$(y_1, y_2, y_3) = \mathcal{E}'_{pk}(x_b, r_1, r_2)$
$= (g^{r_1} \bmod p, g^{r_2} \bmod p, x_b \cdot (Y^{r_1} \oplus (Y^{r_2})^2) \bmod p)$
Return (y_1, y_2, y_3)
End.

Adversary: $\mathcal{A}''_2(x_0, x_1, state, (y_1, y_2, y_3))$
CASE 1: If $y_3 \in \mathrm{QR}_p$, then outputs 1
CASE 2: If $y_3 \in \mathrm{QNR}_p$, then outputs 0
End.

Except the values when $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p = 0$, the Legendre symbol of

$Y^{r_1} \oplus (Y^{r_2})^2 \bmod p$ is

$$\left(\frac{Y^{r_1} \oplus (Y^{r_2})^2}{p}\right) = -1,$$

By the multiplicative property of Legendre symbol,

$$\left(\frac{y_3}{p}\right) = \left(\frac{x_b}{p}\right)\left(\frac{Y^{r_1} \oplus (Y^{r_2})^2}{p}\right),$$

we can determine x_b is $x_0 \in \text{QR}_p$ or $x_1 \in \text{QNR}_p$, according to the Legendre symbol $\left(\frac{y_3}{p}\right)$. This forms CASE 1 and CASE 2 of $\mathcal{A}_2''$, respectively. The advantage of $\mathcal{A}''$ for Π'' is $\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{A}'',\Pi''}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A}'',\Pi''}(k)|\neg\mathsf{Fail}] - 1 = 1$.

From the specification of $\mathcal{A}''$, it runs within at most $t'' \leq t_1 + t_{\text{QR}}$. Obviously, the both breaking algorithms $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2')$ and $\mathcal{A}'' = (\mathcal{A}_1'', \mathcal{A}_2'')$ are in a polynomial time in Theorem 2 and Theorem 3, respectively. □

We can see that no matter what the ElGamal-like cryptosystem employs $\mathcal{K}$ or $\widehat{\mathcal{K}}$, the scheme is insecure in the IND-CPA sense, even the cryptosystem will be failed to encrypt and/or decrypt. Though the probability of event Fail will decrease when we chose a large prime q or p (the security parameter k), for both cryptosystems $\Pi' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ and $\Pi'' = (\widehat{\mathcal{K}}, \mathcal{E}', \mathcal{D}')$, the values after exclusive-or operation may not in the group $\mathbb{G}_p$ and $\mathbb{G}_q$, respectively. This results in their scheme is insecure in the IND-CPA sense.

The ElGamal cryptosystem has been proven to be secure in the IND-CPA sense in the standard model if the operation is in QR_p [83]. The IND-CPA sense is considered as a basic requirement for most provably secure public public key cryptosystems. In many applications, plaintexts may information which can be guessed easily such as in a BUY/SELL instruction to a stock broker.

In this chapter, we precisely show that the ElGamal cryptosystem is insecure in the IND-CPA sense if the operation is in not QR_p. For the ElGamal-like cryptosystem, we give two simple examples to prove it is insecure in the IND-CPA sense either operated in QR_p or not (employ the key generation $\mathcal{K}$ or $\widehat{\mathcal{K}}$). Besides, the cryptosystem has the probability to be crashed when $Y^{r_1} \oplus (Y^{r_2})^2 \bmod p = 0$. Since the exclusive-or operation is not suitable for the group operation, the computed values cannot be expected in that group.

However, the motivation for encrypting large messages in public public key cryptosystem is practical, since they have bad compared to symmetric cryptosystems. Attempt to propose a public key cryptosystem for encrypting large messages and proven GOAL-ATK security in the RO or standard model is exhilaratingly.

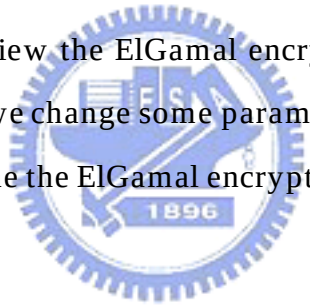
In the next chapter, an efficient conversion from the semantically secure ElGamal encryption scheme against chosen-plaintext attacks to a semantically secure extension of the ElGamal encryption scheme against adaptive chosen-ciphertext attacks in the random oracle model is presented. In the encryption algorithm of the converted scheme, only two random numbers are generated for each encryption. The result of the converted version of the ElGamal encryption scheme not only provides a higher security level but also is more efficient than the ElGamal encryption scheme when encrypting a large plaintext. An analyses of the modified encryption scheme is given to demonstrate its enhanced security.

Chapter 4

An ElGamal-Extension Cryptosystem

4.1 ElGamal-Extension Cryptosystem

Here, we again briefly review the ElGamal encryption scheme (this is the same as in Section 3.1 but we change some parameters for easy-to-read) and show that how to extend the the ElGamal encryption scheme for encrypting a large plaintext.



ElGamal Encryption Scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the ElGamal encryption scheme, which is secure in the IND-CPA sense [83].

- Key generation algorithm $\mathcal{K}$: $(pk, sk) \leftarrow \mathcal{K}(1^{k_0+2k_1+l})$, $pk = (p, q, g, Y)$ and $sk = (p, q, g, s)$, where $Y = g^s \bmod p$, $|p| = k = k_0 + 2k_1 + l$, $s \in \mathbb{Z}/q\mathbb{Z}$, $q|p-1$, and $\# < g >= q$.

- Encryption algorithm $\mathcal{E}$:

$$(y_1, y_3) = \mathcal{E}_{pk}(x, r) = (g^r \bmod p, x \cdot Y^r \bmod p),$$

where the plaintext $x \in \{0,1\}^k$ (the plaintext should be chosen from a subgroup [83], however, to simply the notation, we release this restriction.) and $r \in_R \mathbb{Z}_q$.

- Decryption algorithm $\mathcal{D}$:

$$x = \mathcal{D}_{sk}(y_1, y_3) = y_3 \cdot (y_1^s)^{-1} \bmod p.$$

If the plaintext $x > p$ ($|x| > k$), it should be divided into several pieces, says $x_1, \ldots, x_n$, where $x_i < p$ ($|x_i| < k$). For each x_i, the random number r should be chosen distinct in the encryption algorithm.

ElGamal-Extension Encryption Scheme $\overline{\Pi} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$

Let $\overline{\Pi} = (\overline{\mathcal{K}}, \overline{\mathcal{E}}, \overline{\mathcal{D}})$ be the ElGamal-Extension encryption scheme.

- Key generation algorithm $\overline{\mathcal{K}}$: $(pk, sk) \leftarrow \overline{\mathcal{K}}(1^k) = \mathcal{K}(1^k)$.

  The key generation algorithm $\overline{\mathcal{K}}$ is the same as in $\mathcal{K}$.

- Hash functions $\mathcal{H}$ and $\mathcal{J}$:

  $\mathcal{H} : \{0,1\}^{k_0+2k_1} \rightarrow \{0,1\}^l$, $\mathcal{J} : \{0,1\}^k \rightarrow \{0,1\}^k$.

- Encryption algorithm $\overline{\mathcal{E}}$:

  A large plaintext x is divided into $x_1, x_2, \cdots, x_n$ subtexts.

$$(y'_1, y'_2, y'_{3,i}) = \overline{\mathcal{E}}_{pk}(x_i, r_1, r_2),$$

1. Concatenate $X_i = x_i||r_1||r_2$, where $x_i \in \{0,1\}^{k_0}$, $r_1, r_2 \in_R \{0,1\}^{k_1} \in \mathbb{Z}_q$, and $||$ denotes concatenation.

2. Compute $J_i = \mathcal{J}(i \cdot Y^{r_2} \bmod p)$.

3. Compute $(y_1, y_3) = \mathcal{E}_{pk}(X_i||\mathcal{H}(X_i), r_1) = (g^{r_1} \bmod p, (X_i||\mathcal{H}(X_i)) \cdot Y^{r_1} \bmod p)$.

4. Compute $(y'_1, y'_2, y'_{3,i}) = (y_1, g^{r_2} \bmod p, y_3 \cdot J_i \bmod p)$.

- Decryption algorithm $\overline{\mathcal{D}}$:

$$x_i = \overline{\mathcal{D}}_{sk}(y'_1, y'_2, y'_{3,i}),$$

1. Compute $J_i = \mathcal{J}(i \cdot y'^{s}_2 \bmod p)$.

2. Compute $W_i = \mathcal{D}_{sk}(y'_1, y'_{3,i} \cdot J_i^{-1} \bmod p)$.

3. Output

$$\begin{cases} [W_i]^{k_0}, \text{ if } \mathcal{H}([W_i]^{k_0+2k_1}) = [W_i]_l \\ null, \text{ otherwise} \end{cases}$$

The notations of $[W_i]^a$ and $[W_i]_b$ denote the first a-bit and the last b-bit of W_i, respectively

Finally, the whole plaintext x can be concatenated as $x_1|| \cdots ||x_n$.

To understand what the ElGamal-Extension encryption scheme can achieve consider the following. The ElGamal encryption scheme is long and involved and there is an additional random value J_i for each x_i. Even if there are only two random numbers r_1 and r_2, the hash value J_i still makes the encryption scheme probabilistic. If the adversary can obtain the hash value $\mathcal{J}(i \cdot Y^{r_2} \bmod p)$, she is still faced with the of breaking the ElGamal encryption scheme, i.e. $D_{sk}(y'_1, y'_3 \cdot J_i^{-1} \bmod p) = W_i$. It already knows the ElGamal encryption scheme is IND-CPA secure [83] under the DDH assumption, in which the adversary cannot obtain any bit about the plaintext $W_i = x_i||r_1||r_2||\mathcal{H}(X_i)$.

Furthermore, to compute the hash value $\mathcal{J}(i \cdot Y^{r_2} \bmod p)$ with the knowledge of the public key Y and the value y_2' is equivalent to solve the Computational Diffie-Hellman (CDH) assumption in Definition 5, which is weaker than the DDH assumption in the same group [71]. If the DDH assumption is held in the group, then the CDH assumption must be held in that group. Therefore, the security of the proposed scheme can be solely based on the DDH assumption.

To reveal other plaintext x_j's, the adversary cannot compute J_j $(\forall j \neq i)$ under the assumption of hash function $\mathcal{J}(\cdot)$, since the values of J_i and J_j are nonlinearly related. To meet IND-CCA2, the plaintext x_i is protected under the hash function $\mathcal{H}(\cdot)$ to ensure the data integrity and has a data integrity validating step in the decryption algorithm. Without this validating step, the adversary could trivially generate ciphertext for which the corresponding plaintext is unknown. To do this, she just outputs the random strings. In the next section, we give the analyses of the reduction for proving its securities.

4.2 Security Analysis

In this section, our first goal is to show that the ElGamal-Extension encryption scheme is secure in the IND-CCA2 sense via Proposition 2. Theorem 4 and Theorem 5 shows that there is a plaintext extractor in the ElGamal-Extension encryption and is secure in the IND-CPA sense, respectively. Here, we only consider that the plaintext x is smaller than p. The sequence number i of x_i presented in the ElGamal-Extension encryption scheme is omit-

ted. The sequence number is involved in the ElGamal-Extension encryption scheme to show how the security notation IND-CPA$_{\mathsf{PAIR}}$ is achieved when using only two random numbers.

Theorem 4 (Plaintext extractor $\mathcal{PE}$ of $\overline{\Pi}$). *If there exists a $(t, q_{\mathcal{H}}, q_{\mathcal{J}})$-adversary $\mathcal{B}$, then there exists a constant c and a $(t', \lambda(k))$-plaintext extractor $\mathcal{PE}$ such that*

$$t' = t + q_{\mathcal{J}} q_{\mathcal{H}}(t_{\mathcal{E}_{pk}} + c) \; and \; \lambda(k) = 1 - (q_{\mathcal{J}} \cdot 2^{-k} + |H_s| \cdot 2^{-l}).$$

$t_{\mathcal{E}_{pk}}$ denotes the computational running time of the encryption algorithm $\mathcal{E}_{pk}$ and $|H_s|$ denotes the number of pairs (h, H_v) in the set H_s such that $(y'_1, y'_3 \cdot (Y^{[[h_v]_{2k_1}]_{k_1}})^{-1} \bmod p) = \mathcal{E}_{pk}(h||H_v, [[h]_{2k_1}]^{k_1})$ in the following specification of $\mathcal{PE}$.

Proof. We construct a plaintext extractor $\mathcal{PE}$ as follows:

Extractor: $\mathcal{PE}(\Lambda_{\mathcal{H}}, \Lambda_{\mathcal{J}}, C, (y'_1, y'_2, y'_3), pk)$
For $u = 1, \cdots, q_{\mathcal{J}}$ do
For $v = 1, \cdots, q_{\mathcal{H}}$ do
$(y_1, y_3) \leftarrow (y'_1, y'_3 \cdot J_u^{-1} \bmod p)$
If $(y_1, y_3) == \mathcal{E}_{pk}(h_v||H_v, [[h_v]_{2k_1}]^{k_1})$
If $j_u == Y^{[[h_v]_{2k_1}]_{k_1}} \bmod p$
then $x \leftarrow [h_v]^{k_0}$ and break
Else $x \leftarrow null$
Return x
End.

Let c be the computation time of comparing two strings is equal or not, and some overhead. From the specification of $\mathcal{PE}$, it runs within $t + q_{\mathcal{J}} q_{\mathcal{H}}(t_{\mathcal{E}_{pk}} + c)$.

Since there exists an additional random oracle $\mathcal{J}(\cdot)$, $\Lambda_{\mathcal{J}} = \{(j_1, J_1), \ldots, (j_{q_{\mathcal{J}}}, J_{q_{\mathcal{J}}}))\}$ denotes the set of all $\mathcal{B}$'s queries and the corresponding answers of $\mathcal{J}(\cdot)$. Intuitionally, the plaintext x together with the random numbers r_1, r_2 are inputs to the random oracle $\mathcal{H}(\cdot)$. Moreover, all the answers

to queries should be obtained by the random oracles in the random oracle model. Furthermore, those queries and the corresponding answers are recorded in the lists $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{J}}$. Any generation of valid ciphertext should be obtained via that step. Hence, upon input of the valid ciphertext, $\mathcal{PE}$ can find out the corresponding plaintext by watching the lists $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{J}}$.

Now the probability that $\mathcal{PE}$ correctly outputs the plaintext x, that is $x = \overline{\mathcal{D}}_{sk}(y'_1, y'_2, y'_3)$. Consider the following events.

Con1$\wedge$Con2: the product of events Con1 and Con2, which is assigned to be true if there exists (j, J) in the list $\Lambda_{\mathcal{J}}$ and (h, H) in the list $\Lambda_{\mathcal{H}}$ such that the conditions $(y_1, y_3) == \mathcal{E}_{pk}(h_v||H_v, [[h_v]_{2k_1}]^{k_1})$ and $j_u == Y^{[[h_v]_{2k_1}]_{k_1}} \bmod p$ in the specification of $\mathcal{PE}$ hold. Two conditions are separately denoted as Con1 and Con2.

Fail: an event assigned to be true if $x \neq \overline{\mathcal{D}}_{sk}(y'_1, y'_2, y'_3)$.

We now bound the failure probability as follows:

$$\begin{aligned}
\Pr[\mathsf{Fail}] &= \Pr[\mathsf{Fail}|\mathsf{Con1} \wedge \mathsf{Con2}] \cdot \Pr[\mathsf{Con1} \wedge \mathsf{Con2}] + \\
&\quad \Pr[\mathsf{Fail}|\mathsf{Con1} \wedge \neg\mathsf{Con2}] \cdot \Pr[\mathsf{Con1} \wedge \neg\mathsf{Con2}] + \\
&\quad \Pr[\mathsf{Fail}|\neg\mathsf{Con1}] \cdot \Pr[\neg\mathsf{Con1}] \\
&\leq \Pr[\mathsf{Fail}|\mathsf{Con1} \wedge \mathsf{Con2}] + \Pr[\mathsf{Con1} \wedge \neg\mathsf{Con2}] \\
&\quad \Pr[\mathsf{Fail}|\neg\mathsf{Con1}]
\end{aligned}$$

In the following, we upper bound $\Pr[\mathsf{Fail}|\mathsf{Con1} \wedge \mathsf{Con2}]$, $\Pr[\mathsf{Con1} \wedge \neg\mathsf{Con2}]$, and $\Pr[\mathsf{Fail}|\neg\mathsf{Con1}]$, respectively.

The specification of $\mathcal{PE}$ is as follows. If Con1$\wedge$Con2 is true then $\mathcal{PE}$ never fails to guess the plaintext x and hence $\Pr[\mathsf{Fail}|\mathsf{Con1} \wedge \mathsf{Con2}] = 0$.

We further upper bound $\Pr[\mathsf{Con1} \wedge \neg\mathsf{Con2}]$ as follows:

$$\Pr[\mathsf{Con1} \wedge \neg\mathsf{Con2}] \leq \Pr[\mathsf{Con1}|\neg\mathsf{Con2}]$$

When $\neg$Con2 is true, there is a J_u in the list $\Lambda_{\mathcal{J}}$ such that $(y'_1, y'_3 \cdot J_u^{-1} \bmod p) == \mathcal{E}_{pk}(h_v||H_v, [[h_v]_{2k_1}]^{k_1})$. Under the random oracle model assumption in Definition 8, the probability of such J_u is 2^{-k}. The conditional probability $\Pr[\mathsf{Con1}|\neg\mathsf{Con2}]$ is $q_{\mathcal{J}} \cdot 2^{-k}$.

For $\Pr[\mathsf{Fail}|\neg\mathsf{Con1}]$, $\neg$Con1 is true and $\mathcal{PE}$ outputs $null$. That is, it guesses (y'_1, y'_2, y'_3) is a *invalid* ciphertext. Therefore, Fail is true implies $\mathcal{B}$ outputs the *valid* ciphertext (y'_1, y'_2, y'_3). For a fixed (y'_1, y'_2, y'_3) and $J = \mathcal{J}(Y^{[[h_v]_{2k_1}]_{k_1}} \bmod p)$, let H_s be the set of (h, H_v) such that $(y'_1, y'_3 \cdot J^{-1} \bmod p) = \mathcal{E}_{pk}(h||H_v, [[h]_{2k_1}]^{k_1})$. Then since $(y'_1, y'_2, y'_3) \notin C = \{(y'_1, y'_2, y'_3)_1, \cdots, (y'_1, y'_2, y'_3)_{q_E}\}$ and hence $\mathcal{D}_{sk}((y'_1, y'_3 \cdot J^{-1} \bmod p)_i) \neq h||\mathcal{H}(h)$ for every $(y'_1, y'_2, y'_3)_i \in C$. For a fixed (y'_1, y'_2, y'_3) and a fixed h, since $\mathcal{B}$ doesn't ask query h to oracle $\mathcal{H}(\cdot)$,

$$\Pr[\mathsf{Fail}|\neg\mathsf{Con1}] = \Pr_{\mathcal{H}\leftarrow\Omega}[\mathcal{H}(h) \in H_s] = |H_s| \cdot 2^{-l},$$

where $|H_s|$ denotes the number of pairs in H_s. Obviously, $|H_s|$ is small.

We conclude that $\Pr[\mathsf{Fail}] \leq q_{\mathcal{J}} \cdot 2^{-k} + |H_s| \cdot 2^{-l}$. Hence, $\lambda(k) = 1 - \Pr[\mathsf{Fail}] = 1 - (q_{\mathcal{J}} \cdot 2^{-k} + |H_s| \cdot 2^{-l})$. □

Theorem 5 ($\overline{\Pi}$: IND-CPA). *If there exists a $(t, q_{\mathcal{H}}, q_{\mathcal{J}}, \epsilon)$-breaker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for $\overline{\Pi}$ in the* IND-CPA *sense in the random oracle model, then there exists a constants c and a $(t', 0, 0, \epsilon')$-breaker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for Π in the* IND-CPA *sense in the standard model, where*

$$t' = t + q_{\mathcal{H}} \cdot c + q_{\mathcal{J}} \cdot c \text{ and } \epsilon' = \epsilon - q_{\mathcal{H}} \cdot 2^{-(2k_1-2)}.$$

Proof. We construct a breaking algorithm $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ in the IND-CPA and standard model setting by using $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as an oracle.

Firstly, $\mathcal{A}'$ initiates two lists $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{J}}$, to empty. Basically, when $\mathcal{A}$ asks query h and j, $\mathcal{A}'$ simulates two random oracles $\mathcal{H}(\cdot)$ and $\mathcal{J}(\cdot)$ as follows: If h has not been asked in the list $\Lambda_{\mathcal{H}}$, $\mathcal{A}'$ provides a random string H of length l-bit, and adds an entry (h, H) to the list $\Lambda_{\mathcal{H}}$. Similarly, if j has not been asked in the list $\Lambda_{\mathcal{J}}$, $\mathcal{A}'$ provides a random string J of length k-bit, and adds an entry (j, J) to the list $\Lambda_{\mathcal{J}}$. When $\mathcal{A}_1$ halts and outputs $(x_0, x_1, state)$, $\mathcal{A}'_1$ outputs $(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state)$ where γ_0, γ_1 are $(2k_1)$-bit random strings and β_0, β_1 are l-bit random strings.

Adversary: $\mathcal{A}'_1(pk)$
$\Lambda_{\mathcal{H}}$, $\Lambda_{\mathcal{J}}$←empty
Run $\mathcal{A}_1(pk)$
  Do while $\mathcal{A}_1$ does not make $\mathcal{H}$-query h and $\mathcal{J}$-query j
    If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
      If $j \notin \Lambda_{\mathcal{J}}$
        $J \leftarrow_R \{0,1\}^k$
        Put (j, J) on $\Lambda_{\mathcal{J}}$
        Answer J to $\mathcal{A}_1$
      Else $j \in \Lambda_{\mathcal{J}}$
        Answer J to $\mathcal{A}_1$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
    Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
      If $h \notin \Lambda_{\mathcal{H}}$
        $H \leftarrow_R \{0,1\}^l$
        Put (h, H) on $\Lambda_{\mathcal{H}}$
        Answer H to $\mathcal{A}_1$
      Else $h \in \Lambda_{\mathcal{H}}$
        Answer H to $\mathcal{A}_1$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
  $\mathcal{A}_1$ outputs $(x_0, x_1, state)$
$\gamma_0, \gamma_1 \leftarrow_R \{0,1\}^{2k_1}$
$\beta_0, \beta_1 \leftarrow_R \{0,1\}^l$
Return $(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state)$
End.

Then, outside of $\mathcal{A}'$, the ciphertext $(y_1, y_3) = \mathcal{E}_{pk}(x_b||\gamma_b||\beta_b, R)$ is computed by the encryption oracle $\mathcal{O}_{EN}$, where $b \in \{0,1\}$ is a random bit and $R \in \mathbb{Z}_q$ is a random string. Finally, $(x_0, x_1, state, (y_1, y_3))$ is input to $\mathcal{A}_2$.

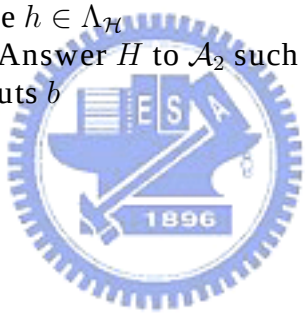
Encryption oracle: $\mathcal{O}_{EN}(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, pk)$
$R \leftarrow_R \mathbb{Z}_q$
$b \leftarrow_R \{0,1\}$
$(y_1, y_3) \leftarrow \mathcal{E}_{pk}(x_b||\gamma_b||\beta_b, R)$
Return (y_1, y_3)

End.

$\mathcal{A}'_2$ chooses a random string $r_2 \in \mathbb{Z}_q$ and k-bit random string J^*. Then it sets $y'_1 = y_1$, $y'_2 = g^{r_2} \bmod p$, and $y'_3 = y_3 \cdot J^* \bmod p$. Note that (y'_1, y'_2, y'_3) is treated as the ciphertext of x_b.

Adversary: $\mathcal{A}_2'(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state, (y_1, y_3))$
$r_2 \leftarrow_R \mathbb{Z}_q$; $J^* \leftarrow_R \{0,1\}^k$
$y_1' \leftarrow y_1$; $y_2' \leftarrow g^{r_2} \bmod p$; $y_3' \leftarrow y_3 \cdot J^* \bmod p$
Run $\mathcal{A}_2(x_0, x_1, state, (y_1', y_2', y_3'))$
Do while $\mathcal{A}_2$ does not make $\mathcal{H}$-query h and $\mathcal{J}$-query j
Ask$j$ $\leftarrow$false
If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
If $j = Y^{r_2} \bmod p$
Answer J^* to $\mathcal{A}_2$
Put (j, J^*) on $\Lambda_{\mathcal{J}}$
Ask$j$ $\leftarrow$true
Else if $j \notin \Lambda_{\mathcal{J}}$
$J \leftarrow_R \{0,1\}^k$
Answer J to $\mathcal{A}_2$
Else $j \in \Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_2$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
If Ask$j$ =true and $h = x_b||\gamma_b$
Stop $\mathcal{A}_2$ and output b
Else if $h \notin \Lambda_{\mathcal{H}}$
$H \leftarrow_R \{0,1\}^l$
Put (h, H) on $\Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$
Else $h \in \Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
$\mathcal{A}_2$ outputs b
Return b

End.



The argument behind the proof is as follows: When $\mathcal{A}_2$ asks the query $j = Y^{r_2} \bmod p$, $\mathcal{A}_2'$ answers J^* and Ask$j$ is set be true. Since the random string r_2 is chosen by $\mathcal{A}_2'$, it has the ability to check whether the query j is equal to $Y^{r_2} \bmod p$ or not. Once Ask$j$ is true and $\mathcal{A}_2$ asks a query $h = x_b||\gamma_b$, it is almost equivalent to $\mathcal{D}_{sk}(y_1, y_3) = \mathcal{D}_{sk}(y_1', y_3' \cdot (J^*)^{-1} \bmod p)$, since $\mathcal{A}_2$ has no clue to $\gamma_{\bar{b}}$ where $\bar{b}$ is the complement of bit b. The probability to ask $h = x_{\bar{b}}||\gamma_{\bar{b}}$ is $2^{-(2k_1)}$ which is negligible. Under the condition Ask$j$=true, $\mathcal{A}_2'$ can expect that it will output a correct bit b if $\mathcal{A}_2$ asks either $h = x_0||\gamma_0$ or $h = x_1||\gamma_1$. If $\mathcal{A}_2$ asks neither of them, $\mathcal{A}_2'$ can expect that $\mathcal{A}_2$ cannot distinguish (y_1', y_2', y_3')

from a correct ciphertext.

To analyze the success probability of $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$, we recall the definitions of success probabilities of $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in Definition 9. Consider the follows events to capture the success probabilities of $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$

Ask$j$: is true if a $\mathcal{J}$-query $j = Y^{r_2} \bmod p$ was made by $\mathcal{A}_2$.

Ask$b$: is true if a $\mathcal{H}$-query $h = x_b||\gamma_b$ was made by $\mathcal{A}_2$.

Ask$\bar{b}$: is true if a $\mathcal{H}$-query $h = x_{\bar{b}}||\gamma_{\bar{b}}$ was made by $\mathcal{A}_2$.

The probability of $\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)$ can be obtained by considering the conditions of the product of events Ask$j \wedge$Ask$b$ and its complement. Then,

$$\begin{aligned}\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)] &= \Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)|\mathsf{Ask}j \wedge \mathsf{Ask}b] \cdot \Pr[\mathsf{Ask}j \wedge \mathsf{Ask}b] + \\ &\quad \Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)|\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b] \cdot \Pr[\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b].\end{aligned}$$

The probability of $\neg$Ask$j \vee \neg$Ask$b$ can be written as,

$$\Pr[\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b] = \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] + \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}].$$

Then,

$$\begin{aligned}&\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)] = \Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)|\mathsf{Ask}j \wedge \mathsf{Ask}b] \cdot \Pr[\mathsf{Ask}j \wedge \mathsf{Ask}b] + \\ &\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] \cdot \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] + \\ &\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A}',\Pi}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] \cdot \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] \quad (4.1)\end{aligned}$$

Similarly,

$$\begin{aligned}&\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A},\overline{\Pi}}(k)] = \Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A},\overline{\Pi}}(k)|\mathsf{Ask}j \wedge \mathsf{Ask}b] \cdot \Pr[\mathsf{Ask}j \wedge \mathsf{Ask}b] + \\ &\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A},\overline{\Pi}}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] \cdot \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] + \\ &\Pr[\mathsf{Succ}^{\text{IND-CPA}}_{\mathcal{A},\overline{\Pi}}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] \cdot \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] \quad (4.2)\end{aligned}$$

From the specification of $\mathcal{A}'$, we have the following equations,

$$\begin{cases} \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\overline{\Pi}}(k)|\mathsf{Ask}j \wedge \mathsf{Ask}b] = 1, \\ \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\overline{\Pi}}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] = 0 \\ \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\overline{\Pi}}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] = \\ \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \neg\mathsf{Ask}\bar{b}] \end{cases}$$

Equation (4.1) and Equation (4.2) are computed as follows.

$$\begin{aligned} & \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)] - \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\overline{\Pi}}(k)] \\ = \; & (1 - \Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A},\overline{\Pi}}(k)|\mathsf{Ask}j \wedge \mathsf{Ask}b]) \cdot \Pr[\mathsf{Ask}j \wedge \mathsf{Ask}b] - \\ & \Pr[\mathsf{Succ}^{\mathsf{CPA}}_{\mathcal{A},\overline{\Pi}}(k)|(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] \cdot \Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] \\ \geq \; & -\Pr[(\neg\mathsf{Ask}j \vee \neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b}] \\ = \; & -\Pr[(\neg\mathsf{Ask}j \wedge \mathsf{Ask}\bar{b}) \vee (\neg\mathsf{Ask}b) \wedge \mathsf{Ask}\bar{b})] \\ \geq \; & -(\Pr[\neg\mathsf{Ask}j \wedge \mathsf{Ask}\bar{b}] + \Pr[\neg\mathsf{Ask}b \wedge \mathsf{Ask}\bar{b}]). \end{aligned}$$

Since $\gamma_{\bar{b}}$ is a uniform random string over $\{0,1\}^{2k_1}$, we have $\Pr[\neg\mathsf{Ask}j \wedge \mathsf{Ask}\bar{b}] \leq q_{\mathcal{H}} \cdot 2^{-2k_1}$ and $\Pr[\neg\mathsf{Ask}b \wedge \mathsf{Ask}\bar{b}] \leq q_{\mathcal{H}} \cdot 2^{-2k_1}$. Thus,

$$\begin{aligned} \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)] &\geq \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A},\overline{\Pi}}(k)] - (\Pr[\neg\mathsf{Ask}j \wedge \mathsf{Ask}\bar{b}] + \Pr[\neg\mathsf{Ask}b \wedge \mathsf{Ask}\bar{b}]) \\ &\geq \frac{\epsilon+1}{2} - \frac{q_{\mathcal{H}}}{2^{2k_1-1}}. \end{aligned} \tag{4.3}$$

and we obtain that $\epsilon' = \epsilon - q_{\mathcal{H}} \cdot 2^{-(2k_1-2)}$.

The running time of $\mathcal{A}'$ is at most time $t + q_{\mathcal{H}} \cdot c + q_{\mathcal{J}} \cdot c$. □

Theorem 4 and Theorem 5 show that the encryption scheme is secure in the PA sense. Intuitively, $\mathcal{PE}$ can simulate the decryption oracle in the IND-CCA2 sense with an overwhelming probability. Via Proposition 1, we can prove $\overline{\Pi}$ is secure in the IND-CCA2 sense in Theorem 6.

Theorem 6 ($\overline{\Pi}$: IND-CCA2). *If there exists a $(t, q_{\mathcal{H}}, q_{\mathcal{J}}, q_D, \epsilon)$-breaker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for $\overline{\Pi}$ in the sense of* IND-CCA2 *in the random oracle model, then there exist a constant c and a $(t', 0, 0, 0, \epsilon')$-breaker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for Π in the sense of* IND-CPA *in the standard model where*

$$t' = t + q_{\mathcal{H}} q_{\mathcal{J}} (T_{\mathcal{E}_{pk}} + c) + q_{\mathcal{H}} c + q_{\mathcal{J}} c \; and \; \epsilon' = (\epsilon - q_{\mathcal{H}} \cdot 2^{-(2k_1 - 2)}) \cdot \lambda(k)^{q_D}.$$

Proof. From the result of Theorem 6, it is found out that the encryption scheme $\overline{\Pi}$ is secure in the IND-CCA2. The proof is omitted since it is clear from the following specification of adversary $\mathcal{A}'$ combined with the proofs in Theorem 4 and Theorem 5.

Adversary: $\mathcal{A}_1'(pk)$
$\Lambda_{\mathcal{H}}, \Lambda_{\mathcal{J}} \leftarrow$empty
Run $\mathcal{A}_1^{\mathcal{D}_{sk},\mathcal{H},\mathcal{J}}(pk)$
Do while $\mathcal{A}_1$ does not make $\mathcal{H}$-query h, $\mathcal{J}$-query j, D-query $(y_1, y_2, y_3)'$
If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
If $j \notin \Lambda_{\mathcal{J}}$
$j \leftarrow_R \{0,1\}^k$
Put (j, J) on $\Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_1$
Else $j \in \Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_1$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
If $h \notin \Lambda_{\mathcal{H}}$
$H \leftarrow_R \{0,1\}^l$
Put (h, H) on $\Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_1$
Else $h \in \Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_1$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
Else if $\mathcal{A}_1$ makes D-query $(y_1, y_2, y_3)'$
Run $\mathcal{PE}(\Lambda_{\mathcal{H}}, \Lambda_{\mathcal{J}}, C, (y_1, y_2, y_3)', pk)$
$\mathcal{PE}$ outputs x'
Answer x' to $\mathcal{A}_1$
$\mathcal{A}_1$ outputs $(x_0, x_1, state)$
$\gamma_0, \gamma_1 \leftarrow_R \{0,1\}^{2k_1}$
$\beta_0, \beta_1 \leftarrow_R \{0,1\}^l$
Return $(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state)$

End.

Encryption oracle: $\mathcal{O}_{EN}(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, pk)$
$b \leftarrow_R \{0,1\}$
$(y_1, y_3) \leftarrow \mathcal{E}_{pk}(x_b||\gamma_b||\beta_b, R)$
Return (y_1, y_3)

End.

Adversary: $\mathcal{A}_2'(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state, (y_1, y_3))$
$r_2 \leftarrow_R \mathbb{Z}_q$; $J^* \leftarrow_R \{0,1\}^k$
$y_1' \leftarrow y_1$; $y_2' \leftarrow g^{r_2} \bmod p$; $y_3' \leftarrow y_3 \cdot J^* \bmod p$
Run $\mathcal{A}_2^{\mathcal{D}_{sk},\mathcal{H},\mathcal{J}}(x_0, x_1, state, (y_1', y_2', y_3'))$
$C \leftarrow (y_1', y_2', y_3')$
Do while $\mathcal{A}_2$ does not make $\mathcal{H}$-query h and $\mathcal{J}$-query j
D-query $(y_1, y_2, y_3)'$
Ask$j \leftarrow$false
If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
If $j = Y^{r_2} \bmod p$
Answer J^* to $\mathcal{A}_2$
Put (j, J^*) on $\Lambda_{\mathcal{J}}$
Ask$j \leftarrow$true
Else if $j \notin \Lambda_{\mathcal{J}}$
$J \leftarrow_R \{0,1\}^k$
Answer J to $\mathcal{A}_2$
Else $j \in \Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_2$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
If Ask$j$ =true and $h = x_b||\gamma_b$
Stop $\mathcal{A}_2$ and output b
Else if $h \notin \Lambda_{\mathcal{H}}$
$H \leftarrow_R \{0,1\}^l$
Put (h, H) on $\Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$
Else $h \in \Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
Else if $\mathcal{A}_1$ makes D-query $(y_1, y_2, y_3)'$
Run $\mathcal{PE}(\Lambda_{\mathcal{H}}, \Lambda_{\mathcal{J}}, C, (y_1, y_2, y_3)', pk)$
$\mathcal{PE}$ outputs x'
Answer x' to $\mathcal{A}_1$
$\mathcal{A}_2$ outputs b
Return b

End.

□

Now, we have to consider whether the ElGamal-Extension encryption scheme $\overline{\Pi}$ is secure when using only two random numbers r_1, r_2 for each piece x_i. We first show that when using only one random number r in the ElGamal encryption scheme Π, what the advantage of the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is in the IND-CPA$_{\text{PAIR}}$ sense in Definition 12.

Basically, $\mathcal{A}_1$ with the input pk arbitrarily outputs three plaintexts x^*, x_0, x_1 with the same length $|x^*| = |x_0| = |x_1|$.

Adversary: $\mathcal{A}_1(pk)$
Return $(x^*, x_0, x_1, state)$
End.

Then, the ciphertexts $(y_1^*, y_3^*) = \mathcal{E}_{pk}(x^*, r)$ and $(y_1, y_3) = \mathcal{E}_{pk}(x_b, r)$ computed by the encryption oracle $\mathcal{O}_{EN}$, where $b \in \{0, 1\}$ is a random bit and $r \in \mathbb{Z}_q$ is a random string.

Encryption oracle: $\mathcal{O}_{EN}(x^*, x_0, x_1, pk)$
$r \leftarrow_R \mathbb{Z}_q$
$(y_1^*, y_3^*) \leftarrow \mathcal{E}_{pk}(x^*, r)$
$b \leftarrow_R \{0, 1\}$
$(y_1, y_3) \leftarrow \mathcal{E}_{pk}(x_b, r)$
Return $(x^* \rightleftarrows (y_1^*, y_3^*), (y_1, y_3))$
End.

Finally, $(x_0, x_1, state, x^* \rightleftarrows (y_1^*, y_3^*), (y_1, y_3))$ are inputted to $\mathcal{A}_2$. The aim of $\mathcal{A}_2$ is to output the correct b.

Adversary: $\mathcal{A}_2(x_0, x_1, state, x^* \rightleftarrows (y_1^*, y_3^*), (y_1, y_3))$
$Y^r \leftarrow y_3^* \cdot (x^*)^{-1} \bmod p$
If $y_3 \cdot (Y^r)^{-1} \bmod p == x_0$
  Return 0
Else
  Return 1
End.

From the specification of $\mathcal{A}_2$, since $y_3^* = x^* \cdot Y^r \bmod p$ and $x^* \rightleftarrows (y_1^*, y_3^*)$, the value of Y^r is easily revealed by computing $y_3^*(x^*)^{-1} \bmod p$. Then, since

$y_3 = x_b \cdot Y^r \bmod p$, $\mathcal{A}_2$ can determine if $y_3 \cdot (Y^r)^{-1} \bmod p$ is equal to x_0. If it is then outputs 0 otherwise output 1. Thus, $\mathcal{A}_2$ always correctly outputs b, that is, $\Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k)] = 1$ and $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k) = 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}}_{\mathcal{A},\Pi}(k)] - 1 = 1$.

The pair of plaintext-ciphertext $x^* \rightleftarrows (y_1^*, y_3^*)$ is the "cryptanalysis training" for the adversary. Here, the reason for giving the pair $x^* \rightleftarrows (y_1^*, y_3^*)$ from $\mathcal{O}_{EN}$ not generated by the adversary herself is that the adversary cannot generate the pair $x^* \rightleftarrows (y_1^*, y_3^*)$ using the same random number r. Hence, the training is provided by $\mathcal{O}_{EN}$.

Obviously, if the encryption oracle chooses a different r, this training does not give any help to the adversary and the scheme is secure in the $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$ sense. In the following theorem, we show only two random numbers r_1, r_2 in the ElGamal-Extension encryption scheme is secure in the $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$ sense.

Theorem 7 ($\overline{\Pi}$: $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$). *If there exists a $(t, q_{\mathcal{H}}, q_{\mathcal{J}}, \epsilon)$-breaker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for $\overline{\Pi}$ in the $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$ sense in the random oracle model and the probability $\Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]$ is non-negligible, then there exists a constant c and a $(t', 0, 0, \epsilon')$-breaker $\mathcal{A}' = (\mathcal{A}'_1, \mathcal{A}'_2)$ for Π in the $\mathsf{IND\text{-}CPA}_{\mathsf{PAIR}}$ sense in the standard model, where*

$$t' = t + q_{\mathcal{H}} \cdot c + q_{\mathcal{J}} \cdot c \text{ and } \epsilon' = (\epsilon - \frac{q_{\mathcal{H}}}{2^{2k_1-2}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] - \Pr[\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)].$$

Proof. The event $\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)$ will be defined later. Basically, $\mathcal{A}'_1$ calls $\mathcal{A}_1$ as a subroutine. The answers for $\mathcal{H}$-query and $\mathcal{J}$-query are the same as in Theorem 5. When $\mathcal{A}_1$ halts and outputs $(x^*, x_0, x_1, state)$, $\mathcal{A}'_1$ outputs $(x^*||\gamma^*||\beta^*, x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state)$ where γ^*, γ_0, γ_1 are $(2k_1)$-bit random

strings and β^*, β_0, β_1 are l-bit random strings.

Adversary: $\mathcal{A}_1'(pk)$
$\Lambda_{\mathcal{H}}$, $\Lambda_{\mathcal{J}}\leftarrow$empty
Run $\mathcal{A}_1(pk)$
Do while $\mathcal{A}_1$ does not make $\mathcal{H}$-query h and $\mathcal{J}$-query j
If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
If $j \notin \Lambda_{\mathcal{J}}$
$j \leftarrow_R \{0,1\}^k$
Put (j, J) on $\Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_1$
Else $j \in \Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_1$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
If $h \notin \Lambda_{\mathcal{H}}$
$H \leftarrow_R \{0,1\}^l$
Put (h, H) on $\Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_1$
Else $h \in \Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_1$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
$\mathcal{A}_1$ outputs $(x^*, x_0, x_1, state)$
$\gamma^*, \gamma_0, \gamma_1 \leftarrow_R \{0,1\}^{2k_1}$
$\beta^*, \beta_0, \beta_1 \leftarrow_R \{0,1\}^l$
Return $(x^*||\gamma^*||\beta^*, x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, state)$
End.

The ciphertexts $(y_1^*, y_3^*) = \mathcal{E}_{pk}(x^*||\gamma^*||\beta^*, R^*)$ and $(y_1, y_3) = \mathcal{E}_{pk}(x_b||\gamma_b||\beta_b, R_1)$ are computed by the encryption oracle $\mathcal{O}_{EN}$, where $b \in \{0,1\}$ is a random bit and $R^*, R_1 \in \mathbb{Z}_q$ are random strings.

Encryption oracle: $\mathcal{O}_{EN}(x^*||\gamma^*||\beta^*, x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, pk)$
$R^*, R_1 \leftarrow_R \mathbb{Z}_q$
$(y_1^*, y_3^*) \leftarrow \mathcal{E}_{pk}(x^*||\gamma^*||\beta^*, R^*)$
$b \leftarrow_R \{0,1\}$
$(y_1, y_3) \leftarrow \mathcal{E}_{pk}(x_b||\gamma_b||\beta_b, R_1)$
Return $(x^* \rightleftarrows (y_1^*, y_3^*), (y_1, y_3))$
End.

The algorithm $\mathcal{A}_2'$ is similar to Theorem 5. The difference is that it sets $y_1^* = y_1'$, $y_2^* = y_2' = g^{r_2}$, and $y_3^* = \alpha$, where α is a k-bit random string. Note that $x^* \rightleftarrows (y_1^*, y_2^*, y_3^*)$ is treated as a pair of plaintext-ciphertext.

Adversary: $\mathcal{A}_2'(x_0||\gamma_0||\beta_0, x_1||\gamma_1||\beta_1, x^*||\gamma^*||\beta^* \rightleftarrows (y_1^*, y_3^*), state, (y_1, y_3))$
$r_2 \leftarrow_R \mathbb{Z}_q$; $J^* \leftarrow_R \{0,1\}^k$
$y_1^*, y_1' \leftarrow y_1$; $y_2^*, y_2' \leftarrow g^{r_2} \bmod p$; $y_3' \leftarrow y_3 \cdot J^* \bmod p$
$\alpha \leftarrow_R \{0,1\}^k$; $y_3^* \leftarrow \alpha$
Run $\mathcal{A}_2(x_0, x_1, x^* \rightleftarrows (y_1^*, y_2^*, y_3^*), state, (y_1', y_2', y_3'))$
Do while $\mathcal{A}_2$ does not make $\mathcal{H}$-query h and $\mathcal{J}$-query j
Ask$j \leftarrow$false
If $\mathcal{A}_1$ makes $\mathcal{J}$-query j
If $j = 2 \cdot Y^{r_2} \bmod p$
Answer J^* to $\mathcal{A}_2$
Put (j, J^*) on $\Lambda_{\mathcal{J}}$
Ask$j \leftarrow$true
Else if $j \notin \Lambda_{\mathcal{J}}$
$J \leftarrow_R \{0,1\}^k$
Answer J to $\mathcal{A}_2$
Else $j \in \Lambda_{\mathcal{J}}$
Answer J to $\mathcal{A}_2$ such that $(j, J) \in \Lambda_{\mathcal{J}}$
Else if $\mathcal{A}_1$ makes $\mathcal{H}$-query h
If Ask$j$ =true and $h = x_b||\gamma_b$
Stop $\mathcal{A}_2$ and output b
Else if $h \notin \Lambda_{\mathcal{H}}$
$H \leftarrow_R \{0,1\}^l$
Put (h, H) on $\Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$
Else $h \in \Lambda_{\mathcal{H}}$
Answer H to $\mathcal{A}_2$ such that $(h, H) \in \Lambda_{\mathcal{H}}$
$\mathcal{A}_2$ outputs b
Return b

End.

From the specification of $\mathcal{A}_2'$, it simulates the encryption oracle $\mathcal{O}_{EN}$ to generate a pair $x^* \rightleftarrows (y_1^*, y_2^*, y_3^*)$ for the cryptanalysis training. If $\mathcal{A}_2$ detects that pair is not valid, the simulation fails. Thus, $\mathcal{A}_2'$ cannot make full use of the $\mathcal{A}_2$'s ability to get non-negligible advantage. Let $\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)$ be the

event that $\mathcal{A}_2$ detects that pair is not valid. We can rewrite the probability of $\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)$ in the inequality (4.3) as the conditional probability of $\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)$ given $\neg\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)$.

$$\Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)|\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] \geq \frac{\epsilon+1}{2} - \frac{q_{\mathcal{H}}}{2^{2k_1-1}}. \tag{4.4}$$

By the law of total probability,

$$\begin{aligned}\Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)] &= \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)|\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] \cdot \Pr[\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] + \\ &\quad \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)|\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)].\end{aligned}$$

From the specification of $\mathcal{A}'$, we know that

$$\Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)|\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] = 0 \tag{4.5}$$

Via Inequality (4.4) and Equation (4.5), we obtain

$$\Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)] \geq (\frac{\epsilon+1}{2} - \frac{q_{\mathcal{H}}}{2^{2k_1-1}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)].$$

To calculate the advantage of $\mathcal{A}'$,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k) &= 2 \cdot \Pr[\mathsf{Succ}^{\mathsf{IND\text{-}CPA}}_{\mathcal{A}',\Pi}(k)] - 1 \\ &\geq 2 \cdot \left((\frac{\epsilon+1}{2} - \frac{q_{\mathcal{H}}}{2^{2k_1-1}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] \right) - 1 \\ &= (\epsilon + 1 - \frac{q_{\mathcal{H}}}{2^{2k_1-2}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] - 1 \\ &= (\epsilon - \frac{q_{\mathcal{H}}}{2^{2k_1-2}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] - (1 - \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]) \\ &= (\epsilon - \frac{q_{\mathcal{H}}}{2^{2k_1-2}}) \cdot \Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)] - \Pr[\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]\end{aligned}$$

Finally, it should be determined if the probability $\Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]$ is negligible or not. Obviously, if $\Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]$ is non-negligible, then the proof is concluded. For the running time of $\mathcal{A}'$, it is similar to that in Theorem 5. To show that $\Pr[\neg\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\overline{\Pi}}(k)]$ is non-negligible consider the following. □

We claim that the adversary can distinguish whether a pair of plaintext-ciphertext is at least as hard as the DDH problem. Here, we construct a game, called PAIR. PAIR is defined via the following game played by the adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

First, the encryption scheme's key generation algorithm is run, with a security parameter as input. Next, the adversary $\mathcal{A}_1$ chooses a plaintext x^* and sends it to an encryption oracle. The encryption oracle encrypts x^* to obtain the ciphertext c_0 and chooses a random string c_1 with the same length as c_0 ($|c_0| = |c_1|$). The encryption oracle chooses a bit b at random and give a pair $x^* \rightleftarrows c_b$ to the adversary $\mathcal{A}_2$.

After receiving the pair from the encryption oracle, the adversary $\mathcal{A}_2$ determines if the pair $x^* \rightleftarrows c_b$ is correct or not. At the end of the game, the adversary $\mathcal{A}_2$ outputs "Yes" if she thinks the pair is correct; otherwise, she outputs "No". If the probability that the answer is correct, $\Pr[\mathsf{Succ}^{\mathsf{PAIR}}_{\mathcal{A},\Pi}(k)] = \frac{1}{2} + \frac{\epsilon}{2}$, then the adversary's advantage is defined to be ϵ. Recall the proof in Theorem 4, once the adversary has the ability to ask $\mathcal{J}$-query $j = 2 \cdot Y^{r_2} \bmod p$, she also can ask $\mathcal{J}$-query $j = Y^{r_2} \bmod p$. Assume that she can ask $j = Y^{r_2} \bmod p$. We can see that the game PAIR is in the original ElGamal encryption scheme. The following theorem shows that PAIR in the ElGamal encryption is at least as hard as the DDH problem.

Theorem 8 (PAIR of Π as least as hard as the DDH). *If there exists a (t, ϵ)-breaker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for Π in the* PAIR *sense in the standard model, then there exists constants c and a (t', ϵ')-translator $\mathcal{T}$ solves the* DDH *problem with overwhelming probability, where*

$$\epsilon' > \epsilon \; and \; t' = t + c.$$

and c denotes the time of a coin flipping and some overhead.

Proof. We construct a translator $\mathcal{T}$ to solve an instance (g, X, Y, Z) of the DDH problem in Definition 6 as follows.

Translator: $\mathcal{T}(g, X, Y, Z)$
Run $\mathcal{A}_1(pk)$
$\mathcal{A}_1$ outputs x^*
$c_0 \leftarrow x^* \cdot Z$
$c_1 \leftarrow_R \{0,1\}^k$
$b \leftarrow_R \{0,1\}$
Output 1 if $\mathcal{A}_2(x^* \rightleftarrows (X, c_b))$ =Yes
End.

Let $p_1 = \Pr[\mathcal{T}(g, g^r, g^s, g^{rs}) = 1]$ and $p_2 = \Pr[\mathcal{T}(g, g^r, g^s, g^z) = 1]$. By the assumption of $\mathcal{A}$, $p_1 \geq \frac{1}{2} + \frac{\epsilon}{2}$. If $z \leftarrow_R \mathbb{Z}_q$, $x^* \cdot g^z$ and c_1 are both uniformly distributed over $\mathbb{Z}_p$, then we have $p_2 = \frac{1}{2}$. Thus, $\mathcal{T}$ solves the DDH problem with advantage $\frac{\epsilon'}{2} = |p_1 - p_2| > \frac{\epsilon}{2}$. From the specification of $\mathcal{T}$, it runs within time $t + c$. □

4.3 Performance Analysis

In this section, we shall compare the computational complexity and ciphertext size of the ElGamal-Extension encryption scheme with those of the ElGamal encryption scheme and Fujisaki-Okamoto's conversion of the ElGamal encryption scheme [31]. We first define the following notations.

T_{EXP}	the time for computing modular exponentiation.
T_{MUL}	the time for computing modular multiplication.
T_{INV}	the time for computing modular inversion.
T_H	the time for computing the adopted $\mathcal{H}(\cdot)$.
T_J	the time for computing the adopted $\mathcal{J}(\cdot)$.
T_{EQU}	the time for comparing two strings are equal or not.

Assume that the whole plaintext x with length $n \cdot k$ is divided into x_1, x_2, $\cdots$, x_n and the length of each x_i is k. To encrypt x_i, the ElGamal encryption scheme requires $2T_{\text{EXP}} + T_{\text{MUL}}$ and the corresponding ciphertext (y_1, y_3) has the length $2 \cdot k$. To derive the plaintext x_i, it requires $T_{\text{EXP}} + T_{\text{MUL}} + T_{\text{INV}}$. Hence, the total computational complexity of encrypting x requires $n \cdot (2T_{\text{EXP}} + T_{\text{MUL}})$, the total length of the corresponding ciphertexts is $n \cdot 2 \cdot k$, and the total computational complexity of decrypting requires $n \cdot (T_{\text{EXP}} + T_{\text{MUL}} + T_{\text{INV}})$.

For the same plaintext x with the length $n \cdot k$ in the ElGamal-Extension encryption scheme, the maximal length of plaintext is limited by k_0. The number of divisions is $\frac{n \cdot k}{k_0} = n + \lceil \frac{n \cdot (2k_1 + l)}{k_0} \rceil$. Let $n' = n + \lceil \frac{n \cdot (2k_1 + l)}{k_0} \rceil$. To encrypt x_1, the ElGamal-Extension scheme requires $4T_{\text{EXP}} + 3T_{\text{MUL}} + T_H + T_J$ and the corresponding ciphertext $(y_1, y_2, y_{3,1})$ has the length $3 \cdot k$. To derive the plaintext x_1, it requires $2T_{\text{EXP}} + 3T_{\text{MUL}} + T_H + T_J + T_{\text{INV}} + T_{\text{EQU}}$. Note that, to encrypt other $n' - 1$ plaintexts $x_2, \cdots, x_{n'}$, it is not necessary to compute the values $y_1 = g^{r_1} \bmod p$, $y_2 = g^{r_2} \bmod p$, $Y^{r_1} \bmod p$, and $Y^{r_2} \bmod p$ again. Hence, the time $4T_{\text{EXP}}$ is only needed for x_1. The total computational complexity of encrypting x requires $4T_{\text{EXP}} + n' \cdot (3T_{\text{MUL}} + T_H + T_J)$. The ciphertext $(y_1, y_2, y_{3,1}, \cdots, y_{3,n'})$ has the length $2 \cdot k + n' \cdot k$. To decrypt other $n' - 1$ ciphertexts $(y_{3,2}, \cdots, y_{3,n'})$, the values $y_1^s \bmod p$ and $y_2^s \bmod p$ have also been computed. The total computational complexity of decrypting requires

$2T_{\text{EXP}} + n' \cdot (3T_{\text{MUL}} + T_H + T_J + T_{\text{INV}} + T_{\text{EQU}})$.

For the same plaintext x with the length $n \cdot k$ in Fujisaki-Okamoto's conversion of the ElGamal encryption scheme, the maximal length of plaintext is limited by $k_0 + k_1 + l$. Let $n'' = \frac{n \cdot k}{k_0 + k_1 + l} = \frac{n \cdot (k_0 + k_1 + l) + n \cdot k_1}{k_0 + k_1 + l} = n + \lceil \frac{n \cdot k_0}{k_0 + k_1 + l} \rceil$ be number of pieces. For each encryption and decryption, which only require additional one random function operation in the ElGamal encryption scheme. Here, we directly show the computational complexity of their scheme in Table 4.1. Please see [31] for more details.

According to Table 4.1, under the same length $n \cdot k$ of plaintext, it is obvious that the ElGamal-Extension encryption scheme is more efficient than the ElGamal encryption scheme. As we know, to compute $g^c \bmod p$ by repeatedly squaring and multiplying requires an average of 240 1024-bit modular multiplications, i.e., $T_{\text{EXP}} = 240T_{\text{MUL}}$. For the computational complexity of encryption and decryption in the ElGamal encryption scheme and Fujisaki-Okamoto's scheme, the number of T_{EXP} is dependent on n and n'', respectively. But it is fixed in the ElGamal-Eextension encryption scheme. For the ciphertext size, it is $2n$ times of k in the ElGamal encryption scheme and Fujisaki-Okamoto's scheme. In the ElGamal-Extendsion encryption scheme, it is $2 + n'$ times of k. The ElGamal-extension encryption scheme has less ciphertext size if $n' < 2 \cdot (n - 1)$. On the other hand, the ElGamal-Extension encryption scheme provide the same highest level of security given IND-CCA2 notation as in Fujisaki-Okamoto's scheme.

Table 4.1: Computational complexity, ciphertext size among three encryption schemes

	Encryption	Decryption	Ciphertext size	GOAL-ATK
ElGamal scheme	$n \cdot (2T_{\text{EXP}} + T_{\text{MUL}})$	$n \cdot (T_{\text{EXP}} + T_{\text{MUL}} + T_{\text{INV}})$	$n \cdot 2 \cdot k$	IND-CPA
Fujisaki-Okamoto scheme	$n'' \cdot (2T_{\text{EXP}} + T_{\text{MUL}} + T_J)$	$n'' \cdot (T_{\text{EXP}} + T_{\text{MUL}} + T_{\text{INV}} + T_J + T_{\text{EQU}})$	$n \cdot 2 \cdot k$	IND-CCA2
ElGamal-extension scheme	$2T_{\text{EXP}} + n' \cdot (3T_{\text{MUL}} + T_H + T_J)$	$2T_{\text{EXP}} + n' \cdot (3T_{\text{MUL}} + T_H + T_J + T_{\text{INV}} + T_{\text{EQU}})$	$2 \cdot k + n' \cdot k$	IND-CCA2

* with the same size $n \cdot k$ of a plaintext x.

4.4 Discussions

In this Chapter, we have proposed an efficient ElGamal-Extension encryption scheme and showed it is secure in the IND-CCA2 sense in the random oracle model. Not only does the proposed scheme provides higher secure level, but also the computational complexities of encryption and decryption in the proposed scheme are more efficient than those in the ElGamal encryption scheme and Fujisaki-Okamoto's scheme when the plaintext is large enough.

Furthermore, we design a new IND-CPA$_{PAIR}$ to demonstrate the security of the ElGamal-Extension encryption scheme when using only two random numbers. There is a question as to whether IND-CPA$_{PAIR}$ accurately demonstrate the security. There maybe a more suitable pair goal and adversary model for demonstrating the security. It is conjectured that involvement of the decryption oracle in providing the adversary with plaintext-ciphertext cryptanalysis training in IND-CCA and IND-CCA2 (says IND-CCA$_{PAIR}$ and IND-CCA2$_{PAIR}$) do not provide the adversary with any undue advantage. If we give the decryption oracle a different position in IND-CCA$_{PAIR}$ and IND-CCA2$_{PAIR}$, the results and advantage given to the adversary may be different. For example, if the decryption oracle is lain in after the adversary obtain the plaintext-ciphertext pair and before she sends x_0, x_1, what are the effects? Future work will undertake to answer these and other questions.

Chapter 5

Password Authenticated Key Exchange Protocols

5.1 Introduction

The rapid growth of networks in both number and size encourages more and more computers to link together for sharing various kinds of data and exchanging huge amounts of information. Two parties need to encrypt and authenticate their message in order to protect the privacy and authenticity of these messages. One way of doing so is to use public key encryption and signatures, but which need the support of the Public Key Infrastructure (PKI). The cost associated with these primitives may be too high for certain applications. Furthermore, Law et al. [51] proposed the MQV protocol, which is still protected under the PKI. Smart [75] and Yi [86] further proposed identity-based authenticated key exchange protocols based on Weil pairing to obtain lower communication overhead and less computation complexity. However, the involved certification management, cryptography calculation, and the additional communication overhead caused

by the digital signature.

Another way of addressing this problem is for users to first establish a comment secret key via a key exchange protocol such as the Diffie-Hellman key exchange protocol and then use this key to derive keys for symmetric encryption and message authentication schemes. However, the secret key established from the key exchange protocol should be authenticated first. Otherwise, the channel between the users is not safe. For example, Diffie-Hellman key exchange protocol suffers from the Man-in-the-Middle attack.

There are many types of key exchange protocols currently in use. They all have their own strengths and weaknesses. One of the most popular protocols is the 3-party Kerberos authentication system [77]. The Internet Key Exchange (IKE) protocol uses the 2-party SIGMA protocol [48] as a standard for signature-based modes. Password based protocols are another type of key exchange system that have received attention recently.

PASSWORD-BASED AUTHENTICATED KEY EXCHANGE. Password-based authenticated key exchange (PAKE) protocols are the most widely used methods. They assume a more realistic scenario in which two parties share a common secret key are not uniformly distributed over a large space, but rather chosen from a small set of possible values (a four-digit pin, for example). It is more convenient since human-memorable passwords are simpler to use without any additional cryptographic devices. In practice, people hardly find long random string passwords easy to use and remember. It would be much more user-friendly if the password is a meaningful string that people can recognize easily such as a natural language phrase. However, the human-memorable passwords narrow down the possibilities and

make it easier for adversary to succeed guessing the passwords with a non-negligible chance, so-called *dictionary* (*password guessing*) attacks. Dictionary attacks are attacks in which an adversary tries to break the security of a scheme by a brute-force method, in which it tries all possible combinations of passwords in a given a small set of values (i.e., the dictionary). The dictionary attacks are usually divided in two categories: *off-line* and *on-line* dictionary attacks.

The goal of password-based key exchange protocols is to restrict the adversary's success to on-line dictionary attacks only. In these attacks, the adversary must be present and interact with the system in order to be able to verify whether its guess is correct. The system can detect on-line guessing by counting the failed trials. If a certain number of failed attempts has occurred, the use of a password is invalidated or blocked.

PASSWORD-BASED AUTHENTICATED KEY EXCHANGE IN THE 3-PARTY SETTING. In large-scale communication environments, similar to the disadvantage of symmetric cryptosystem in Section 2.1, password management can be a tough task. Assume that a communication network has n users, and any two of them exchange a key via the 2-party key exchange protocol. Therefore, there will be $\frac{n(n-1)}{2}$ passwords to be shared, and all those passwords have to be stored securely. Many works [37, 38, 50, 78] have extended the 2-party key exchange protocol into the 3-party applications, in which a trusted server S exists to mediate between the two communication parties A and B to allow their mutual authentication. In this way, any user only needs to share a password with the server. It is particularly well suited for large-scale communication environments.

A nature generic construction of a 3-party PAKE protocol from any 2-party PAKE protocol presented by Abdalla, Fouque, and Ponitcheval [4]. In this thesis, we focus on the 2-party PAKE and present a protect password change (PPC) protocol. If the use of password has a certain number of failed attempts, the users can via the PPC protocol to arbitrarily change their passwords.

5.2 Related Work

Password-based authenticated key exchange (PAKE) has been extensively studied in the last few year. The seminal work in this area is the encrypted key exchange protocol proposed by Bellovin and Merritt [16], where two users in this area is the encrypted version of the Diffie-Hellman key exchange protocol. In their protocol, each flow is encrypted using the password shared between these two users as the symmetric key.

On the other hand, by using a pre-shared password technique along with the Diffie-Hellman scheme, Seo and Sweeney [70] proposed a PAKE protocol without any symmetric cryptosystems or asymmetric cryptosystems. Two parties (a client C and a server S) online can use a pre-shared password technique to authenticate each other and apply the Diffie-Hellman scheme to establish a session key. Sun [80], Tseng [81] and Lu, Lee, and Hwang [54] separately showed that the Seo-Sweeney PAKE protocol is insecure under the threat of the replay attack and off-line dictionary attack. At the same time, Lin, Chang, and Hwang [53] and Tseng [81] separately proposed an improvement on the Seo-Sweeney PAKE protocol to withstand these at-

tacks. However, Hsieh, Sun, and Hwang [42] have pointed out that the Lin-Chang-Hwang scheme is still vulnerable to the off-line dictionary attack. On the other hand, Ku and Wang [49] have also shown that Tseng's scheme is vulnerable to the backward replay attack [36] and forged authenticator attack, and they gave an improvement on Tseng's scheme in the meantime.

Unfortunately, the above schemes or improved schemes lack a proper security model. The first security model for 2-party authenticated key exchange protocol was introduced by Bellare and Rogaway [11]. Later, Bellare et al. [10] and Boyko et al. [20] separately extended the security model to the password-based setting, with security analyses of the above 2-party password-based key exchange, under idealized assumptions, such as the random oracle and the ideal cipher models. Furthermore, some 2-party PAKE protocols [32, 34, 46] are provably secure in the standard model. For the 3-party setting, the first work in this area is the protocol of Needham and Schroeder [60], which inspired the *Kerberos* distributed system. Later, Bellare and Rogaway [14] introduced a formal security model in this scenario along with a construction of the first provably secure symmetric-key-based key distribution scheme. Recently, the first provably secure 3-party PAKE protocol was proposed Abdalla, Fouque, and Ponitcheval [4], which define a new notation called key privacy. That is even though the server's help is required to establish a session key between two users in the system, the server should not be able to gain any information on the value of that session key. In [4], they called their new and stronger model as the Real-Or-Random (ROR) model and Bellare et al.'s model as the Find-Then-Guess (FTG) model. It is worth pointing out that, as proven in [4], any scheme that is proven in the ROR model is also secure in the FTG model. The converse,

however, is not necessarily true due to the non-tightness of the security reduction.

In Section 5.4, we examine some PAKE-related schemes [49, 53, 70, 81] and mounted a forged authenticator attack on those schemes to successfully cheat the two parties into believing in the wrong session key. Table 5.1 below is a summary table of the security of all those schemes. Recently, Yeh and Sun [85], and Kobara and Imai [47] have also combined the pre-shared password technique and the Diffie-Hellman scheme to achieve the same purpose the PAKE protocol intends to, respectively. Both schemes can withstand those attacks shown in Table 5.1 and provide perfect forward secrecy [45]. Lee et al. [52] further proposed the parallel version of the Yeh-Sun scheme. Two parties in their scheme compute the message during the protocol simultaneously. In fact, the scheme still need that one of two parties to send out the request message first and then another one knows to prepare the reply message. Hence, the protocol is not real parallel.

On the other hand, some schemes additionally provides the protected password change (PPC) protocols, which allow a client changes its password freely. However, we point out that the Tseng-Jan-Chien [82] and the Hwang-Yeh [43] schemes are vulnerable to the forged authenticated; that is, any adversary can intercept the request for changing passwords sent by a legal client and modify it with a wrong password along with a forged authenticator.

In Chapter 6, we shall present a simpler authenticated key exchange protocol by modifying the Yeh-Sun scheme [85]. This scheme is proven secure when the symmetric-encryption primitive is instantiated via a mask genera-

Table 5.1: Summary of related schemes in PAKE

	Seo-Sweeney [70]	Tseng [81]	Lin et al. [53]	Ku-Wang [49]
Withstand Man-in-Middle Attack	Yes	Yes	Yes	Yes
Withstand Dictionary Attack	*No [54, 80]	*No [Section 5.4]	*No [42]	*No [Section 5.4]
Withstand Replay Attack	*No [81]	Yes	Yes	Yes
Withstand Backward Replay Attack	*No [49]	*No [49]	Yes	Yes
Withstand Forged Authenticator Attack	*No [Section 5.4]	*No [49], [Section 5.4]	*No [Section 5.4]	*No [Section 5.4]
Provide Perfect Forward Secrecy	*No [80]	Yes	Yes	*No [Section 5.4]

*No [reference]: [reference] points out that the scheme cannot withstand / achieve the attack / perfect forward secrecy.

tion function that is the product of the message with a hash of the password. At the same time, we shall also present a new protected password change protocol which unlike the previously proposed schemes [47, 49, 53, 70, 81, 85] where the parties cannot arbitrarily change their own passwords, offers users the freedom of changing passwords at will. The proposed PAKE and PPC schemes are formally proven using Ballare, Poincheval and Rogaway's security model [10].

5.3 The Security Model

In this section, we recall the security model of Bellare, Pointcheval, and Rogaway (FTG model) [10], which is principally used formally as follows.

(1) Define the Characteristics of Participating Entities

PROTOCOL PARTICIPANTS. We denote by C and S two parties that can participate in the key exchange protocol P. A party may have several *instances*, called *oracles*, involved in distinct concurrent executions of P. We denote U^i as the instance i of a participant U, which is either a client or a server.

LONG-LIVED KEYS. Each client $C \in$ client holds a low-entropy pw_C. Each server $S \in$ server holds a value $pw_S[C]$. The value $pw_S[C]$ is denoted a derived password. Schemes where $pw_S[C] = pw_C$ are called symmetric; in general, $pw_S[C]$ may differ from pw_C, i.e. S employ a hash $\mathcal{G}$ and stores pw_C as $\mathcal{G}(pw_C)$. We call pw_C and $pw_S[C]$ as the long-lived keys and assume that the password is drawn from the dictionary Dict according to the distribution $\mathcal{D}_{pw}$. $\mathcal{D}_{pw}(q)$ denotes as the probability to be in the most probable set of q

passwords as follows:

$$\mathcal{D}_{pw}(q) = \max_{\rho \subseteq \mathsf{Dict}} \left\{ \Pr_{pw \in \mathcal{D}_{pw}} [pw \in \rho | \#\rho \leq q] \right\}.$$

Note that if we denote by $\mathcal{U}_N$ the uniform distribution among N passwords, $\mathcal{U}_N(q) = q/N$.

SESSION IDENTITY AND PARTNER IDENTITY. The session identity SID is used to uniquely name the ensuing session. SID(U^i) is the concatenation of all flows with the oracle U^i. PID(U^i)=U', denoted as U^i, is the communication with another participant U'. Both SID and PID are publicly available.

ACCEPTING AND TERMINATING. There are two states, ACC(U^i) and TERM(U^i), for an oracle U^i. ACC(U^i)=$true$ denotes that U^i has enough information to compute a session key SK. At any time an oracle can accept messages right away. As soon as U^i is accepted, SK(U^i), SID(U^i) and PID(U^i) are defined. When an oracle sends or receives the last message of the protocol, receives an invalid message, or misses an expected message, the state of TERM(U^i) is set to $true$. As long as U^i is terminated, no message will be sent out.

(2) Define an Adversary's Capabilities

The adversary $\mathcal{A}$ has an endless supply of oracles and models various queries to them. Each query models a capability of the adversary, such as forward secrecy, know-key security, etc. The six queries and their responses are listed below.

- Send(U^i, m): This query models $\mathcal{A}$ sending a message m to U^i. $\mathcal{A}$ gets back from his query the response which U^i would have generated in processing message m and updates SID, PID, and its state. $\mathcal{A}$ in the

form Send(U^i, start) initiates an execution of the protocol.

- Execute(C^i, S^j): This query models $\mathcal{A}$ obtaining an honest execution of the protocol in the middle of two oracles C^i and S^j. Execute(C^i, S^j) models $\mathcal{A}$ obtaining an honest execution of the protocols between two oracles C^i and S^j.

- Reveal(U^i): This query models $\mathcal{A}$ obtaining a session key SK with an unconditional return by U^i. The query is for dealing with know-key security. The Reveal query is only available if the state ACC(U^i)=$true$.

- Corrupt(U): This query models $\mathcal{A}$ obtaining a long-lived key pw with an unconditional return by U. The query is for dealing with forward secrecy. As in [10], we assume the weak corruption model in which the internal states of all instances of that user are not returned to $\mathcal{A}$.

- Hash(q): In the ideal hash model, $\mathcal{A}$ gets hash results by making queries to a random oracle. After receiving this query, the random oracle will check whether q has been queried. If so, it returns the result previously generated to $\mathcal{A}$; otherwise it generates a random number r and sends it to $\mathcal{A}$, and stores (q, r) into the hash list $\Lambda_{\mathcal{H}}$, which is a record set used to record all previous Hash queries.

- Test(U^i): This query models the semantic security of the session key SK (the *indistinguishability* between the real session key and a random string). During an execution of the protocol, $\mathcal{A}$ can make any of the above queries, and at once, asks for a Test query. Then, U^i flips a coin b and returns SK if $b = 1$ or a random string with length |SK| if $b = 0$. The query is only available if U^i is *fresh*. $\mathcal{A}$ outputs a bit b' and *wins* the game of breaking the protocol if $b = b'$.

Execute-query may at first seem useless since $\mathcal{A}$ already can carry out an honest execution among oracles. Yet, the query is essential for properly dealing with password guessing attacks. The number q_s of Send-queries directly asked by the adversary does not take into account the number of Execute-queries. Therefore, q_s represents the number of flows the adversary has built by itself, and the therefore the number of passwords it would have tried.

(3) Definitions of Security

FRESHNESS. An oracle U is identified as *fresh* (or holds a *fresh* SK) if the following three conditions are satisfied. (1) U^i has been accepted, (2) No oracle has been asked for a Corrupt-query before U^i is accepted, and (3) Neither U^i nor its partner has been asked for a Reveal-query.

PARTNERING. We say two oracles C^i and S^j are partnered if the following conditions are satisfied. (1) C^i and S^j have been accepted, (2) SK(C^i)=SK(S^j), (3) SID(C^i)∩SID(S^j)≠0, (4) PID(C^i)=S and PID(S^j)=C, and (5) No other oracle accepts SK=SK(C^i)=SK(S^j).

AKE SEMANTIC SECURITY. AKE referred to as Authenticated Key Exchange. Consider an execution of the protocol P by the adversary $\mathcal{A}$, in which the latter is given to access to the Execute, Send, and Test oracles and asks at most one Test query to a *fresh* instance U^i. Let b' be his output. Such an adversary is said to win the experiment defining the sematic security if $b' = b$, where b is the hidden bit used by the Test-query. Let Succ denote the event which the adversary wins this game.

The advantage of $\mathcal{A}$ in violating the AKE sematic security of the protocol

P and the advantage function of the protocol P, when passwords are draw from a dictionary Dict, are defined, respectively, as follows:

$$\mathsf{Adv}^{\mathsf{AKE}}_{P,\mathsf{Dict}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1,$$
$$\mathsf{Adv}^{\mathsf{AKE}}_{P,\mathsf{Dict}}(t, R) = \max_{\mathcal{A}} \left\{ \mathsf{Adv}^{\mathsf{AKE}}_{P,\mathsf{Dict}}(\mathcal{A}) \right\},$$

where maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of oracle queries). The definition of time-complexity is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size [2]. The probability rescaling was added to make the advantage of an adversary that simply guesses the bit b equal to 0.

MUTUAL AUTHENTICATION. A protocol is said to achieve *mutual authentication* if each party can be ensured that it has established a session key with the players it intended to. The above property of AKE semantic security means that only legitimate participants can obtain the secret session key, and any adversary cannot learn information about the key. This is also know as *implicit authentication*. In the context of password-based schemes, authentication between the players is often done through authenticators. An authenticator is only can computed withe the knowledge of a secret password. We denote by $\mathsf{Succ}^{\mathsf{auth}_S}_{P,\mathsf{Dict}}(\mathcal{A})$ the success probability of an adversary $\mathcal{A}$ trying to impersonate the server in the protocol P. This is the probability with which a client instance accepts without having a server partner. Similarly, $\mathsf{Succ}^{\mathsf{auth}_C}_{P,\mathsf{Dict}}(\mathcal{A})$ denotes the success probability of an adversary $\mathcal{A}$ trying to impersonate the client in the protocol P. We denote the probability of violating mutual authentication by $\mathsf{Succ}^{\mathsf{MA}}_{P,\mathsf{Dict}}(\mathcal{A})$. It is trivial that

$$\mathsf{Succ}^{\mathsf{MA}}_{P,\mathsf{Dict}}(\mathcal{A}) = \mathsf{Succ}^{\mathsf{auth}_S}_{P,\mathsf{Dict}}(\mathcal{A}) + \mathsf{Succ}^{\mathsf{auth}_C}_{P,\mathsf{Dict}}(\mathcal{A}).$$

We say the protocol P is MA-secure if $\mathsf{Succ}^{\mathsf{MA}}_{P,\mathsf{Dict}}(\mathcal{A})$ is negligible.

5.4 Attacks on Some Password Authenticated Key Exchange Protocols

Here, we give the following notations whichthrough this chapter.

κ	Let κ be the cryptographic security parameter.
$\mathbb{G}$	Let $\mathbb{G}$ denote a finite cyclic group of order q, where $\|q\| = \kappa$. Let g be a generator of $\mathbb{G}$ and assume it is included in the description of $\mathbb{G}$.
C, S	The communication parties between a client C and a sever S.
id_C	C's identity, which should to be unique to index the verified table stored in the server's database.
id_S	S's identity, which should to be unique to index the server.
pw	C's password secretly shared with S.
$\bigoplus$	The exclusive operator.
$A \rightarrow B :< m >$	The message m sent from A to B.
SK	The session key SK.
$\mathcal{G}(\cdot), \mathcal{H}(\cdot)$	Two secure one-way hash functions.

In the following, we show some password authenticated key exchange protocols are insecure, which only give attack-response analyses.

The Tseng Scheme

We review the Tseng scheme [81] as follows. The scheme has a predetermined way to generate the two integers $Q \in \mathbb{Z}_q$ and $Q^{-1} \in \mathbb{Z}_q$ from the password pw and secretly shared between C and S. It is similar to employ a hash function $\mathcal{G}$ with an input pw and maps to $\mathbb{Z}_q$, i.e. $Q = \mathcal{G}(pw)$. The protocol is composed of two phases, the key establishment phase and the

key verification phase, as follows:

key Establishment Phase

Step 1. $C \rightarrow S$: $< id_C, X_1 >$

C chooses an integer $a \in_R \mathbb{Z}_q$ and computes $X_1 = g^{aQ}$. Then, C sends id_C and X_1 to S.

Step 2. $S \rightarrow C$: $< id_S, Y_1 >$

After receiving C's message, S chooses an integer $b \in_R \mathbb{Z}_q$ to compute $Y_1 = g^{bQ}$. Then, S sends id_S and Y_1 to C.

Step 3. After receiving S's message, C computes $Y = Y_1^{Q^{-1}} = g^b$ and $\mathsf{SK}_C = Y^a = g^{ab}$.

Step 4. After receiving C's message, S computes $X = X_1^{Q^{-1}} = g^a$ and $\mathsf{SK}_S = X^b = g^{ab}$.

key Verification Phase

Step 5. $C \rightarrow S$: $< id_C, Y >$

C sends id_C and X_1 to S.

Step 6. $S \rightarrow C$: $< id_S, X >$

S sends id_S and X to C.

Step 7. C and S check whether $X = g^a$ and $Y = g^b$ hold or not, respectively. If they hold, C and S are sure that they have the common session key $SK = g^{ab}$.

Ku and Wang [49] have shown that the Tseng scheme is vulnerable to the backward replay attack and forged authenticator attack. They proposed an improved version of the Seo-Sweeny scheme [70]. However, we will present another forged authenticator attack on the Tseng scheme, and it will still successfully break the Ku-Wang scheme as well as the others [53, 70]. On the other hand, we will show that the Tseng scheme and the Ku-Wang scheme are also weak in front of the off-line dictionary attack.

FORGED AUTHENTICATOR ATTACK. An adversary $\mathcal{A}$ tries to fool C and S into believing a wrong session key in the Tseng scheme. $\mathcal{A}$ first prepares a value $e \in \mathbb{Z}_q$ and its inversion $e^{-1} \in \mathbb{Z}_q$. In the key establishment phase, upon seeing X_1 sent by C in Step 1, $\mathcal{A}$ replaces it with $X_1' = (X_1)^e = g^{aQe}$. Then, C performs Step 3 and S performs Steps 2 and 4. C and S will separately obtain the session keys $\mathsf{SK}_C = Y^a = g^{ab}$ and $\mathsf{SK}_S = X^b = g^{abe}$, where $Y = (Y_1)^{Q^{-1}} = g^b$ and $X = (X_1')^{Q^{-1}} = g^{ae}$. Next, they separately verify the validity of the session keys SK_C and SK_S. After receiving Y in Step 5, the check equation $Y = g^b$ will hold in Step 7 on S's side, so S will believe that it and C have agreed on a common session key. Upon seeing X is sent by S in Step 6, $\mathcal{A}$ replaces it with $X' = (X)^{e^{-1}} = g^a$. The check equation $X' = g^a \bmod p$ will hold in Step 7 on C's side, and C will too believe that it and S have agreed on a common session key. However, $\mathsf{SK}_C = g^{ab}$ is not equal to $\mathsf{SK}_S = g^{abe}$.

OFF-LINE DICTIONARY ATTACK. $\mathcal{A}$ tries to reveal the secret values Q and Q^{-1} shared between C and S by mounting the off-line dictionary attack. $\mathcal{A}$ first intercepts $X_1 = g^{aQ}$ sent by C in Step 1 and masquerades as S to send $Y_1 = g^e$ in Step 2. C computes $Y = Y_1^{Q^{-1}} = g^{eQ^{-1}}$ and $\mathsf{SK}_C = Y^a = g^{eQ^{-1}a}$. In

Step 5, C sends Y to S. After intercepting Y, $\mathcal{A}$ can verify the correctness of the guessing password by checking whether $Y^Q = g^e$ holds or not because $Y^Q = Y_1^{Q^{-1}Q} = g^e$.

The Ku-Wang Scheme

We have presented another forged authenticator attack in the Tseng scheme, which also threatens the security of the Ku-Wang scheme. Furthermore, the off-line dictionary attack is still successful in their scheme. The key establishment phase in the Ku-Wang scheme is the same as that in the Tseng scheme. We only briefly review the key verification phase of the Ku-Wang scheme that makes a difference.

key Verification Phase

Step 5. $C \rightarrow S$: $< id_C, K_C >$

C computes $K_C = (\mathsf{SK}_C)^Q = g^{abQ}$. Then, C sends id_C and K_C to S.

Step 6. $S \rightarrow C$: $< id_S, X >$

When K_C is received, S checks whether $K_S = (\mathsf{SK}_S)^{Q^{-1}}$. If it holds, S believes that it has obtained the correct X_1 and C has obtained the correct Y_1. Then, S sends id_S and X to C.

Step 7. When X is received, C checks whether $X = g^a$. If it holds, C believes that it has obtained the correct Y_1 and S has obtained the correct X_1.

FORGED AUTHENTICATOR ATTACK. $\mathcal{A}$ performs the same work in the key establishment phase, described in the above. In the key verification phase, upon seeing K_C is sent by C in Step 5, $\mathcal{A}$ replaces it with $K'_C = (K_C)^e = (g^{abQ})^e$. The check equation $\mathsf{SK}_S = (K'_1)^{Q^{-1}} = g^{abe}$ in Step 6 will hold. S will

believe that it has obtained the correct X_1 and C has obtained the correct Y_1. Then, S sends $X = g^{ae}$ to C. $\mathcal{A}$ replaces it with $X' = (X)^{e^{-1}} = g^a$. The check equation $X_1' = g^a \bmod p$ in Step 7 will hold, which will make C believe that it has obtained the correct Y_1 and S has obtained the correct X_1. However, $\mathsf{SK}_C = g^{ab}$ is not equal to $\mathsf{SK}_S = g^{abe}$.

As a matter of fact, the forged authenticator attack can easily be mounted to break the existing password-related methods [53, 70]. Because all the schemes have the common weakness, any adversary can use some value to replace the original value sent by C in the key establishment phase and then use its inversion to make S return to the original value sent to C in the key verification phase. This will make C and S believe the wrong session key.

OFF-LINE DICTIONARY ATTACK. For the same reason, in the Ku-Wang scheme, $\mathcal{A}$ performs the same work in the key establishment phase. In the key verification phase, C computes $K_C = (\mathsf{SK}_C)^Q = g^{ae}$ and sends it to S in Step 5. After intercepting K_C, $\mathcal{A}$ can verify the correctness of the guessing password by checking whether $K_C = (X_1)^{Q^{-1}}$ holds or not because $K_C = (X_1)^{Q^{-1}} = g^{ae}$.

In both schemes, if the password pw is poorly chosen, the adversary can determine Q or Q^{-1} by using the equations to verify if the guessing password is correct. On the other hand, in the Ku-Wang scheme, when a password is compromised, the old session key SK_C can be recovered by computing $(K_C)^{Q^{-1}} = \mathsf{SK}_C = g^{ab}$. Therefore, their scheme cannot provide perfect forward secrecy.

We have presented the forged authenticator attack and the off-line dictio-

nary attack to subvert the security of the Tseng scheme and the Ku-Wang scheme. As we have proved, the Ku-Wang scheme is weak against the forged authenticator attack and the off-line dictionary attack; moreover, the forged authenticator attack can be used to break all the existing PAKE-related schemes.

The Tseng-Jan-Chien Scheme

We shall first briefly review the Tseng-Jan-Chien protected password change protocol [82] and then show how the forged authenticator attack can work on their scheme. In the system, the server stores a client C's id_C and pw as $v\_idpw = \mathcal{H}(id_C, pw)$ in the database.

The protected password changing scheme works as follows:

Step 1. $C \rightarrow S$: $< id_C, C\_idpw >$

C chooses a random number $c \in_R \mathbb{Z}_q$ and computes $rc = g^c \bmod p$. Then, it computes $C\_idpw = \mathcal{H}(id_C, pw) \oplus rc$ and sends it along with id_C to S.

Step 2. $S \rightarrow C$: $< id_S, S\_idpw, S\_auth\_token >$

S first recovers rc from C_idpw by computing $C\_idpw\_digest \oplus v\_idpw$. Then, S chooses a random number $s \in_R \mathbb{Z}_q$ and computes $rs = g^s$ and $rcs = (rc)^s = g^{cs}$. Next, S computes $S\_idpw = v\_idpw \oplus rs$ and $S\_auth\_token = \mathcal{H}(v\_idpw, rcs, rc)$, and then it sends them and id_S to C.

Step 3. $C \rightarrow S$: $< id_C, C\_auth\_token, C\_new\_idpw >$

C first recovers rs from S_idpw by computing $S\_idpw \oplus \mathcal{H}(id_C, pw)$.

Then it computes $rcs = (rs)^c = g^{sc}$ and uses it together with its own $H(id_C, pw)$ and rc to compute $\mathcal{H}(\mathcal{H}(id_C, pw), rcs, rc)$, which is then compared with the received S_auth_token from S. If they match, C computes $C\_auth\_token = \mathcal{H}(\mathcal{H}(id_C, pw), rcs, rs)$. Then, C chooses a new password new_pw and computes

$$C\_new\_idpw = \mathcal{H}(id_C, new\_pw) \oplus \mathcal{H}(\mathcal{H}(id_C, pw), rcs).$$

Finally, C sends $< id_C, C\_auth\_token, C\_new\_idpw >$ to S.

Step 4. S uses its own v_idpw, rcs and rs to compute $\mathcal{H}(v\_idpw, rcs, rs)$ and compares it with the received C_auth_token. If they match, S recovers $H(id_C, new\_pw)$ from C_new_idpw by computing

$$\mathcal{H}(id_C, new\_pw) = C\_new\_idpw \oplus \mathcal{H}(v\_idpw, rcs)$$

and then stores $v\_idpw = \mathcal{H}(id_C, new\_pw)$ in the database.

The difference between the protected password transmission scheme and protected password changing scheme is that C additionally sends C_new_idpw to S for changing passwords in the latter scheme. In the following, we point out Tseng-Jan-Chien protected password changing scheme is vulnerable to the forged authenticator attack; that is, any adversary can intercept the request for changing passwords sent by a legal client and modify it with a wrong password.

FORGED AUTHENTICATOR ATTACK. Note that C sends $< id_C, C\_auth\_token$, $C\_new\_idpw >$ in Step 3 to S, and the messages C_auth_token and C_new_idpw are used to enable the server to authenticate the client and to obtain $\mathcal{H}(id_C$, $new\_pw)$, respectively. However, because the two messages are separated,

the adversary can replace C_new_idpw with a random number ra. After receiving $< id_C, C\_auth\_token, ra >$, S checks the validity of C_auth_token. Since C_auth_token is generated by the legal client, S will accept it. Then, S computes $ra \oplus H(v\_idpw, rcs)$ and stores v_idpw=$ra \oplus H(v\_idpw, rcs)$ in the database.

Unfortunately, the client is mistakenly convinced that it has successfully changed from the old password pw to a new password new_pw. When the client tries to login the server the next time, the server will reject the client's login request because the client cannot recover rs from S_idpw by computing $S\_idpw \oplus H(id_C, new\_pw)$ and therefore cannot compute C_auth_token correctly. As a result, the server will conclude that the client is illegal, and the client will not be able to change its password successfully.

The Hwang-Yeh Scheme

The different from the Tseng scheme and the Ku-Wang scheme is that the Hwang-Yeh scheme [43] employs the public key cryptosystem. However, there are still some security flaws in the Hwang-Yeh password change scheme. Any adversary can intercept the request for changing passwords sent by a legal user and modify it with a wrong password. As a result, the user will not be able to successfully login the server next time.

The main difference between Hwang-Yeh password transmission scheme and password change scheme is that in the latter the client sends a password change request to the server. In the system, the server stores $v\_pw = H(pw)$ instead of pw for each client in the database. Here, we only present the password transmission scheme.

Step 1. $C \rightarrow S$: $< id_C, C\_cipher >$

C encrypts the random number rc and along with pw with the server's public key PK_S denoted as $C\_cipher = \mathcal{E}_{PK_S}(rc, pw)$ and send it with id_C as a login request to S.

Step 2. $S \rightarrow C$: $< id_S, S\_auth\_token, S\_rs >$

S decrypts C_cipher to obtain rc and pw by using its private key. Then, it computes the hash value $\mathcal{H}(pw)$ and checks whether $\mathcal{H}(pw) = v\_pw$ holds or not. If it holds, S chooses a random number rs and computes $S\_auth\_token = rs \oplus rc$ and $S\_rs = \mathcal{H}(rs)$. Then, S sends $< id_S, S\_auth\_token, S\_rs >$ to C.

Step 3. $C \rightarrow S$: $< id_C, C\_auth\_token, C\_new\_pw >$

C retrieves rs by computing $S\_auth\_token \oplus rc$ and then verifies the consistency between the retrieved rs and the received S_rs. If the result is positive, C chooses a new password new_pw and computes $C\_auth\_token = \mathcal{H}(rc, rs)$ and $C\_new\_pw = \mathcal{H}(new\_pw) \oplus \mathcal{H}(rc+1, rs)$. Finally, C sends $< id_C, C\_auth\_token, C\_new\_pw >$ to S.

Step 4. $S \rightarrow C$: access granted or access denied

S computes the hash value $\mathcal{H}(rc, rs)$ and checks whether $\mathcal{H}(rc, rs) = C\_auth\_token$ holds or not. If it holds, S can obtain $\mathcal{H}(new\_pw)$ by computing $C\_new\_pw \oplus \mathcal{H}(rc+1, rs)$ and then store $v\_pw = \mathcal{H}(new\_pw)$ in the database.

Obviously, by employing the public key cryptosystem on the server's side to protect the transmitted password, Hwang and Yeh have effectively avoided the guessing attack and server spoofing that treated the Peyravian-Zunic schemes. However, we show that the Hwang-Yeh scheme is also vulnerable

to the forged authenticator attack as follows.

FORGED AUTHENTICATOR ATTACK. Upon seeing $< id_C, C\_auth\_token, C\_new\_pw >$ sent by C in Step 3, the adversary $\mathcal{A}$ replaces C_new_pw with a random number ra. After receiving $< id_C, C\_auth\_token, rc >$, S first computes the hash value $\mathcal{H}(rc, rs)$ and checks whether $\mathcal{H}(rc, rs) = C\_auth\_token$ holds or not. Since C_auth_token is computed by C, the equation $\mathcal{H}(rc, rs) = C\_auth\_token$ checked by the server will turn out positive. Then, S computes $ra \oplus \mathcal{H}(rc + 1, rs)$ and stores $v\_pw = ra \oplus \mathcal{H}(rc + 1, rs)$ in place of $\mathcal{H}(pw)$ in the database.

However, C is under the impression that it has successfully changed from an old password pw to a new password new_pw. Once the client logins to S the next time, it sends $< id_C, C\_cipher = \mathcal{E}_{PK_S}(rc, new\_pw) >$ to S in Step 1. In Step 2, S decrypts the message to obtain rc and new_pw with its private key. Then, S computes the hash value $\mathcal{H}(new\_pw)$ and check whether $\mathcal{H}(new\_pw) = v\_pw$ holds or not. However, $\mathcal{H}(new\_pw)$ is not equal to v_pw because $v\_pw = ra \oplus \mathcal{H}(rc + 1, rs)$. The server will reject the client's login request.

Chapter 6

Simple Password Authenticated Key Exchange and Protected Password Change Protocols

6.1 Password Authenticated Key Exchange Protocol

In this chapter, we shall present a simple password authenticated key exchange (PAKE) protocol by modifying the Yeh-Sun scheme [85]. This scheme is proven secure when the symmetric-encryption primitive is instantiated via a mask generation function that is the product of the message with a hash of the password. At the same time, we shall also present a new protected password change (PPC) protocol which unlike the previously proposed schemes [47, 49, 53, 70, 81, 85] where the parties cannot arbitrarily change their own passwords, offers users the freedom of changing passwords at will. The proposed PAKE protocol is formally proven using the Ballare-Poincheval-Rogaway security model. The provable security is

demonstrated by reduction. Here, we give the following notations which through this chapter.

$\mathbb{G}$	Let $\mathbb{G}$ denote a finite cyclic group of order q, where $\|q\| = \kappa$. Let g be a generator of $\mathbb{G}$ and assume it is included in the description of $\mathbb{G}$.
C, S	The communication parties between a client C and a sever S.
id_C	C's identity, which should to be unique to index the verified table stored in the server's database.
id_S	S's identity, which should to be unique to index the server.
pw	C's password secretly shared with S.
$A \rightarrow B :< m >$	The message m sent from A to B.
SK	The session key SK.
$\mathcal{G}$	A full-domain hash from $\{0,1\}^*$ into $\mathbb{G}$.
$\mathcal{H}_i$	Two hash functions from $\{0,1\}^*$ to $\{0,1\}^\kappa$, for $i = 0, 1$.

The parties initially share a low-quality password pw. The password authenticated key exchange protocol then runs as in Figure 6.1 and described as follows.

Step 1. $C \rightarrow S$: $< id_C, R_C^* >$

C chooses a random number $c \in_R \mathbb{Z}_q$, computes $R_C = g^c$ and $R_C^* = R_C \times \mathsf{PW}$, where $\mathsf{PW} = \mathcal{G}(pw)$. Then C sends $< id_C, R_C^* >$ to S,

Step 2. $S \rightarrow C$: $< id_S, R_S, \mathsf{Auth}_S >$

After receiving $< id_C, R_C^* >$, S recovers R_C by computing R_C^*/PW. Then S chooses a random number $s \in_R \mathbb{Z}_q$, computes $R_S = g^s$, $K_S = (R_C)^s = g^{cs}$, $\mathsf{Auth}_S = \mathcal{H}_1(K_S, R_C)$ and sends $< id_S, R_S, \mathsf{Auth}_S >$ to C.

Step 3. $C \rightarrow S$: $< id_C, \mathsf{Auth}_C >$

After receiving $< id_S, R_S, \mathsf{Auth}_S >$, C computes $K_C = (R_S)^c = g^{sc}$ and verifies whether the received Auth_S is equal to $\mathcal{H}_1(K_C, R_C)$ or not. If it is, C computes $\mathsf{Auth}_C = \mathcal{H}_1(K_C, R_S)$ and sends it to S.

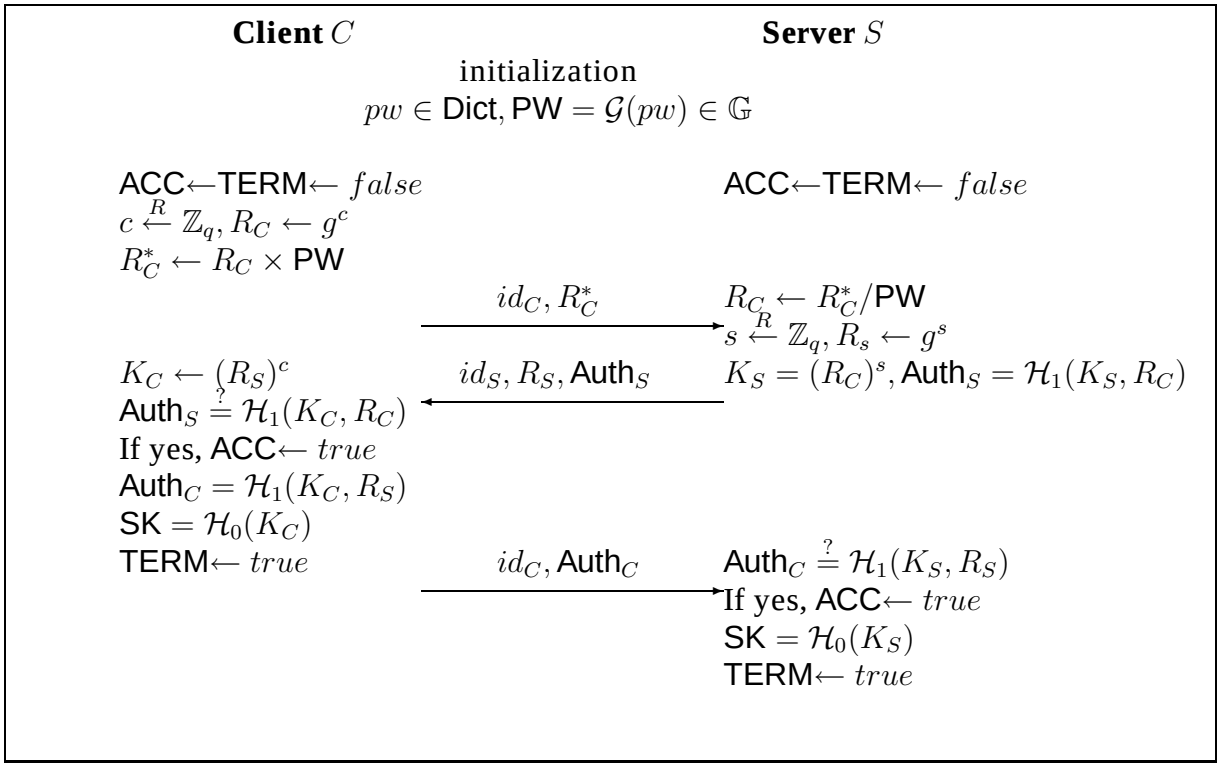


Figure 6.1: An execution of the protocol PAKE

After receiving Auth_C, S verifies whether it is equal to $\mathcal{H}_1(K_S, R_S)$ or not. If it is, S and C agree on the common session key $\mathsf{SK} = \mathcal{H}_0(K_C) = \mathcal{H}_0(K_S) = \mathcal{H}_0(g^{cs})$.

6.2 Protected Password Change Protocol

Assume that C wants to change it's old password pw to a new password $newpw$, C needs to follow these steps and illustrated in Figure 6.2.

Step 1*. $C \rightarrow S$: $< id_C, R_C^*, R_C^\dagger >$

C chooses a random number $c \in_R \mathbb{Z}_q$, computes $R_C = g^c$, $R_C^* = R_C \times$

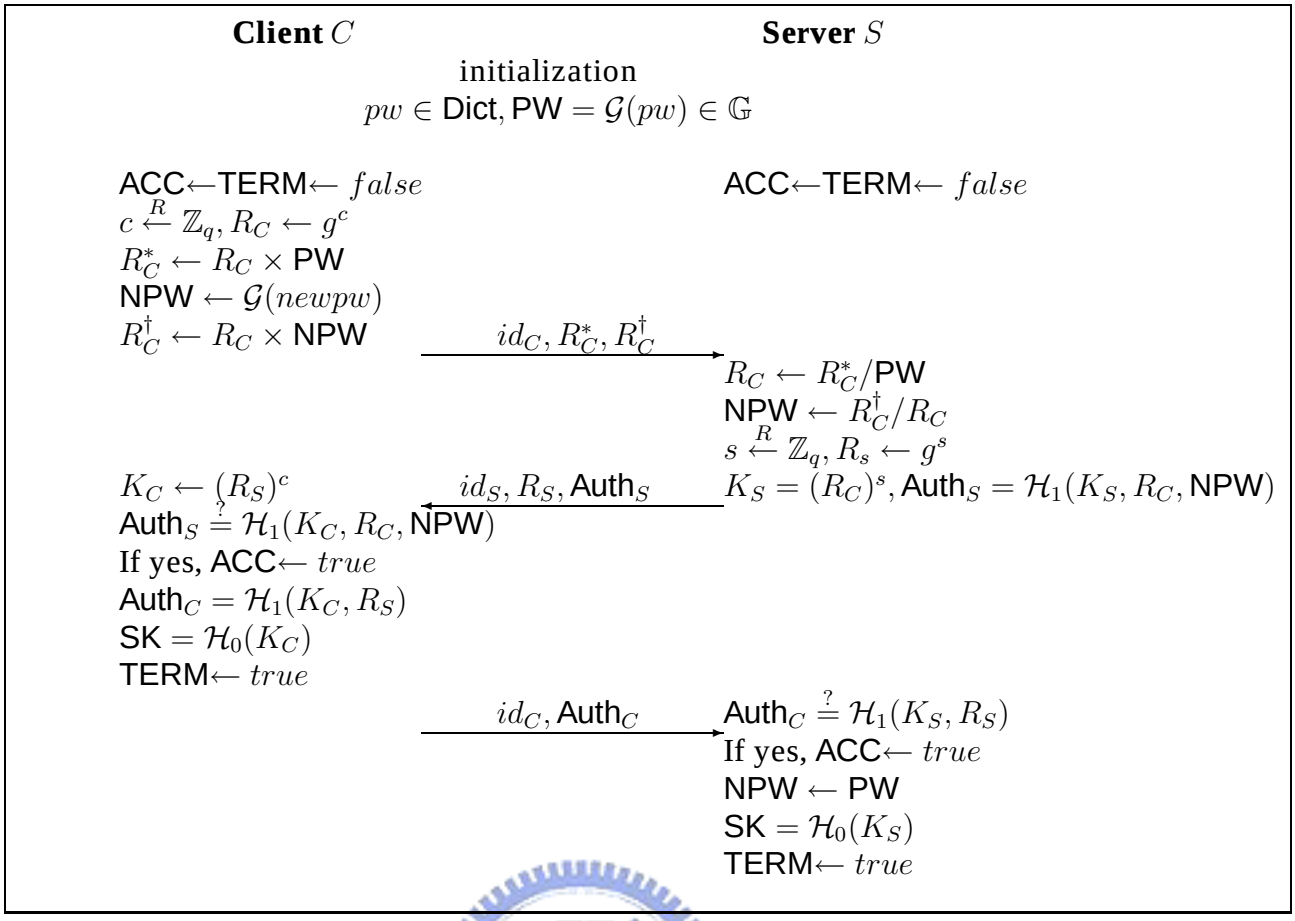


Figure 6.2: An execution of the protocol SPC

PW and $R_C^\dagger = R_C \times \mathsf{NPW}$, where $\mathsf{NPW} = \mathcal{G}(newpw)$. Then C sends $< id_C, R_C^*, R_C^\dagger >$ to S.

Step 2*. $S \rightarrow C$: $< id_S, R_S, \mathsf{Auth}_S) >$

After receiving $< id_C, R_C^*, R_C^\dagger >$, S recovers R_C by computing R_C^*/PW and uses the recovered R_C to obtain NPW by computing $R_C^\dagger/R_C$. Then S chooses a random number $s \in_R \mathbb{Z}_q$, computes $R_S = g^s$, $K_S = (R_C)^s = g^{cs}$, and $\mathsf{Auth}_S = \mathcal{H}_1(K_S, R_C, \mathsf{NPW})$. Then S sends $< id_S, R_S, \mathsf{Auth}_S) >$ to C.

Step 3*. $C \rightarrow S$: $< id_C, \mathsf{Auth}_C >$

After receiving $< id_S, R_S, \mathsf{Auth}_S >$, C computes $K_C = (R_S)^c = g^{sc}$ and verifies whether the received Auth_S is equal to $\mathcal{H}_1(K_C, R_C, \mathsf{NPW})$ or not. If it holds, C computes $\mathsf{Auth}_C = \mathcal{H}_1(K_C, R_S)$ and sends it to S.

After receiving $< id_C, \mathsf{Auth}_C >$, S verifies whether the recovered Auth_C is equal to $\mathcal{H}_1(K_S, R_S)$ or not. If it is, C has successfully changed its old password pw to the new password $newpw$ and S has successfully updated its PW to NPW in its database. At the same time, S and C agree on the common session key $\mathsf{SK} = \mathcal{H}_0(K_S) = \mathcal{H}_0(K_C)$.

6.3 Security Analysis

In this section, we show the scheme is provable security in the random oracle model. We shall employ and simplify the security model [10] to formally prove the security of PAKE and PPC in the random oracle model. The PAKE protocol distributes session keys that are semantically secure and provide mutual authentication. Figure 6.3 shows the initialization of both protocols. Figures 6.4 and 6.5 separately show how instances in the PAKE and PPC protocols behave in response to messages (runs the PAKE and PPC protocols).

Before putting the protocols to work, each oracle sets $\mathsf{ACC}(U^i) \leftarrow \mathsf{TERM}(U^i) \leftarrow false$; and $\mathsf{SK}(U^i) \leftarrow \mathsf{SID}(U^i) \leftarrow \mathsf{PID}(U^i) \leftarrow null$;.

AKE Security. We separately denote the AKE advantage of $\mathcal{A}$ in attacking PAKE and PPC as $\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A})$ and $\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PPC},\mathsf{Dict}}(\mathcal{A})$; the advantages are taken over all bit tosses. The advantage of $\mathcal{A}$ distinguishing the session key

Initialize(1^κ)
- Select a finite cyclic group $\mathbb{G}$ of prime order q with g as a generator, where $\|q\| = \kappa$. - Hash functions. $\mathcal{H}_i : \{0,1\}^* \to \{0,1\}^\kappa$ for $i = 0, 1$, $\mathcal{G} : \{0,1\}^* \to \mathbb{G}$. - A client $C \in$ client holds a password pw. $pw \leftarrow$ Dict. - A server $S \in$ server holds the hash values PW of pw. PW $\leftarrow \mathcal{G}(pw)$.

Figure 6.3: Specification of protocol initialization

is given by

$$\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1,$$

$$\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PPC},\mathsf{Dict}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{Succ}] - 1.$$

Protocols PAKE and PPC are AKE-secure if $\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A})$ and $\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PPC},\mathsf{Dict}}(\mathcal{A})$ are negligible, respectively.

Computational Diffie-Hellman Assumption. A (t, ε)-$\mathsf{CDH}_{g,\mathbb{G}}$ attacker, in finite cyclic group $\mathbb{G}$ of prime order q with g as a generator, is a probabilistic machine Δ running in time t such that its success probability $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(\Delta)$, given random elements g^x and g^y to output g^{xy}, is greater than ε:

$$\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(\Delta) = \Pr[\Delta(g^x, g^y) = g^{xy}] \geq \varepsilon.$$

We denote by $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t)$ the maximal success probability over every adversaries running time within time t. The computational Diffie-Hellman assumption states that $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t) \leq \varepsilon$ for ant t/ε not too large.

Theorem 9. *Let $\mathcal{A}$ be an adversary against the* AKE*-security of the* PAKE *protocol*

Execute-queries

Execute(C^i, S^j)

1. Send$_1(C^i,$ start$)$

   $c \overset{R}{\leftarrow} \mathbb{Z}_q, R_C \leftarrow g^c, R_C^* \leftarrow R_C \times$ PW, msg-out$_1 \leftarrow < id_C, R_C^* >$,

   internal-state$_C^i \leftarrow < c, R_C >$

   return msg-out$_1$

2. Send$_2(S^j, m_1)$

   $< ID_I, \alpha > \leftarrow m_1, R_C \leftarrow \alpha /$PW, $s \overset{R}{\leftarrow} \mathbb{Z}_q, R_S \leftarrow g^s, K_S \leftarrow R_C^s$,

   Auth$_S \leftarrow \mathcal{H}_1(K_S, R_C)$, msg-out$_2 \leftarrow < id_S, R_S,$ Auth$_S >$

   internal-state$_S^j \leftarrow < R_S, K_S >$

   return msg-out$_2$

3. Send$_3(C^i, m_2)$, where $m_2 \neq$ start

   $< ID_I, R_S,$ Auth$_S > \leftarrow m_2, < c, R_C > \leftarrow$ internal-state$_C^i, K_C \leftarrow R_S^c$

   if $\mathcal{H}_1(K_C, R_C) =$ Auth$_S$

   Auth$_C = \mathcal{H}_1(K_C, R_S)$, msg-out$_3 \leftarrow < id_C,$ Auth$_C >$

   SK$(C^i) \leftarrow \mathcal{H}_0(K_C)$, SID$(C^i) \leftarrow <$ msg-out$_1, m_2,$ msg-out$_3 >$

   PID$(C^i) \leftarrow id_S,$ ACC$(C^i) \leftarrow$ TERM$(C^i) \leftarrow true$

   else msg-out$_3 \leftarrow *$

   return msg-out$_3$

4. Send$_4(S^j, m_3)$

   $< ID_I,$ Auth$_C > \leftarrow m_3, < R_B, K_S > \leftarrow$ internal-state$_S^j$

   if $\mathcal{H}_1(K_S, R_S) =$ Auth$_C$

   SK$(S^j) \leftarrow \mathcal{H}_0(K_S)$, SID$(S^j) \leftarrow < m_1,$ msg-out$_2 >$

   PID$(S^j) \leftarrow id_C,$ ACC$(S^j) \leftarrow$ TERM$(S^j) \leftarrow true$

   return $null$

Figure 6.4: Specification of protocol PAKE

Execute-queries

Execute(C^i, S^j)

1. Send$_1(C^i,$ start$)$
   $c \overset{R}{\leftarrow} \mathbb{Z}_q, R_C \leftarrow g^c, R_C^* \leftarrow R_C \times$ PW, NPW $\leftarrow \mathcal{G}(newpw)$,
   $R_C^\dagger \leftarrow R_C \times$ NPW, msg-out$_1 \leftarrow < id_C, R_C^*, R_C^\dagger >$,
   internal-state$_C^i \leftarrow < c, R_C,$ NPW $>$
   return msg-out$_1$
2. Send$_2(S^j, m_1)$
   $< ID_I, \alpha, \beta > \leftarrow m_1, R_C \leftarrow \alpha /$PW, NPW $\leftarrow \beta / R_C, s \overset{R}{\leftarrow} \mathbb{Z}_q$,
   $R_S \leftarrow g^s, K_S \leftarrow R_C^s,$ Auth$_S \leftarrow \mathcal{H}_1(K_S, R_C,$ NPW$)$,
   msg-out$_2 \leftarrow < id_S, R_S,$ Auth$_S >$, internal-state$_S^j \leftarrow < R_C, K_S,$ NPW $>$
   return msg-out$_2$
3. Send$_3(C^i, m_2)$, where $m_2 \neq$ start
   $< ID_I, R_S,$ Auth$_S > \leftarrow m_2, < c, R_C,$ NPW $> \leftarrow$ internal-state$_C^i, K_C \leftarrow R_S^c$
      if $\mathcal{H}_1(K_C, R_C,$ NPW$) =$ Auth$_S$
         Auth$_C = \mathcal{H}_1(K_C, R_S)$, msg-out$_3 \leftarrow < id_C,$ Auth$_C >$
         SK$(C^i) \leftarrow \mathcal{H}_0(K_C)$, SID$(C^i) \leftarrow <$ msg-out$_1, m_2,$ msg-out$_3 >$
         PID$(C^i) \leftarrow id_S,$ ACC$(C^i) \leftarrow$ TERM$(C^i) \leftarrow true$
      else msg-out$_3 \leftarrow *$
   return msg-out$_3$
4. Send$_4(S^j, m_3)$
   $< ID_I,$ Auth$_C > \leftarrow m_3, < R_B, K_S > \leftarrow$ internal-state$_S^j$
      if $\mathcal{H}_1(K_S, R_S) =$ Auth$_C$
         PW $\leftarrow$ NPW
         SK$(S^j) \leftarrow \mathcal{H}_0(K_S)$, SID$(S^j) \leftarrow < m_1,$ msg-out$_2 >$
         PID$(S^j) \leftarrow id_C,$ ACC$(S^j) \leftarrow$ TERM$(S^j) \leftarrow true$
   return $null$

Figure 6.5: Specification of protocol PPC

within a time bound t, after q_s and q_h. Then we have:

$$\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE},\mathsf{Dict}}(t, q_s, q_h) \leq \mathcal{D}_{pw}(q_s) + q_s \times q_h \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t_1) + \frac{q_s}{2^\kappa},$$

where t_1 is the running time of $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}$.

Proof. There are two ways that might lead to $\mathcal{A}$ successfully attacking the AKE-security of the PAKE protocol. First, $\mathcal{A}$ might obtain the long-lived key and impersonate C or S by mounting the password guessing attack. Second, $\mathcal{A}$ might directly obtain the session key by solving the CDH problem. In the following, we shall analyze the probability of the two situations one by one. To analyze a situation, the others are assumed to be under some known probability.

Dictionary Attacks: C and S separately chooses $c \in_R \mathbb{Z}_q$ and $s \in_R \mathbb{Z}_q$ at random, which implies R_C and R_S are random numbers. Hence, $\mathcal{A}$ observes that the message $R^*_C = R_C \times \mathsf{PW}$ returned from Send_1 is independent of other messages. Therefore, the adversary gets no advantage for the off-line dictionary attack. The probability of the on-line dictionary attack making way is bounded by $\mathcal{D}_{pw}(q_s)$ as follows:

$$\lambda \leq \mathcal{D}_{pw}(q_s)$$

The on-line guessing attack can be prevented by letting the server take the appropriate intervals between trials. Furthermore, we also provide the PPC protocol to allow clients to change their own passwords.

CDH Attack (Session key): $\mathcal{B}$ plays the role of a *simulator* for *indistinguishability*. It uses the PAKE protocol to respond to all $\mathcal{A}$'s queries and deal with the CDH problem. $\mathcal{B}$ sets up the long-lived key pw, picks a random number

i from $[1, q_{s_1}]$, and sets a counter $cnt = 0$. With the challenge $\psi = (g^x, g^y)$, $\mathcal{B}$ tries to output g^{xy}. When $\mathcal{A}$ makes Send_1, $\mathcal{B}$ answers according to the protocol to return $\mathsf{msg\text{-}out}_1$ to Send_1 and increases cnt by 1. However, if $cnt = i$, $\mathcal{B}$ answers using the element g^x from the challenge ψ. When $\mathcal{A}$ makes a Send_2 query, $\mathcal{B}$ answers what the protocol says to. However, if the input is the flow corresponding to the challenge ψ i.e. $< id_C, g^x \times \mathsf{PW} >$, $\mathcal{B}$ answers with $< id_S, g^y, \mathcal{H}_1(random, g^x) >$ by using the element g^y from the challenge (g^x, g^y), where $random$ is a random number over $\mathbb{Z}_q$. When $\mathcal{A}$ makes a Send_3 query, $\mathcal{B}$ answers what the protocol says to. If the input is the flow corresponding to the challenge (g^x, g^y) i.e. $< id_S, \mathcal{H}_1(random, g^x) >$, $\mathcal{B}$ answers with $< id_C, \mathcal{H}_1(random, g^y) >$ by using the element g^y from the challenge ψ. If not, $\mathcal{B}$ answers with $\mathsf{msg\text{-}out}_3$ to Send_3. When $\mathcal{A}$ makes a Send_4 query, $\mathcal{B}$ answers what the protocol says to.

When $\mathcal{A}$ makes a $\mathsf{Reveal}(C^i)$ or $\mathsf{Reveal}(S^j)$, $\mathcal{B}$ checks whether the oracle has been accepted and is *fresh*. If so, $\mathcal{B}$ answers by using the session key SK. However, if the session key has to be constructed from the challenge ψ, $\mathcal{B}$ halts. When $\mathcal{A}$ makes a $\mathsf{Corrupt}(C)$, $\mathsf{Corrupt}(S)$, $\mathsf{Execute}(C^i, S^j)$, or $\mathsf{Hash}(m)$, $\mathcal{B}$ answers in a straightforward way. When $\mathcal{A}$ makes a single Test query, $\mathcal{B}$ answers in a straightforward way. However, if the session key has to be constructed from the challenge ψ, $\mathcal{B}$ answers with a random string for the $\mathsf{Test}(C^i)$ or $\mathsf{Test}(S^j)$.

This simulation is perfectly indistinguishable from any execution of the real PAKE protocol except for one execution in which the challenge ψ is involved. The probability α of $\mathcal{B}$ correctly guessing the session key $\mathcal{A}$ will

use Test(U^i) is the probability of $cnt = i$. Then, we have:

$$\alpha = \frac{1}{q_{s_1}} \geq \frac{1}{q_s}$$

Assume that $\mathcal{A}$ has broken the CDH problem ($\mathcal{A}$, outputting b' after the Test query, *wins*), then at least one of the Hash queries equals SK. The probability of $\mathcal{B}$ correctly choosing among the possible Hash queries is:

$$\beta \geq \frac{1}{q_h}$$

From the above description, the probability $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(\mathcal{B})$ that $\mathcal{B}$ outputs g^{xy} from the challenge ψ is the probability ε that $\mathcal{A}$ breaks the AKE-secure scheme multiplied by the probability α that $\mathcal{B}$ correctly guesses the moment at which $\mathcal{A}$ breaks the AKE-secure scheme multiplied by the probability β that $\mathcal{B}$ correctly chooses among the possible Hash queries:

$$\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(\mathcal{B}) = \varepsilon \times \alpha \times \beta \geq \varepsilon \times \frac{1}{q_s} \times \frac{1}{q_h}$$

□

Theorem 10. *Let $\mathcal{A}$ be an adversary against the* AKE*-security of the* PPC *protocol within a time bound t, after q_s and q_h. Then we have:*

$$\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PPC},\mathsf{Dict}}(t, q_s, q_h) \leq \mathcal{D}_{pw}(q_s) + q_s \times q_h \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t_1) + \frac{q_s}{2^\kappa},$$

where t_1 is the running time of $\mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}$.

Proof. This proof is similar to the analysis of PAKE. We omit it here. □

Recently, some PAKE schemes [3, 5, 21] are also proven in Ballare, Poincheval and Rogaway's security model. They incrementally define a sequence of

games and use Shoup's lemma [72] to exactly bound the probability of each event in these games. In the following Theorem 11, we starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_5$ and use Shoup's lemma to bound the probability of each event in these games.

Theorem 11. *Consider the protocol* PAKE*, over a group* $\mathbb{G}$ *of prime order* q*, where* Dict *is a dictionary equipped the distribution* $\mathcal{D}_{pw}$*. Let* $\mathcal{A}$ *be an adversary against the* AKE*-security and* UA*-security of the* PAKE *protocol within a time bound* t*, with less than* q_s *active interactions with the parties (*Send*-queries) and* q_p *passive interactions (*Execute*-queries), and asking* $q_{\mathcal{G}}$ *and* $q_{\mathcal{H}}$ *to* $\mathcal{G}$ *and any* $\mathcal{H}_i$ *respectively,*

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A}) \leq\ & 2 \times \Bigg(\frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
& \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
& 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A}) \Bigg) \\
\mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A}) \leq\ & \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
& \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
& 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{q_s}{2^{\kappa}}
\end{aligned}$$

where τ_e *denotes the computational time for an exponentiation in* $\mathbb{G}$*.*

Proof. We are interested in the event S, which occurs if the adversary correctly guess the bit b involved in the Test(C^i)-queries. Consider the server unilateral authentication (UA) in Figure 6.1, if the authenticator Auth_S checked by C is correct, C will accepts and set the session key. Let A be an event if a client accepts without any server partner. In any game $\mathbf{G}_n$ below, we study the event A_n, and restricted event $\mathsf{SA}_n = \mathsf{S}_n \wedge \neg \mathsf{A}_n$ which means the adversary guess the bit b without breaking authentication.

Game $\mathbf{G}_0$: This is the real protocol, in the random oracle model:

$$\begin{cases} \mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE,Dict}}(\mathcal{A}) = 2 \cdot \Pr[\mathsf{S}_0] - 1, \\ \mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE,Dict}}(\mathcal{A}) = \Pr[\mathsf{A}_0]. \end{cases} \tag{6.1}$$

Game $\mathbf{G}_1$: We simulate the hash oracles $\mathcal{G}$, $\mathcal{H}_0$, and $\mathcal{H}_1$ by maintaining hash lists $\Lambda_{\mathcal{G}}$ and $\Lambda_{\mathcal{H}}$, illustrated in Figure 6.6. We also simulate all the instances, as the real players would do, for the Send (illustrated in Figure 6.7), Execute, Reveal and Test queries (illustrated in Figure 6.8).

$\mathcal{G}$ and $\mathcal{H}_i$ oracles
For a hash-query $\mathcal{H}_i(q)$ - If a record (i, q, r) appears in $\Lambda_{\mathcal{H}}$, the answer is r. - Otherwise, choose a random element $r \in \{0,1\}^{\kappa}$, answer with it, and add the record (i, q, r) to $\Lambda_{\mathcal{H}}$. For a hash-query $\mathcal{G}(q)$ - If a record $(q, r, -)$ appears in $\Lambda_{\mathcal{G}}$, the answer is r. - Otherwise, the answer is r according to the following rule: ► **Rule** $\mathcal{G}^{(1)}$. - Chose a random element $r \in \mathcal{G}$, and adds the record $(q, r, -)$ to $\Lambda_{\mathcal{G}}$. The third element of the records $(q, r, -)$ in $\Lambda_{\mathcal{G}}$ will be used in $\mathbf{G}_4$.

Figure 6.6: Simulation of the hash functions $\mathcal{G}, \mathcal{H}_i$

From this simulation, we easily see that the game is perfectly indistinguishable from the real protocol. We have the following equations.

$$\begin{cases} \Pr[\mathsf{S}_1] = \Pr[\mathsf{S}_0], \\ \Pr[\mathsf{SA}_1] = \Pr[\mathsf{SA}_0]. \end{cases} \tag{6.2}$$

Game $\mathbf{G}_2$: We cancel games in which some collisions appear:

- Collisions on the partial transcripts $((id_C, R^*_C), (id_S, R_S))$, where at least

Send-queries to C

We answer to Send-queries to C^i as follows:

For a $\mathsf{Send}(C^i, \mathsf{start})$-query to C^i is according the following rule:

► **Rule C1**$^{(1)}$

- Choose a random number $\alpha \in \mathbb{Z}_q$, compute $R_C = g^\alpha$ and $R_C^* = R_C \times \mathsf{PW}$.
- Answer (id_C, R_C^*).

For a $\mathsf{Send}(C^i, (id_S, R_S, \mathsf{Auth}_S))$-query is according the following rule:

► **Rule C2**$^{(1)}$

- Compute $K_C = R_S^\alpha$.

► **Rule C3**$^{(1)}$

- Check the authenticator Auth_S.
  - Compute $\mathsf{Auth}'_S = \mathcal{H}_1(K_C, R_C)$.
  - If $\mathsf{Auth}_S = \mathsf{Auth}'_S$, then $\mathsf{ACC}(C^i) = ture$, $\mathsf{SK}(C^i) = \mathcal{H}_0(K_C)$, and compute $\mathsf{Auth}_C = \mathcal{H}_1(K_C, R_S)$ and answer (id_C, Auth_C).
  - Else $\mathsf{ACC}(C^i) = false$.
  - In any case, $\mathsf{TERM}(C^i) = false$.

Send-queries to S

We answer to Send-queries to S^j as follows:

For a $\mathsf{Send}(S^j, (id_C, R_C^*))$-query to S^j is according the following rule:

► **Rule S1**$^{(1)}$

- Choose a random number $\beta \in \mathbb{Z}_q$, compute $R_S = g^\beta$.

► **Rule S2**$^{(1)}$

- Compute $R_C = R_C^*/\mathsf{PW}$, $K_S = R_C^\beta$.

► **Rule S3**$^{(1)}$

- Compute $\mathsf{Auth}_S = \mathcal{H}_1(K_S, R_C)$.
- Answer $(id_S, R_S, \mathsf{Auth}_S)$.

For a $\mathsf{Send}(S^j, (id_C, \mathsf{Auth}_C))$-query is according the following rule:

► **Rule S4**$^{(1)}$

- Check the authenticator Auth_C.
  - Compute $\mathsf{Auth}'_C = \mathcal{H}_1(K_S, R_S)$.
  - If $\mathsf{Auth}_C = \mathsf{Auth}'_C$, then $\mathsf{ACC}(S^j) = ture$, $\mathsf{SK}(S^j) = \mathcal{H}_0(K_S)$, and $\mathsf{TERM}(S^j) = true$.

Figure 6.7: Simulation of protocol PAKE (1)

Other queries
An Execute(C^i, S^j)-query is processed using successively the above simulations of the Send-queries to output $((id_C, R_C^*), (id_S, R_S, \mathsf{Auth}_S), (id_C, \mathsf{Auth}_C))$. An Reveal($U$)-query returns the session key if U accpets. An Test(U)-query first get SK from Reveal(U), and flip a coin b. If $b = 1$, return SK; otherwise, return a random number from $\{0,1\}^\kappa$.

Figure 6.8: Simulation of protocol PAKE (2)

one element of each transcript is generated by an honest party. In other words, at least one of them in each of the q_s active attacks, and all of them in the q_p passive attacks. Thus one of R_C^* or R_S is truly uniformly distributed.

- Collisions on the output of $\mathcal{G}$.
- Collision on the output of $\mathcal{H}_i$.

All probabilities are bounded by the birthday paradox as follows:

$$\Pr[\mathsf{Coll}_2] \leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}}.$$

We have the following inequalities:

$$\begin{cases} |\Pr[\mathsf{A}_1] - \Pr[\mathsf{A}_2]| \leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}}, \\ |\Pr[\mathsf{SA}_1] - \Pr[\mathsf{SA}_2]| \leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}}. \end{cases} \tag{6.3}$$

Game $\mathbf{G}_3$: We simulate the private hash oracle $\mathcal{H}_i' : \{0,1\}^* \to \{0,1\}^\kappa$ for $i = 0, 1$ as usual by maintaining the hash list $\Lambda_{\mathcal{H}'}$, illustrated in Figure 6.9.

$\mathcal{H}'_i$ oracle
For a hash-query $\mathcal{H}'_i(q)$ - If a record (i, q, r) appears in $\Lambda_{\mathcal{H}'}$, the answer is r. - Otherwise, choose a random element $r \in \{0,1\}^\kappa$, answer with it, and adds the record (i, q, r) to $\Lambda_{\mathcal{H}'}$.

Figure 6.9: Simulation of the hash functions $\mathcal{H}'_i$

We compute the authenticator Auth_S and the session key SK using the private oracle $\mathcal{H}'_1$ and $\mathcal{H}'_0$, respectively.

- **Rule S3**$^{(3)}$
  - Compute $\mathsf{Auth}_S = \mathcal{H}'_1(R^*_C)$.
  - Answer $(id_S, R_S, \mathsf{Auth}_S)$.

- **Rule C3**$^{(3)}$
  - Check the authenticator Auth_S
    - Compute $\mathsf{Auth}'_S = \mathcal{H}'_1(R^*_C)$.
    - If $\mathsf{Auth}_S = \mathsf{Auth}'_S$, then $\mathsf{ACC}(C^i) = ture$, $\mathsf{SK}(C^i) = \mathcal{H}'_0(R^*_C, R_S)$, and compute $\mathsf{Auth}_C = \mathcal{H}_1(K_C, R_S)$ and answer (id_C, Auth_C).
    - Else $\mathsf{ACC}(C^i) = false$.
    - In any case, $\mathsf{TERM}(C^i) = false$

- **Rule S4**$^{(3)}$
  - Check the authenticator Auth_C.
    - Compute $\mathsf{Auth}'_C = \mathcal{H}_1(K_S, R_S)$.
    - If $\mathsf{Auth}_C = \mathsf{Auth}'_C$, then $\mathsf{ACC}(S^j) = ture$, $\mathsf{SK}(S^j) = \mathcal{H}'_0(R^*_C, R_S)$, and $\mathsf{TERM}(S^j) = true$.

Since we do no longer need to compute the values K_C and K_S, we can simply the following rules:

- **Rule S2/C2**$^{(3)}$
  Do nothing.

► **Rule C1**$^{(3)}$
- Choose a random number $c \in \mathbb{Z}_q$ and compute $R_C^* = g^c$
- Answer (id_C, R_C^*).

Let $\mathsf{AskH1}_3$ be the event that $\mathcal{A}$ queries $\mathcal{H}_1(K_C, R_S)$ or $\mathcal{H}_1(K_S, R_C)$. Let $\mathsf{AskH0w1}_3$ be the event that $\mathcal{A}$ queries $\mathcal{H}_0(K_C)$ or $\mathcal{H}_0(K_S)$, where some party has accepted, but event $\mathsf{AskH1}_3$ did not happens. Thus, the event $\mathsf{AskH}_3 = \mathsf{AskH1}_3 \vee \mathsf{AskH0w1}_3$ occurs, $\mathcal{A}$ can distinguish the games $\mathbf{G}_3$ and $\mathbf{G}_2$.

$$\begin{cases} |\Pr[\mathsf{A}_2] - \Pr[\mathsf{A}_3]| \leq \Pr[\mathsf{AskH}_3], \\ |\Pr[\mathsf{SA}_2] - \Pr[\mathsf{SA}_3]| \leq \Pr[\mathsf{AskH}_3]. \end{cases} \tag{6.4}$$

The authenticator Auth_S is computed with the simulator's private random oracle $\mathcal{H}_1'$, then it can not be guessed by $\mathcal{A}$, better than at random for each attempt, unless the same transcript $((id_C, R_C^*), (id_S, R_S))$ appeared in another session with a real instance S^j. $\mathbf{G}_2$ has already excluded this case. For the same reason, SK is computed with the simulator's private oracle $\mathcal{H}_0'$, the probability of $\mathcal{A}$ guessing the bit b is $\frac{1}{2}$. We have the following inequality and equation:

$$\begin{cases} \Pr[\mathsf{A}_3] \leq \frac{q_s}{2^\kappa}, \\ \Pr[\mathsf{SA}_3] = \frac{1}{2}. \end{cases} \tag{6.5}$$

Now, consider the probability of event $\mathsf{AskH1}_3$, since the collisions of partial transcripts have been excluded, the event $\mathsf{AskH1}_3$ can be split in three disjoint sub-cases:

1. $\mathsf{AskH1\text{-}Passive}_3$: the transcript $((id_C, R_C^*), (id_S, R_S))$ come from an execution between instances of C and S. This means that both R_C^* and R_S have been simulated.

2. AskH1-WithC_3: the execution involved an instance of C, but R_S has not been sent by any instance of S. This means that R_C^* has been simulated, but R_S has been produced by $\mathcal{A}$.

3. AskH1-WithS_3: the execution involved an instance of S, but R_C^* has not been sent by any instance of C. This means that R_S has been simulated, but R_C^* has been produced by $\mathcal{A}$.

Thus, we have the following equation:

$$\begin{aligned}\Pr[\mathsf{AskH}] &= \Pr[\mathsf{AskH1\text{-}Passive}] + \Pr[\mathsf{AskH1\text{-}WithC}] + \\ &\quad \Pr[\mathsf{AskH1\text{-}WithS}] + \Pr[\mathsf{AskH0w1}] \end{aligned} \tag{6.6}$$

Game $\mathbf{G}_4$: To evaluate the probability of the event AskH, we introduce a random Diffie-Hellman instance (X, Y). We first modify the simulation of the oracle $\mathcal{G}$, involving the element X as follows:

- **Rule $\mathcal{G}^{(4)}$**
  - Choose a random number $\gamma \in \mathbb{Z}_q$, compute $r = X^{-\gamma}$, and the record (q, r, γ) is added to $\Lambda_{\mathcal{G}}$.

The other part Y of the Diffie-Hellman instance in the simulation of the party S.

- **Rule $\mathbf{S1}^{(4)}$**
  - Choose a random number $\delta \in \mathbb{Z}_q$, compute $R_S = Y^{\delta}$.

We excluded that case PW $= 1$ or $R_S = 1$, which are separately denoted events eventPW and eventR_S.

$$\Pr[\mathsf{eventPW} \vee \mathsf{event}R_S] \leq \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q}.$$

We have the following inequality:

$$|\Pr[\mathsf{AskH}_4] - \Pr[\mathsf{AskH}_3]| \leq \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q}. \tag{6.7}$$

Game $\mathbf{G}_5$: It is now possible to evaluate the probability of the event AskH. We cancel a few more games, wherein for some pairs (R_C^*, R_S), R_S is generated by an instance S^j and R_C^* is generated by either $\mathcal{A}$ or an instance C^i, there are two distinct elements PW such that the tuple $(\mathsf{CDH}_{g,\mathbb{G}}(R_C^*/\mathsf{PW}, R_S), R_C^*/\mathsf{PW})$ is in $\Lambda_{\mathcal{H}}$. This denotes as the event CollH_5.

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_4]| \leq \Pr[\mathsf{CollH}_5]. \tag{6.8}$$

We set the upper-bounded of CollH_5 in the following lemma:

Lemma 3. *If for some pair (R_C^*, R_S), involved in a communication with an instance S^j, there are two distinct elements PW_0 and PW_1 such that $(Z_i, R_C^*/\mathsf{PW}_i)$ are in $\Lambda_{\mathcal{H}}$ with $Z_i = \mathsf{CDH}_{g,\mathbb{G}}(R_C^*/\mathsf{PW}, R_S)$, one can solve the computational Diffie-Hellman problem $\mathsf{CDH}_{g,\mathbb{G}}(X, Y)$:*

$$\Pr[\mathsf{CollH}_5] \leq q_{\mathcal{H}}^2 \times \mathsf{Succ}_{g,\mathbb{G}}^{\mathsf{CDH}}(t, \tau_e) \tag{6.9}$$

Proof. There exists such elements $(R_C^*, R_S = Y^{\delta})$, $\mathsf{PW}_0 = X^{-\gamma_0}$, and $\mathsf{PW}_1 = X^{-\gamma_1}$.

$$\begin{aligned} Z_i &= \mathsf{CDH}_{g,\mathbb{G}}(R_C^*/\mathsf{PW}, R_S) \\ &= \mathsf{CDH}_{g,\mathbb{G}}(R_C^* \times X^{\gamma_i}, R_S) \\ &= \mathsf{CDH}_{g,\mathbb{G}}(R_C^*, R_S) \times \mathsf{CDH}_{g,\mathbb{G}}(X, R_S)^{\gamma_i} \\ &= \mathsf{CDH}_{g,\mathbb{G}}(R_C^*, R_S) \times \mathsf{CDH}_{g,\mathbb{G}}(X, Y)^{\delta\gamma_i} \end{aligned}$$

As a consequence, $Z_0/Z_1 = \mathsf{CDH}_{g,\mathbb{G}}(X, Y)^{\delta(\gamma_0 - \gamma_1)}$, and thus $\mathsf{CDH}_{g,\mathbb{G}}(X, Y) = (Z_0/Z_1)^{-\delta(\gamma_0 - \gamma_1)}$. We have excluded the cases $\mathsf{PW} = 1$ and $R_S = 1$ in $\mathbf{G}_4$, thus $\gamma_0 \neq 0$, $\gamma_1 \neq 0$ and $\delta \neq 0$. And since $\mathsf{PW}_0 \neq \mathsf{PW}_1$, we have $\gamma_0 \neq \gamma_1$. Let

CDH be the event that solving the computational Diffie-Hellman problem. If CollH_5 occurs, we can choose the two queries in $\Lambda_{\mathcal{H}}$ to make CDH occurs. The conditional probability of CDH given CollH_5 is as follows:

$$\Pr[\mathsf{CDH}|\mathsf{CollH}_5] = \frac{1}{q_{\mathcal{H}}^2},$$
$$\Pr[\mathsf{CollH}_5] \leq q_{\mathcal{H}}^2 \times \Pr[\mathsf{CDH}].$$

The probability of CDH is defined as $\mathsf{Succ}_{g,\mathbb{G}}^{\mathsf{CDH}}(t, \tau_e)$ with one exponentiation computation in $\mathbb{G}$. ■

We consider the three sub-cases AskH1-Passive, AskH1-WithC, AskH1-WithS of AskH1 and then AskH0w1 as follows:

1. AskH1-Passive: Both R_C^* and R_S have been simulated. We set the upper-bounded of AskH1-Passive in the following lemma:

   Lemma 4. *If for some pair (R_C^*, R_S), involved in a communication with the instances C^i and S^j, such that $(Z, R_C^*/\mathsf{PW})$ are in $\Lambda_{\mathcal{H}}$ with $Z = \mathsf{CDH}_{g,\mathbb{G}}(R_C^*/\mathsf{PW}, R_S)$, one can solve the computational Diffie-Hellman problem $\mathsf{CDH}_{g,\mathbb{G}}(P, Q)$:*

   $$\Pr[\mathsf{AskH1-Passive}_5] \leq q_{\mathcal{H}} \times \mathsf{Succ}_{g,\mathbb{G}}^{\mathsf{CDH}}(t, 2\tau_e) \quad (6.10)$$

   Proof. There exists such elements $(R_C^* = g^c, R_S = Y^\delta)$ and $\mathsf{PW} = X^\gamma$.

   $$\begin{aligned} Z &= \mathsf{CDH}_{g,\mathbb{G}}(R_C^*, R_S) \times \mathsf{CDH}_{g,\mathbb{G}}(X, R_S)^\gamma \\ &= Y^{c\delta} \times \mathsf{CDH}_{g,\mathbb{G}}(X, Y)^{\delta\gamma}. \end{aligned}$$

   As a consequence, $\mathsf{CDH}_{g,\mathbb{G}}(X, Y) = (Z/Y^{c\delta})^{\delta\gamma}$. Since the cases $\delta = 0$ and $\gamma = 0$ have been excluded. If AskH1-Passive occurs, we can solve the computational Diffie-Hellman problem by choosing the query in $\Lambda_{\mathcal{H}}$. It is similar to Lemma 3 to conclude the proof. ■

2. AskH1-WithC: R_C^* has been simulated and R_S has been produced by $\mathcal{A}$. This event corresponds to an attack where $\mathcal{A}$ tries to impersonate S to C. But each authenticator Auth_S sent by $\mathcal{A}$ has been computed with at most one PW value. Without any $\mathcal{G}$-collision (we have excluded in $\mathbf{G}_2$), it corresponds to at most one pw:

$$\Pr[\mathsf{AskH1\text{-}WithA}_5] \leq \mathcal{D}_{pw}(q_s). \tag{6.11}$$

3. AskH1-WithS: R_S has been simulated and R_C^* has been produced by $\mathcal{A}$. Assume that $\mathcal{A}$ guesses the password as pw', chooses a number $a \in \mathbb{Z}_q$ and compute $R_C^* = g^a \times \mathcal{G}(pw')$. If AskH1-WithS occurs, there is at most one element pw such that for $\mathsf{PW} = \mathcal{G}(pw) = \mathcal{G}(pw')$ and a query (R_S^a, R_C) in $\Lambda_{\mathcal{H}}$. From Lemma 3, when applied to games where the event CollH_5 did not occur and without $\mathcal{G}$-collision. We have,

$$\Pr[\mathsf{AskH1\text{-}WithS}_5] \leq \mathcal{D}_{pw}(q_s). \tag{6.12}$$

For AskH0w1, the the above events did not occur, it means only the execution with instances C^i and S^j may lead acceptation. It is the same as that analysis in AskH1-Passive, we have,

$$\Pr[\mathsf{AskH0w1}_5] \leq q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e). \tag{6.13}$$

Now, we conclude the proof. Combining equations and inequalities (6.6), (6.10), (6.11), (6.12), and (6.13), we have:

$$\Pr[\mathsf{AskH}_5] \leq 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e). \tag{6.14}$$

Combining inequalities (6.7), (6.8), and (6.9), we have:

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_3]| \leq \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e). \tag{6.15}$$

Combining inequalities (6.14) and (6.15), we have:

$$\begin{aligned}\Pr[\mathsf{AskH}_3] \leq\ & \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) \\ & + 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e). \end{aligned} \tag{6.16}$$

Combining equations and inequalities (6.2), (6.3), (6.4), and (6.5), we have:

$$\begin{cases} \Pr[\mathsf{A}_0] \leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \Pr[\mathsf{AskH}_3] + \frac{q_s}{2^\kappa} \\ \Pr[\mathsf{SA}_0] \leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \Pr[\mathsf{AskH}_3] + \frac{1}{2} \end{cases} \tag{6.17}$$

Combining inequalities (6.16) and (6.17), we have:

$$\begin{cases} \Pr[\mathsf{A}_0] \leq & \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\ & \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\ & 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{q_s}{2^\kappa} \\ \Pr[\mathsf{SA}_0] \leq & \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\ & \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\ & 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{1}{2} \end{cases} \tag{6.18}$$

From equation (6.1), we can conclude $\mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A})$:

$$\begin{aligned}\mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE},\mathsf{Dict}}(\mathcal{A}) \leq\ & \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\ & \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\ & 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{q_s}{2^\kappa} \end{aligned}$$

See $\Pr[\mathsf{SA}_0]$ in inequality (6.18), we can rewrite as follows:

$$\begin{aligned}
\Pr[\mathsf{SA}_0] &\leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
&\quad \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
&\quad 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{1}{2} \\
\Pr[\mathsf{S}_0] - \Pr[\mathsf{S}_0 \wedge \mathsf{A}_0] &\leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
&\quad \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
&\quad 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{1}{2} \\
\Pr[\mathsf{S}_0] &\leq \frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
&\quad \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
&\quad 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \frac{1}{2} + \Pr[\mathsf{A}_0]
\end{aligned}$$

From equation (6.1), we can conclude $\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE,Dict}}(\mathcal{A})$:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{AKE}}_{\mathsf{PAKE,Dict}}(\mathcal{A}) &\leq 2 \times \Bigg(\frac{q_p^2}{2q^2} + \frac{q_s^2}{2q} + \frac{q_{\mathcal{G}}^2}{2q} + \frac{q_{\mathcal{H}}^2}{2^{\kappa+1}} + \\
&\quad \frac{q_{\mathcal{G}}}{q} + \frac{q_s + q_p}{q} + q_{\mathcal{H}}^2 \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, \tau_e) + \\
&\quad 2 \times \mathcal{D}_{pw}(q_s) + 2 \times q_{\mathcal{H}} \times \mathsf{Succ}^{\mathsf{CDH}}_{g,\mathbb{G}}(t, 2\tau_e) + \mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE,Dict}}(\mathcal{A}) \Bigg)
\end{aligned}$$

□

On the other hand, the evaluation of $\mathsf{Succ}^{\mathsf{Auth}_C}_{\mathsf{PAKE,Dict}}$ can be done similarly to $\mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE,Dict}}$ and then $\mathsf{Succ}^{\mathsf{MA}}_{\mathsf{PAKE,Dict}} = \mathsf{Succ}^{\mathsf{Auth}_S}_{\mathsf{PAKE,Dict}} + \mathsf{Succ}^{\mathsf{Auth}_C}_{\mathsf{PAKE,Dict}}$ can be calculated.

Chapter 7

Conclusions

7.1 Conclusions

In this thesis, we focus on two topics: public key cryptosystems and password authenticated key exchange protocols.

Public Key Cryptosystems. The ElGamal cryptosystem has been proven to be secure in the IND-CPA sense in the standard model if the operation is in QR_p [83]. As we know, the IND-CPA sense is considered as a basic requirement for most provably secure public key cryptosystems. We precisely show that the ElGamal cryptosystem is insecure in the IND-CPA sense if the operation is in not QR_p. For the ElGamal-like cryptosystem [44], we give two simple examples to prove it is insecure in the IND-CPA sense either operated in QR_p or not (employ the key generation $\mathcal{K}$ or $\widehat{\mathcal{K}}$). Besides, the cryptosystem has the probability to be crashed when $Y^{r_1} \oplus (Y^{r_2})^{2^i} \bmod p = 0$.

Since the exclusive-or operation is not suitable for the group operation, the computed values cannot be expected in that group.

However, the motivation for encrypting large messages in public key cryptosystem is practical, since they have bad performance as compared to symmetric cryptosystems. We present an efficient conversion from IND-CPA secure ElGamal encryption scheme to a IND-CCA2 secure extension of the ElGamal encryption scheme in the random oracle model, called the ElGamal-extension cryptosystem. To demonstrate that the ElGamal-Extension scheme is secure using only two random numbers, a new pair GOAL and ATK are constructed called $\text{IND-CPA}_{\text{PAIR}}$. We also prove the proposed scheme is secure in the $\text{IND-CPA}_{\text{PAIR}}$ sense.

Password Authenticated Key Exchange Protocols. The authenticated key exchange protocols are essential for user authentication and session key establishment. Among techniques used for authenticating users, allowing users to choose possibly weak passwords as their own secrets is the most convenient for users. We have broken some PAKE protocols, which only be analyzed in the response-attack. We also proposed a simple PAKE and together with a new PPC protocol, where the parties cannot arbitrarily change their own passwords, offers users the freedom of changing passwords at will. The proposed PAKE protocol is formally proven using the Ballare-Poincheval-Rogaway security model.

Bibliography

[1] A. Shamir A. Biryukov and D. Wagner, "Real-time cryptanalisis of A5/1 on a PC," in *Proceedings of Fast Software Encryption 2000 (FSE 2000)*, pp. 77–86, 1978 of LNCS, 2000.

[2] Michel Abdalla, Mihir Bellare, and Phillip Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," in *Topics in Cryptology-CT-RSA 2001*, Lecture Notes in Computer Science 2020, pp. 8–12, 2001.

[3] Michel Adballa, Emmanuel Bresson, Olivier Chevassut, Bodo Möller, and David Pointcheval, "Provably secure password-based authentication in TLS," in *Proceedings of 2006 ACM Symposium on Information, computer and communications security (AsiaCCS'06)*, pp. 35–45, 2006.

[4] Michel Adballa, Pierre-Alain Fouque, and David Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *PKC 2005*, pp. 65–84, Lecture Notes in Computer Science 3386, 2005.

[5] Michel Adballa and David Pointcheval, "Simple password-based encrypted key exchange protocols," in *CT-RSA 2005*, pp. 191–208, Lecture Notes in Computer Science 3376, 2005.

[6] Joonsang Baek. *Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting*. PhD thesis, Monash University, Jan. 2004.

[7] M. Bellare, "Practice-oriented provable-security," in *Lectures on Data Security (Modern Cryptology in Theory and Practice)*, pp. 1–15, Lecture Notes in Computer Science 1561, 1999.

[8] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notations of security for public-key encryption schemes," in *Advances in Cryptology, CRYPTO'98*, pp. 26–45, Lecture Notes in Computer Science 1462, 1998.

[9] M. Bellare and A. Palacio, "Towards plaintext-aware public-key encryption without random oracles," in *Advances in Cryptology, ASIACRYPT'04*, pp. 48–62, Lecture Notes in Computer Science 3329, 1998.

[10] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attack," in *Advances in Cryptology, EUROCRYPT'00*, pp. 122–138, Lecture Notes in Computer Science 1807, 2000.

[11] M. Bellare and P. Rogaway, "Entity authentication and key distribution," *Advances in Cryptology - CRYPTO '93*, pp. 232–249, Aug. 1993.

[12] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *1st annual Conference on Computer and Communications Security,*, ACM, pp. 62–73, 1993.

[13] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Advances in Cryptology, EUROCRYPT'94*, pp. 92–111, Lecture Notes in Computer Science 950, 1994.

[14] M. Bellare and P. Rogaway, "Provably secure session key distribution - the three party case," in *In 28th Annual ACM Symposium on Theory of Computing*, pp. 57–66, 1996.

[15] M. Bellare and P. Rogaway, "Collision-resistant hashing: towards making UOWHFs practical," in *Advances in Cryptology, CRYPTO'97*, pp. 470–484, Lecture Notes in Computer Science 1294, 1997.

[16] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," in *In 1992 IEEE Symposium on Security and Privacy*, pp. 72–84, 1992.

[17] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1," in *Advances in Cryptology, CRYPTO'98*, pp. 1–12, Lecture Notes in Computer Science 1462, 1998.

[18] D. Boneh, "The decision Diffie-Hellman problem, algorithmic number theory symposium," in *Proceedings of ANTS'98*, pp. 48–63, Lecture Notes in Computer Science 1423, 1998.

[19] D. Boneh and R. Venkatesan, "Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes," in *Advances in Cryptology, CRYPTO'96*, pp. 129–142, Lecture Notes in Computer Science 1109, 1996.

[20] V. Boyko, P. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," in *Advances in*

Cryptology - EUROCRYPT'00, pp. 156–171, Lecture Notes in Computer Science 1807, 2000.

[21] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval, "New security results on encrypted key exchange," in *PKC 2004*, pp. 145–158, Lecture Notes in Computer Science 2947, 2004.

[22] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Advances in Cryptology, CRYPTO'98*, pp. 13–25, Lecture Notes in Computer Science 1462, 1998.

[23] J. Daemen and V. Rijmen. "AES Proposal: Rijndael,". tech. rep., National Institution Standard Technology, USA, March 1999. Document Version 2.

[24] J. Daemen and V. Rijmen, "The block cipher Rijndael," in *Smart Card Research and Applications*, vol. LNCS 1820, pp. 288–296, 2000.

[25] J. Daemen and V. Rijmen, "Rijndael, the advanced encryption standard," *Dr. Dobb's Journal*, vol. 26, no. 3, pp. 137–139, 2001.

[26] Hans Delfs and Helmut Knebl, *Introduction to Cryptography: Principles and Applications*, Springer, 2002.

[27] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[28] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, July 1985.

[29] B. Preneel et al., “Security evaluation of NESSIE first phase, new european schemes for signature,” Integrity and Encryption (NESSIE) project report (IST-1999-12324).

[30] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Advances in Cryptology, CRYPTO’86*, pp. 186–194, Lecture Notes in Computer Science 263, 1986.

[31] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, “RSA-OAEP is secure under the RSA assumption,” in *Advances in Cryptology, CRYPTO’01*, pp. 260–274, Lecture Notes in Computer Science 2139, 2001.

[32] R. Gennaro and Y. Lindell, “A framework for password-based authenticated key exchange,” in *Advances in Cryptology, EUROCRYPT’03*, pp. 524–543, Lecture Notes in Computer Science 2656, 2003.

[33] O. Goldreich, “On the foundations of modern cryptography,” in *Advances in Cryptology, CRYPTO’97*, pp. 46–74, Lecture Notes in Computer Science 1294, 1997.

[34] O. Goldreich and Y. Lindell, “Session-key generation using human password only,” in *Advances in Cryptology, CRYPTO’01*, pp. 408–432, Lecture Notes in Computer Science 2139, 2001.

[35] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[36] L. Gong, “Variations on the themes of message freshness and replay,” *Proc. IEEE Computer Security Foundations Workshop VI*, pp. 131–136, 1993.

[37] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," *Proceedings of 8th IEEE Computer Security Foundation Workshop*, pp. 24–29, 1995.

[38] L. Gong, M. Lomas, R. Needham, and J. Saltzer, "Protecting poorly choosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 648–656, 1993.

[39] J. A. Gordon, "Strong primes are easy to find," in *Advances in Cryptology, EUROCRYPT'84*, pp. 216–223, Lecture Notes in Computer Science 209, 1984.

[40] F. Grieu, "A chosen messages attack on the iso/iec 9796-1 signature scheme," in *Advances in Cryptology, EUROCRYPT'00*, pp. 70–80, Lecture Notes in Computer Science 1807, 2000.

[41] J. Herzog, M. Liskov, and S. Micali, "Plaintext awareness via key registeration," in *Advances in Cryptology, CRYPTO'03*, pp. 548–564, Lecture Notes in Computer Science 2729, 2003.

[42] Bin-Tsan Hsieh, Hung-Min Sun, and Tzonelih Hwang, "Cryptanalysis of enhancement for simple authenticated key agreement algorithm," *IEE Electronic Letters*, vol. 38, no. 1, pp. 20–21, 2002.

[43] Jing-Jang Hwang and Tzu-Chang Yeh, "Improvement on Peyravian-Zunic's password authentication schemes," *IEICE Trans. on Communications*, vol. E85-B, pp. 823–825, 2002.

[44] Min-Shiang Hwang, Chin-Chen Chang, and Kuo-Feng Hwang, "An ElGamal-like cryptosystem for enciphering large messages," *IEEE*

Transactions on Knowledge and Data Engineering, vol. 14, no. 2, pp. 445–446, 2002.

[45] D. P. Jablon, "Strong password only authenticated key exchange," *Computer Communication Review*, vol. 26, pp. 5–26, Oct. 1996.

[46] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Advances in Cryptology, EUROCRYPT'01*, pp. 475–494, Lecture Notes in Computer Science 2045, 2001.

[47] Kazukuni Kobara and Hideki Imai, "Pretty-simple password-authenticated key-exchange protocol proven to be secure in the standard model," *IEICE Transactions on Fundamentals*, vol. E85-A, no. 10, pp. 2229–2237, 2002.

[48] H. Krawczyk, "SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols," in *Advances in Cryptology, CRYPTO'03*, pp. 400–425, Lecture Notes in Computer Science 2729, 2003.

[49] Wei-Chi Ku and Sheng-De Wang, "Cryptanalysis of modified authenticated key agreement protocol," *IEE Electronics Letters*, vol. 36, no. 21, pp. 1770–1771, 2000.

[50] T. Kwon, M. Kang, S. Jung, and J. Song, "An improvement of the password-based authentication protocol K1P on security against replay attacks," *IEICE Transactions on Communications*, vol. E82-B, no. 7, pp. 991–997, 1999.

[51] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "Security of authenticated multiple-key," *Technical report CORR 9805, Department of C&O, University of Waterloo*, 1998.

[52] Sung-Woon Lee, Woo-Hun Kim, Hyun-Sung Kim, and Kee-Young Yoo, "Parallizable simple authenticated key agreement protocol," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 3, pp. 17–22, 2003.

[53] Iuon-Chang Lin, Chin-Chen Chang, and Min-Shiang Hwang, "Security enhancement for the simple authentication key agreement algorithm," in *The Twenty-Fourth Annual International Computer Software and Applications Conference(COMPSAC)'2000*, pp. 113–115, 2000.

[54] Eric Jui-Lin Lu, Cheng-Chi Lee, and Min-Shiang Hwang, "Cryptanalysis of some authenticated key agreement protocols," *International Journal of Computational and Numerical Analysis and Applications*, pp. 151–157, Apr. 2003.

[55] W. Mao, *Modern Cryptography: Theory & Practice*, Prentice Hall, 2004.

[56] John C. Martin, *Introduction to Languages and the Theory of Computation*, McGraw Hill, 1997.

[57] U. Maurer, "Towards proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms," in *Advances in Cryptology, CRYPTO'94*, pp. 271–281, Lecture Notes in Computer Science 839, 1994.

[58] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proc. of the 21st STOC*, pp. 33–43, 1989.

[59] M. Naor and M. Yung, "Public-key cryptosystems provably secure against chosen ciphertext attack," in *Proc. of the 22st STOC*, pp. 427–43, 1990.

[60] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the Association for Computing Machinery*, vol. 21, no. 21, pp. 993–999, 1978.

[61] National Institute of Standards and Technology, "Federal information processing standards (FIPS) publication 180-2, secure hash standard (SHS)," 2004.

[62] T. Okamoto and S. Uchiyama, "A new public-key cryptosystemas as secure as factoring," in *Advances in Cryptology, EUROCRYPT '98*, pp. 308–318, Lecture Notes in Computer Science 1403, 1998.

[63] IEEE P1363, "Standard specifications for public key cryptography," 2000.

[64] P. Paillier and David Pointcheval, "Efficient public-key cryptosystems provaly secure against active adversaries," in *Advances in Cryptology, ASIACRYPT'99*, pp. 165–179, Lecture Notes in Computer Science 1716, 1999.

[65] Duong Hieu Phan and David Pointcheval, "On the security notions for public-key encryption schemes," in *4th conference on Security in Communication Networks'04*, pp. 33–46, Lecture Notes in Computer Science 3352, 2005.

[66] D. Pointcheval, "New public key cryptosystems based on the dependent-RSA problems," in *Advances in Cryptology, EUROCRYPT'99*, pp. 239–254, Lecture Notes in Computer Science 1592, 1999.

[67] M.O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," *Technical Report, MIT/LCS/TR212, MIT Lab., Computer Science Cambridge, MA, USA*, January 1979.

[68] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," in *Advances in Cryptology, CRYPTO'91*, pp. 433–444, Lecture Notes in Computer Science 576, 1991.

[69] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[70] D. Seo and P. Sweeney, "Simple authenticated key agreement algorithm," *IEE Electronics Letters*, vol. 35, no. 13, pp. 1073–1074, 1999.

[71] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Advances in Cryptology, EUROCRYPT'97*, pp. 256–266, Lecture Notes in Computer Science 1233, 1997.

[72] V. Shoup, "OAEP reconsidered," in *Advances in Cryptology, CRYPTO'01*, pp. 239–259, Lecture Notes in Computer Science 2139, 2001.

[73] V. Shoup, "A proposal for an ISO standard for public key encryption (version 2.1)," ISO / IEC JTC 1 / SC 27, 2001.

[74] V. Shoup and R. Gennaro, "Securing threhshold cryptosystem against chosen ciphertext attack," in *Advances in Cryptology, EUROCRYPT'98*, pp. 1–16, Lecture Notes in Computer Science 1403, 1998.

[75] N. P. Smart, "Identity-based authenticated key agreement protocol based on Weil pairing," *Electronics Letters*, vol. 38, no. 13, pp. 630–632, 2002.

[76] M. E. Smid and D. K. Branstad, "The data encryption standard: Past and future," *Proc. of the IEEE*, vol. 76, pp. 550–559, May 1988.

[77] J. G. Steiner, B. C. Neuman, and J. I. Schiller, "Kerberos: An authentication service for open network system," in *Usenix Conference*, pp. 191–201, Feb. 1988.

[78] M. Steiner, G. Tsudik, and M. Waidner, "Refinement and extension of encrypted key exchange," *ACM Operating Systems Review*, vol. 29, no. 3, pp. 22–30, 1995.

[79] D. Stinson, *Cryptography: Theory and Practice*, Chapman & Hall/CRC, second edition, 2002.

[80] H. Sun, "On the security of simple authenticated key agreement algorithm," in *Proceedings of the Management Theory Workshop'2000*, 2000.

[81] Yuh-Min Tseng, "Weakness in simple authenticated key agreement protocol," *IEE Electronics Letters*, vol. 36, no. 1, pp. 48–49, 2000.

[82] Yuh-Min Tseng, Jinn-Ke Jan, and Hung-Yu Chien, "On the security of methods for protecting password transmission," *International Journal of Informatica*, vol. 12, no. 2, pp. 1–8, 2001.

[83] Y. Tsiounis and M. Yung, "On the security of ElGamal based encryption," in *PKC'98*, pp. 117–134, Lecture Notes in Computer Science, 1998.

[84] M. J. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Transactions on Information Theory*, vol. 36, no. 3, pp. 553–558, 1990.

[85] Her-Tyan Yeh and Hung-Min Sun, "Simple authenticated key agreement protocol resisant to password guessing attacks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. 4, pp. 14–22, 2002.

[86] Xun Yi, "Efficient ID-based key agreement from Weil pairing," *Electronics Letters*, vol. 39, no. 2, pp. 206–208, 2003.