Master　Thesis

Using Auxiliary Helper Key in Key Insulated Public Key Encryption

補助鍵を用いた鍵隔離公開鍵暗号方式

指導教官　**松浦　幹太　准教授**

東京大学大学院情報理工学系研究科
電子情報学専攻

氏名　　66438　ファン　ティ　ラン　アン

提出日　　**平成 ２０ 年 ２ 月 ４ 日**

Contents

List of Figures

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

First of all, I would like to thank my supervisor, Prof. Kanta Matsuura, for encouraging and guiding me a lot in research and life in Tokyo for two years.

I am deeply indebted to Dr. Goichiro Hanaoka from National Institute of Advanced Industrial Science and Technology and Ms. Kumiko Hanaoka from NTT Docomo, Inc, whose help, stimulating suggestions and encouragement helped me in all the time of research.

Thanks is due to all members of Matsuura laboratories for their kindness and help for my research. Especially, for Mr Peng Yang, a doctor senior, and for my second year master friends, who are always willing to share with me experience in doing research, writing reports or preparing for the seminars.

I would like to express my deep and sincere gratitude to all member of "IBE-learning group" in National Institute of Advanced Industrial Science and Technology, Akibahara, for their suggestions, guidence, supports, valuable hints and encouragement from the first time I attended the group.

On a more personal note, I would like to thank my parents and my dear sister for their unconditional support. I also own thanks to my husband, for his warmly encouragement to me to do research and finish this thesis.

Abstract

To deal with the problem of secret key exposure, Dodis, Katz, Xu and Yung [11] proposed a key-insulated public key encryption (KIE), which uses the secure helper key to update the secret key more often and helps to minimize the damage overall.

We take this idea further in improving the helper key security by introducing an auxiliary helper key besides of the main helper key. The auxiliary helper key is used less than the main helper key and can help the system to reduce the damage even in case of helper key exposure.

This thesis give two different schemes of KIE with auxiliary helper key. There are trade-offs between public key length and ciphertext size, as well as between encryption algorithms and decryption algorithms in these two schemes. With these trade-offs, it is flexible to choose an efficient one to implement in reality. We also show concrete constructions with chosen-ciphertext security. The formal security proofs for all schemes are given in this thesis, based on the CBDH assumption [5] in the random oracle model.

Key words: Key Insulated Public Key Encryption, auxiliary helper key, IND-KE-CCA

Chapter 1

Introduction

1.1 Background

It is widely known that the technology era as today has invented many technological devices to provide substantial information. This is increasingly interested by the adversary. Up to now, most cryptosystems rely on possession of single totally secret entity, normally called secret key, to perform various complex tasks. It should also be remembered that security is much more under the end-users, who do not only really understand how to protect the system from the attacks but also not aware of the risks. Stealing the secret information such as credit card number or passwords, is increasing common, and it is considered to be much easier than breaking the system under the cryptography.

Dodis, Katz, Xu and Yung [11] proposed a new paradigm called *key-insulated public key cryptosystem* (KIE) which gives "resilience" to key exposure by using helper key to change or "evolve" the secret key over time. Therefore, even if the current key is exposed, the security of the system with unexposed keys is still guaranteed. The helper key is kept in a very secure place, which is only connected to the network at updating time. However, the more often we update, the more often we have to connect with the helper key. The helper key, as a result, can be taken by the adversary. Besides, if the helper key and the secret key in one time period are exposed, all the past and future secret keys will be computed easily. This suggests the need to reduce subsequent damages once the helper key is exposed.

Furthermore, in compared with forward secure encryption, the KIE is more expensive in the updating process. To reduce the cost, it is necessary to allow users to update them more easily.

Hanaoka et al. [17] proposed *parallel key-insulated public key encryption schemes* to reduce the damage. In their schemes, there are two helper keys, and these keys are used alternatively for updating of a secret key, one after the other. Therefore, a user can update his secret key more frequently, and significantly reduce the damage of exposure of a helper key. However, since these two helper keys are required to be used with the same frequency and security, it may be not convinient for a user to take care of these two keys which are stored at different places.

1.2 Contribution

In this thesis, we propose new schemes of KIE with improved helper key security. We introduce an *auxiliary helper key* besides of the main helper key. This auxiliary helper key can help the system to be able to reduce the damage once the helper key exposure happens.

We can use the main helper key and auxiliary helper key as follows: These two helper keys are kept in different devices and never used at the same time. The main helper key is used in a shorter interval of time, while the auxiliary helper key is used after a longer one. Therefore, the auxiliary helper key is used less frequently than the main helper key. For example, we can use the main helper key to update the secret key everyday and after one month, instead of main helper key, we use the auxiliary helper key to update. Since the auxiliary helper key is used less often than the main helper key, we can keep it in a safe place and do not need to take care about its security as frequently as the main helper key. It means that in our proposed scheme, we just only need to pay attention to one helper key's security, which is the same as in the original KIE developed by Dodis, Katz, Xu and Yung [11]. However, in case of one helper key exposure, as in the example given above, the damage is reduced to less than one month, which is much smaller than in the normal KIE.

We propose two different schemes with different public key length and ciphertext size. The first scheme has longer public key length compared with the second one's. However, the ciphertext size is shorter than in the second one. One more different point is that, the encryption and decryption algorithms in these two schemes are in different level of efficiency. In the first scheme, the encryption algorithm has two pairing computations, which costs much more than the only one pairing computation in the second scheme's encryption algorithm. But it happens inversely in the decryption algorithm between the two schemes. These two schemes with these trade-offs can give more flexibility to choose an efficient one to implement. In this thesis, we give both the chosen-plaintext secure schemes and chosen-ciphertext secure schemes. Especially, we give the formal security proofs for all these schemes under the CBDH assumption [5] in the random oracle model.

Unlike [17] where both helper keys need to be carefully kept and used with the same level, normally a user utilizes his main helper key to update in our proposed schemes. Hence, we only need to take care of main helper key while storing auxiliary helper key in a safe place. Although the damage of main helper key exposure is larger than

that in [17], the damage of auxiliary helper key exposure here is significantly small.

Initialization in our schemes involves providing main helper device H_{main} and auxiliary helper device H_{aux} with a main helper key mk and an auxiliary helper key ak, respectively, and the user's terminal with a *stage* 0 *user secret key* usk_0. Similarly to the original KIE, user's public encryption key pk is treated like that of an ordinary encryption scheme with regard to certification, but its lifetime is divided into stages $i = 1, 2, ..., N(= n \cdot \ell)$ with encryption in stage i performed as a function of pk, i and the plaintext, and decryption in stage i performed by the user using a *stage* i *user secret key* usk_i obtained by the following key-update process performed at the beginning of stage i:

- If $i \neq k \cdot \ell$ for $k \in \mathbb{Z}_n$, H_{main} sends to the user's terminal over a secure channel, a *stage* i *helper key* hsk_i computed as a function of mk and i,
- If $i = k \cdot \ell$ for $k \in \{1, 2, ..., n\}$, similarly to the above, H_{aux} sends hsk_i computed as a function of ak and i,

the user computes usk_i as a function of usk_{i-1} and hsk_i, and erases usk_{i-1}. Like the original KIE, our schemes also address random access key update [11] in which the user can compute an arbitrary stage user secret key (that could also be a past key). Note that it is reasonable to assume that mk and ak will not be exposed simultaneously as they can be managed separately.

The security intentions are:

1. Similarly to the original KIE, if none of the helpers is compromised, then exposure of any of user secret keys does not compromise the security of the non-exposed stages,
2. even if one of H_{main} and H_{aux} is compromised, security is still guaranteed unless other secret information is exposed as well,
3. if mk and usk_i are compromised for some i $(k \cdot \ell \leq i \leq (k+1) \cdot \ell - 1)$, then security of stages $k \cdot \ell, ..., (k+1) \cdot \ell - 1$ are compromised,
4. if ak and usk_i are compromised for some i $(k \cdot \ell + 1 \leq i \leq (k+1) \cdot \ell - 2)$, then security of stage i is compromised, and
5. if ak and usk_i are compromised for some $i(= k \cdot \ell - 1$ or $k \cdot \ell)$, then security of stages $k \cdot \ell - 1$ and $k \cdot \ell$ are compromised.

Similar to the original KIE, we can further address the case when all of the helper keys are exposed:

6. Even if both helpers H_{main} and H_{aux} are compromised, security of all stages

remain secure as long as user secret key (of any one stage) is not compromised as well.

Application. We can give an application of our proposed scheme in real life. Many of us are familiar with the following setting: a user with his portable device, such a device can be a laptop computer or a cell phone, either way, a portable device which he carries around with him daily where all his secret transactions such as decryption take place; needless to say, risk of leakage of sensitive data inside his device whether by accident or malicious intent is always an issue to him. As an application of our schemes, we can let the laptop be the main helper H_{main} where he stores the main helper key mk, and a dedicated smart card or the auxiliary helper H_{aux} in which the auxiliary helper key ak is stored and also managed securely (preferably at somewhere reasonably safe like home) when it is not in use. Laptop, i.e. H_{main}, is the one mainly being used to update his secret key just like in the original KIE, and only occasionally, his smart card, i.e. H_{aux}, and by doing so can prevent further spreading of the damage that may be caused by key exposure even if H_{main} is ever compromised. To make things more clear, let us consider the next example: daily key updating is carried out on his laptop PC, and "safety guard" updates with his smart card at the first day of each month. As you can see, even if both the master helper key mk and a user secret key (for example, 12/24 user secret key) are exposed at the same time, still, the damage is kept to the minimum by losing only the security of a month of December and the rest remain secure. Also, even if the auxiliary key ak is exposed, the harm cause is merely for a day. End users are ultimately responsible for securing information more than ever before and this is a simple and effective way for the users to give added proof to their system.

1.3 Thesis Organisation

The rest of the thesis is organised as follows: In chapter 2, we will revise some preliminaries of cryptography, included some hard problem like RSA algorithm, discrete log problem etc. We also look at elliptic curves and some mathematical problems in this curves. Security definitions are also revised in this chapter.

Chapter 3 will be devoted to the notion of key insullated public key encryption (KIE) and some related works. Chapter 4 are intended to provide the definition of model, security notions of KIE with auxiliary helper key.

The construction of KIE with auxiliary helper key will be described in chapter 5. Here the CPA secure and strong CPA secure schemes and their security proofs are

given. The CCA secure schemes and proofs are described in chapter 6.

Chapter 7 is the conclusion, where we will summerise our work within this thesis. The last part of this thesis comprises the references and publications.

Chapter 2

Preliminaries

In this chapter, we will look at some preliminaries related to cryptography. Some one-way functions such as factoring, discrete log probelm, which are easy to compute in one direction but difficult to do in the opposite direction, will be revised. Also, the definition of elliptic curves, the security assumption, as well as the security definition will be described in this section. This was summerized from [22, 23, 28].

2.1 Factoring Problem and RSA algorithm

Factoring is the act of splitting an integer into a set of smaller integers which, when multiplied together, form the original integer. Prime factorization is to spilt an integer into factors that are prime numbers. Multiplying two prime integers together is easy, but factoring the product of two prime numbers is much more difficult.

There is no efficient algorithm to factoring a number. Factoring is presumably a hard problem upon which several public-key cryptosystems are based. The most famous is based on RSA algorithm. The security of RSA algorithm depends on the factoring problem being difficult and the presence of no other types of attack.

The algorithm was publicly described in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman at MIT. The letters RSA are the initials of their surnames. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct large random prime number p and q.
2. Compute $n = pq$.
3. Compute $\varphi(n) = (p-1)(q-1)$.
4. Choose an integer e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ share no factors other than 1. e is considered as the public key exponent.
5. Compute d to satisfy $de \equiv 1(mod\ \varphi(n))$. d is kept as the private key exponent.

To encrypt a message m, the sender computes the ciphertext $C = m^e\ mod\ n$. The receiver, after receiving C, will decrypt C to get the plaintext m by computing $C^d\ mod\ n = m$.

A number n with large prime factors is more difficult to factor than a number with small prime factors. This is why the size of the modulus in the RSA algorithm determines how secure an actual use of the RSA cryptosystem is. Namely, an RSA modulus is the product of two large primes, with a larger modulus, the primes become larger and hence an attacker needs more time to factor it. The two primes, p and q, which compose the modulus, should be roughly equal length. This makes the modulus harder to factor than if one of the primes is much smaller than the other. If a 758-bit

modulus is chosen, the primes should each have length approximately 384 bits.

To choose the best size for a modulus, we must consider the security needs in our system. The larger the modulus, the greater the security, but also the slower the RSA algorithm operations. We should choose a modulus length upon consideration of the value of the protected data and how long it needs to be protected. Also, we need to care about how powerful the potential threats might be.

RSA laboratories currently recommends key sizes of 1024 bits for corporate use and 2048 bits for extremely valuable keys like the root key pair used by a certifying authority. Several recent standards specify a 1024-bit minimum for corporate use. Less valuable information may be encrypted using a 768-bit key, as such a key is still beyond the reach of all known key breaking algorithms.

It is also noticed that the key of an individual user expires after a certain time. This gives an opportunity to change keys regularly and to maintain a given level of security. Upon expiration, the user should generate a new key being sure to ascertain whether any changes in cryptanalytic skills make a move to longer key kengths appropriate. Of course, changing a key does not defend against attacks that attempt to recover messages encrypted with an old key, so key size should always be chosen according to the expected lifetime of the data.

As for the slowdown caused by increasing the key size, doubling the modulus length will, on average, increase the time required for public key operations (encryption and signature verification) by a factor of four, and increase the time taken by private key operations (decryption and signing) by a factor of eight. The reason public key operations are affected less than private key operations is that the public exponent can remain fixed while the modulus is increased, whereas the length of the private exponent increases proportionally. Key generation time would increase by a factor of 16 upon doubling the modulus, but this is a relatively infrequent operation for most users.

The RSA system is currently used in a wide variety of products, platforms, and industries around the world. It is found in many commercial software products and is planned to be in many more. The RSA algorithm is built into current operating systems by Microsoft, Apple, Sun, and Novell. In hardware, the RSA algorithm can be found in secure telephones, on Ethernet network cards, and on smart cards. In addition, the algorithm is incorporated into all of the major protocols for secure Internet communications, including S/MIME, SSL, and S/WAN. It is also used internally in many institutions, including branches of the U.S. government, major corporations, national laboratories, and universities.

2.2 Discrete Log Problem

First, we take a brief detour through group theory. Le G be a finite group. For any element $g \in G$, define $< g >= g^0, g^1, g^2, ...$ and call this the subgroup of G generated by g. Note that, since G is finite, the sequence $g^0, g^1, ...$ will eventually start repeating (cycling). In particular, since we have $g^{|G|} = 1$, the sequence can have at most $|G|$ distinct terms in it and we can write $< g >= g^0, g^1, .., g^{|G|-1}$. Of course, some G will cycle before this. If $< g >$ is the entire group G we say that g is a generator of G. If a group G has a generator, we say that G is cyclic. Note that just because a group G is cyclic does not mean that every element in G is a generator.

The discrete logarithm problem applies to mathematical structures called groups. The discrete logarithm problem is as follows: given an element g in a finite group G and another element h in G, find an integer x such that $g^x = h$.

Like the factoring problem, the discrete logarithm problem is believed to be difficult. For this reason, it has been the basis of several public-key cryptosystems. The discrete logarithm problem bears the same relation to these systems as factoring does to the RSA system: the security of these systems rests on the assumption that discrete logarithms are difficult to compute. Although the discrete logarithm problem exists in any group, when used for cryptographic purposes the group is usually Z_n^*.

The discrete logarithm problem has received much attention in recent years. The best discrete logarithm algorithms have expected running times similar to those of the best factoring algorithms. In general, the discrete logarithm in an arbitrary group of size n can be computed in running time $O(On)$ [25], though in many groups it can be done faster.

Some cryptosystems in which security depends upon the difficulty of a certain problem in G related to computing discrete logarithms are ElGamal encryption, Diffie-Hellman key exchange, Digital Signature Algorithm etc.

2.3 Elliptic Curves

Elliptic curves are described by the set of solutions to certain equations in two variables. Elliptic curves defined modulo a prime p are of central importance in public-key cryptography.

Any elliptic curve can be written as a plane algebraic curve defined by an equation

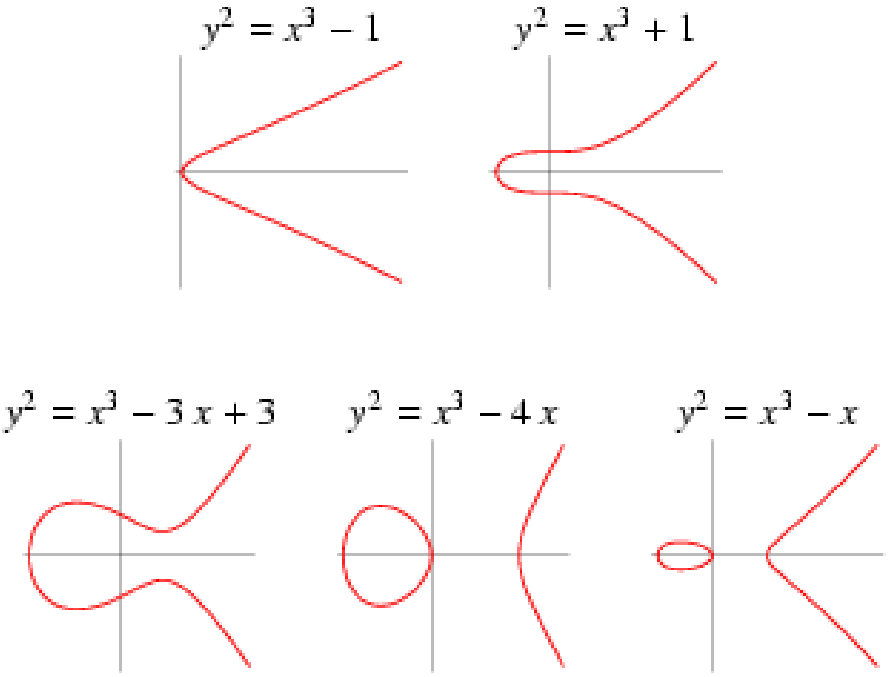

Fig. 2.1 Elliptic curves

of the form

$$y^2 = x^3 + ax + b$$

which is non-singular; that is, its graph has no cusps of self-intersections. Elliptic curves are illustrated in figure 2.1 for various values of a and b.

The set of points on such a curve can be shown to form an abelian group (with the point at infinity as identity element). If the coordinates x and y are chosen from a large finite field, the solutions form a finite abelian group.

Elliptic curves used in cryptography are typically defined over two types of finite fields: fields of odd characteristic (F_p where p is a large prime number) and fields of characteristic two (F_{2^m}). The points on an elliptic curve form an abelian group $(E(F), +)$ with 0, the distinguished point at infinity, playing the role of additive identity. Given two points M_1, M_2 on $E(F)$, there is a point, denoted by $M_1 + M_2$ on $E(F)$ and the following relations hold for all M_1, M_2, M_3

- $M_1 + M_2 = M_2 + M_1$ (commutativity)
- $(M_1 + M_2) + M_3 = M_1 + (M_2 + M_3)$ (associativity)
- $M_1 + 0 = 0 + M_1 = M_1$ (existence of an identity element)

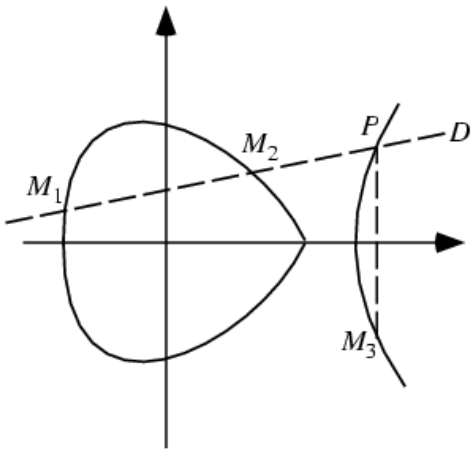

Fig. 2.2 Adding two points in elliptic curve

- There exists $(-M_1)$ such that $-M_1 + M_1 = M_1 + (-M_1) = 0$ (existence of inverses)

Suppose that two distinct points M_1 and M_2 are on an elliptic curve, and the M_1 is not $-M_2$. To add the points M_1 and M_2, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call P. The point P is reflected in the x-axis to the point M_3. The law for addition in an elliptic curve group is $M_1 + M_2 = M_3$. Adding result can be seen in the figure 2.2.

Elliptic curve cryptosystems are analogs of exsiting public key cryptosystems, in which modular arithmetic is replaced by operations defined over elliptic curves. The security of elliptic curve cryptosystems relies on the underlying hard mathematicl problems. It is proven that elliptic curve cryptosystems have no practical advantage over the RSA system, since their security is based on the same underlying problem, namely integer factorization. The situation is quite different with elliptic curve variants of discrete log based systems. The security of such systems depends on the following hard problem: Given two points M and N on an elliptic curve such that $M = kN$, find the integer k.

Presently, the algorithm for computing the discrete log problem in elliptic curves are much less efficient than those for factoring or computing conventional discrete logarithms. As the result, the discrete logarithm problem on such elliptic curve groups is believed to be more difficult than the corresponding problem in the underlying finite field. Thus keys in elliptic curve cryptography can be chosen to be much shorter for a comparable level of security. For example, elliptic curve cryptosystems with a 160-bit key offer the same security of the RSA system and discrete logarithm based systems

with a 1024-bit key. As a result, the length of the public key and private key is much shorter in elliptic curve cryptosystems.

In terms of speed, however, it is quite difficult to give a quantitative comparison, partly because of the various optimization techniques one can apply to different systems. It is perhaps fair to say the following: Elliptic curve cryptosystems are faster than the corresponding discrete logarithm based systems. Elliptic curve cryptosystems are faster than the RSA system in signing and decryption, but slower in signature verification and encryption [26].

2.4 Bilinear Maps

We give brief review of the bilinear maps. Throughout this thesis, we let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order q, and g be a generator of $\mathbb{G}_1$. A *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ satisfies the following properties:

1. Bilinearity: For all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.
3. Computability: There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}_1$.

Note that a bilinear map is symmetric since $e(g^a, g^b) = e(g^b, g^a) = e(g, g)^{ab}$.

Here, we consider two complexity assumptions related to bilinear maps: the Computational Bilinear Diffie-Hellman (CBDH) assumption and the Gap Bilinear Diffie-Hellman (GBDH) assumption.

CBDH Assumption. The CBDH problem [5] in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: given a tuple $(g, g^a, g^b, g^c) \in (\mathbb{G}_1)^4$ as input, output $e(g, g)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}_{cbdh}$ solves CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ with the probability ϵ_{cbdh} if

$$\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \epsilon_{cbdh},$$

where the probability is over the random choice of generator $g \in \mathbb{G}_1 \backslash \{1\}$, the random choice of $a, b, c \in \mathbb{Z}_q$ and random coins consumed by $\mathcal{A}_{cbdh}$.

Definition 1. We say that the $(t_{cbdh}, \epsilon_{cbdh})$-*CBDH assumption* holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if no t_{cbdh}-time algorithm has advantage of at least ϵ_{cbdh} in solving the CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$.

GBDH Assumption. The GBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: given a tuple $(g, g^a, g^b, g^c) \in (\mathbb{G}_1)^4$ as input, output $e(g, g)^{abc} \in \mathbb{G}_2$ with the help of a decision BDH oracle $\mathcal{O}$ which for given $(g, g^a, g^b, g^c, T) \in (\mathbb{G}_1)^4 \times \mathbb{G}_2$, answers "*true*" if $T = e(g, g)^{abc}$,

or "*false*" otherwise [24]. An algorithm $\mathcal{A}_{gbdh}$ solves GBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ with the probability ϵ_{gbdh} if

$$\Pr[\mathcal{A}^{\mathcal{O}}_{gbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc}] \geq \epsilon_{gbdh},$$

where the probability is over the random choice of generator $g \in \mathbb{G}_1 \backslash \{1\}$, the random choice of $a, b, c \in \mathbb{Z}_q$ and random coins consumed by $\mathcal{A}_{gbdh}$.

Definition 2. We say that the $(t_{gbdh}, \epsilon_{gbdh})$-*GBDH assumption* holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if no t_{gbdh}-time algorithm has advantage at least ϵ_{gbdh} in solving the GBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$.

2.5 Random Oracle

In cryptography, a random oracle is an oracle that responds to every query with a truly random response chosen uniformly from its output domain, except that for any specific query, it responds the same way every time it receives that query. Put another way, a random oracle is a mathematical function mapping every possible query to a random response from its output domain.

Random oracles are a mathematical abstraction used in cryptographic proofs. They are typically used when no known implementable function provides the mathematical properties required by the proof. A system that is proven secure using such a proof is described as being secure in the random oracle model, as opposed to secure in the standard model. In practice, random oracles are typically used to model cryptographic hash functions in schemes where strong randomness assumptions are needed of the hash function's output. Such a proof generally shows that a system or a protocol is secure by showing that an attacker must require impossible behavior from the oracle, or solve some mathematical problem believed hard, in order to break the protocol.

In general, we simply do not know how to construct efficient schemes which are provably secure based on standard cryptographic assumptions. Therefore, in order to have efficient schemes, it is considered to use random oracle model and to prove the security of cryptographic constructions in this model. Although a proof in this model does not guarantee security in the real world, it is still provides a useful check that our construction is not inherently flawed.

The random oracle model assumes the following:

- There is a public oracle that everyone (including all honest parties as well as the adversary) has access to.

- Random oracle implements a truly random function in the following sense: the first time anyone asks the oracle a query x, it chooses a completely random value y and returns this value. In the future, whenever someone asks query x again, the same answer y is returned. Moreover, no information about y is achieved if the query is different from x.
- In the real world, random oracles do not exist, and even if we wanted to implement a random function we would be unable to for reasonable input/output sizes. Therefore, what is done is that: design a scheme in the random oracle model and prove the security of the scheme in that model. Then when implementing, replace the random oracle by a cryptographic hash function (e.g., SHA-1 or MD5)
- If the hash function is "good" and really "garbles" its inputs, then it "acts like" a random oracle. Thus a scheme secure in the random oracle model should also be secure in the standard model when the random oracle is replaced by a "good" hash function.
- All the terms of the previous paragraph ("good", "acts like", etc.) are vague; in fact, there is no (known) way to replace a random oracle with any hash function so that the resulting scheme is provably secure in the standard model.

2.6 Security Notions

In cryptography, in order to analyze the security of the system, a security notion is defined. Normally, security notion is presented by using a game between an adversary and a challenger, in which an adversary tries to win the game. The challenger will simulate the real scheme, by calling the encrypting and decrypting oracles.

There are three normal types of attack: chosen plaintext attack, non-adaptive chosen ciphertext attack and adaptive chosen ciphertext attack. Combining these types of attack with security goal, we have the different levels of security.

There are many levels of security, and each level is defined by a different game. The normal security notions are indistinguishability under chosen plaintext attack (IND-CPA), indistinguishability under (non-adaptive) chosen ciphertext attack (IND-CCA1) and indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). IND-CPA is considered a basic requirement for most provably secure public key cryptosystems, though some schemes also provide the other two. Security under either of the latter definition implies security under the previous ones. Thus, IND-CCA2 is the strongest of these three definitions of security.

These three security properties above are in terms of indistinguishability, that is no adversary, given an encryption of a message randomly chosen from a two-element message space determined by an adversary, can identify the message choice (also means that the adversary wins in the game with the challenger) with probability significantly better than that of random guessing (1/2).

There are some other cryptographic goals besides indistinguishalibity, included semantic security, non-malleability, plaintext awareness etc.

In proving the security of one system, it is normally depended on mathematical assumption. For example, we say that one scheme is IND-CPA under the discrete logarithm assumption. It means that if the discrete logarithm is difficult for the adversary, then the adversary can not win the game with probability significantly better than that 1/2 and the system is secure. There are many assumptions used nowadays, such as decision Diffie-Hellman (DDH) assumption, computational Diffie-Hellman (CDH) assumption, etc.

It is considered to be easy to develop a scheme with the lower security level (IND-CPA) and use the Fujisaki-Okamoto transformation [15, 16] to achieve a strong security level (IND-CCA).

We will look carefully about indistinguishability under chosen plaintext attack (IND-CPA), indistinguishability under non-adaptive chosen ciphertext attack (IND-CCA1), and indistinguishability under adaptive chosen ciphertext attack (IND-CCA2).

2.6.1 Indistinguishability Under Chosen Plaintext Attack

For a probabilistic asymmetric key encryption algorithm, indistinguishability under chosen plaintext attack (IND-CPA) is defined by the following game between an adversary and a challenger. For schemes based on computational security, the adversary is modeled by a probabilistic polynomial time Turing machine, meaning that it must complete the game and output a guess within a polynomial number of time steps. In this definition $E(pk, m)$ represents the encryption of a message m under the key pk. sk is the secret key:

1. The challenger generates a key pair pk, sk based on some security parameter k (e.g., a key size in bits), and publishes pk to the adversary. The challenger retains sk.
2. The adversary may perform any number of encryptions or other operations.
3. Eventually, the adversary submits two distinct chosen plaintexts m_0, m_1 to the

challenger.

4. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the challenge ciphertext $C = E(pk, m_b)$ back to the adversary.
5. The adversary is free to perform any number of additional computations or encryptions. Finally, it outputs a guess for the value of b.

A cryptosystem is indistinguishable under chosen plaintext attack if every probabilistic polynomial time adversary has only a negligible "advantage" over random guessing. An adversary is said to have a negligible "advantage" if it wins the above game with probability $(1/2) + \epsilon(k)$, where $\epsilon(k)$ is a negligible function in the security parameter k, that is for every (nonzero) polynomial function $poly()$ there exists k_0 such that $|\epsilon(k)| < |1/poly(k)|$ for all $k > k_0$.

Although the adversary knows M0, M1 and PK, the probabilistic nature of E means that the encryption of Mb will be only one of many valid ciphertexts, and therefore encrypting M0, M1 and comparing the resulting ciphertexts with the challenge ciphertext does not afford any advantage to the adversary.

While the above definition is specific to an asymmetric key cryptosystem, it can be adapted to the symmetric case by replacing the public key encryption function with an "encryption oracle", which retains the secret encryption key and encrypts arbitrary ciphertexts at the adversary's request.

2.6.2 Indistinguishability Under Chosen Ciphertext Attack

Indistinguishability under non-adaptive and adaptive Chosen Ciphertext Attack (IND-CCA1, IND-CCA2) uses a definition similar to that of IND-CPA. However, in addition to the public key (or encryption oracle, in the symmetric case), the adversary is given access to a "decryption oracle" which decrypts arbitrary ciphertexts at the adversary's request, returning the plaintext. In the non-adaptive definition, the adversary is allowed to query this oracle only up until it receives the challenge ciphertext. In the adaptive definition, the adversary may continue to query the decryption oracle even after it has received a challenge ciphertext, with the caveat that it may not pass the challenge ciphertext for decryption (otherwise, the definition would be trivial).

1. The challenger generates a key pair pk, sk based on some security parameter k (e.g., a key size in bits), and publishes pk to the adversary. The challenger retains sk.
2. The adversary may perform any number of encryptions, calls to the decryption

oracle based on arbitrary ciphertexts, or other operations.

3. Eventually, the adversary submits two distinct chosen plaintexts m_0, m_1 to the challenger.
4. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random, and sends the "challenge" ciphertext $C = E(pk, m_b)$ back to the adversary.
5. The adversary is free to perform any number of additional computations or encryptions.
   - In the non-adaptive case (IND-CCA1), the adversary may not make further calls to the decryption oracle.
   - In the adaptive case (IND-CCA2), the adversary may make further calls to the decryption oracle, but may not submit the challenge ciphertext C.
6. Finally, the adversary outputs a guess for the value of b.

A scheme is IND-CCA1/IND-CCA2 secure if no adversary has a non-negligible advantage in winning the above game.

Chapter 3

Key Insulated Public Key Encryption (KIE)

KIE allows user to update the secret key overtime and it reduces the damage once the secret key is exposed. This section will describe about the detail of KIE.

3.1 Model

In a (t, N)-key-insulated scheme, an adversary who compromises the insecure device and obtains secret keys for up to t periods of his choice is unable to violate the security of the cryptosystem for any of the remaining $N - t$ periods. The model of KIE is the following. User first registers a single public key PK. A master secret key SK^* is stored on a device which is physically secure and hence resistant to compromise. However, decryption is done on an insecure device for which exposure is expected to be a problem. Protocol's life time is divided into distinct periods 1,..,N (these time periods' length are suposed to be equal for simplicity).

Suppose user is in the period $i-1$ and having secret key SK_{i-1}. When the user wants to update his secret key of the next period, he first connects with the secret device (sometimes called helper device) to get the update information $SK_i^{'}$. Using the update information $SK_i^{'}$ and his own secret key at the moment, he computes the new secret key SK_i. He then uses this key in decrypting the message encrypted at this period. The public key PK is kept unchanged during all the periods.

The detail of the model of the key-insulated public key encryption is as follows:

Definition 3. *A key-updating (public-key) encryption scheme is a 5-tuple of poly-time algorithms* (G, U^*, U, ϵ, D) *such that:*

- G*, the key generation algorithm: takes as input a security parameters* 1^k *and the total number of time periods* N*. It returns a public key* PK*, a master key* SK^* *and an initial key* SK_0*.*
- U^**, the device key-update algorithm: takes as input an index* i *for a time period* $(1 \leq i \leq N)$ *and the master key* SK^**. It returns the partial secret key* $SK_i^{'}$ *for time period* i*.*
- U*, the user key-update algorithm: takes as input an index* i*, secret key* SK_{i-1} *and a partial secret key* $SK_i^{'}$*. It returns the partial secret key secret key* SK_i *for time period* i *and erase* $SK_{i-1}, SK_i^{'}$
- ϵ*, the encryption algorithm: takes as input a public-key* PK*, a time* i *and a message* M*. It returns a ciphertext* $< i, C >$*.*
- D*, the decryption algorithm: takes as input a secret key* SK_i *and a ciphertext* $< i, C >$*. It returns a message* M *or the special symbol* $\perp$*.*

It is required that for all messages M, $D_{SK_i}(\epsilon_{PK}(i, M)) = M$

A key-updating ecnryption scheme is used as the following. A user begins by generating $(PK, SK^*, sk_0) \leftarrow G(1^k, N)$, registering PK in a central location, storing SK^* on a physically-secure device, and storing SK_0 himself. At the beginning of time period i, the user requests $SK_i^{'} = U^*(i, SK^*)$ from the secure device. Using $SK_i^{'}$ and SK_{i-1}, the user may compute $SK_i = U(i, SK_{i-1}, SK_i^{'})$. This key may be used to decrypt message sent during time period i without further access to the device. After computation of SK_i, the user must erase $SK_i^{'}$ and SK_{i-1}.

One important feature of key updates is random-access, which allows user to update from period j to i in "one shot". The definition above implicitly fixes $j = i - 1$.

3.2 Construction

A generic construction of the KIE is written in the paper [11]. Here we will give the scheme which is proved to be secure under the decision Diffie-Hellman assumption.

- $G(1^k)$: $(g, h, k) \longleftarrow Gen(1^k)$;
  $x_0^*, y_0^*, ..., x_t^*, y_t^* \longleftarrow Z_q$
  $z_0^* := g^{x_0^*} h^{y_0^*}, ..., z_t^* := g^{x_t^*} h^{y_t^*}$
  $PK = (g, h, q, z_0^*, ..., z_t^*)$
  $SK^* = (x_1^*, y_1^*, ..., x_t^*, y_t^*)$; $SK_0 = (x_0^*, y_0^*)$
  return PK, SK^*, SK_0
- $U^*(i, SK_* = (x_1^*, y_1^*, ..., x_t^*, y_t^*))$:
  $x_i^{'} = \sum_{j=1}^{t} x_j^*(i^j - (i-1)^j)$
  $y_i^{'} = \sum_{j=1}^{t} y_j^*(i^j - (i-1)^j)$
  return $SK_i^{'} = (x_i^{'}, y_i^{'})$
- $U(i, SK_{i-1} = (x_{i-1}, y_{i-1}), SK_i^{'} = (x_i^{'}, y_i^{'}))$:
  $x_i = x_{i-1} + x_i^{'}$
  $y_i = y_{i-1} + y_i^{'}$
  return $SK_i = (x_i, y_i)$
- $\epsilon_(g, h, w, z_0^*, ..., z_y^*)(i, M)$:
  $z_i = \prod_{j=0}^{t} (z_j^*)^{i^j}$
  $r \leftarrow Z_q$
  $C := (g^r, h^r, z_i^r M)$ return (i, C)
- $D_{(x_i, y_i)}((i, C = (u, v, w)))$:
  $M := w / u^{x_i} v^{y_i}$

return M

The above scheme is secure based on DDH assumption in standard model. There is also another way to build a KIE scheme by using the Weil pairing which is used in the [5]. The scheme in [5] is considered to be more efficient and therefore using the weil pairing is much better. Therefore, in some next researchs, there are some works that try to apply pairing and KIE in the same scheme [7, 17, 18]. However, building a scheme with pairing in the standard model is still an open problem.

3.3 Security

Let first see the security attack model of this scheme. There are three types of exposures that need to protect against: (1) ordinary key exposure, which models compromise of the insecure storage (i.e, leakage of SK_i); (2) key-update exposure, which models compromise of the insecure device during the key-updating step (i.e., leakage of SK_{i-1} and $SK_i^{'}$; and (3) master key exposure, which models compromise of the physically-secure device (i.e., leakage of SK^*). It is needed to prove the security of KIE under these three types of exposures.

The adversary is allowed to access a key exposure oracle, which, on in put i, returns the temporary secret key SK_i. Moreover, the adversary can access to a decryption oracle that, on input (i, C), computes $D_{SK_i}((i, C))$. This models a chosen-ciphertext attack by the adversary.

The detail of security proof can be seen in the paper [11].

3.4 Related Works

In this section, we look generally about some related works with KIE that have been researched in recent years.

Strongly key-insulated security

Developed from the KIE, strongly key-insulated public key encryption (sKIE) enhances the security of the system in case of helper key exposure. In this scheme, the security of the system is guaranteed even if the attacker can get the helper key, as long as he can not get one of secret keys. It is not difficult to change from the KIE scheme to sKIE scheme [11].

Key-insulated encryption with optimal threshold

The work of Dodis et al. is then considered in terms of realization further towards practice by presenting simple new schemes that provide benefits in terms of scalability, performance and security. Bellare and Palacio [7] proposed a simple, practical, scalable scheme that achieves the best possible security in their framework, based on the Boneh-Franklin indentity-based encryption.

Relation with identity based encryption

In 1984, Shamir asked for a public key encryption in which the public key can be an arbitrary string [27]. This is an original motivation for identity-based encryption. This idea then formalized by Boneh and Franklin in 2001 [5]. The full paper is in [6]. In these papers, identity based encryption is built by using Weil pairing computation. Cock also gave an identity based encryption scheme by using quadratic residue in 2001 [9].

According to the work of Boneh and Franklin, the arbitrary string is considered to be the user's identity, such as email address. When Alice wants to send Bob a message, she can use Bob's email address to encrypt the message without receiving the public key authentication. Bob authenticates himself to the PkG in the same way he would authenticate himself to a CA and obtains his private key from the PKG. Bob can then decrypt the message sent by Alice.

It is easy to see that an ID-based encryption scheme may be converted an (N-1,N)-key-insulated encryption scheme by viewing the period number as an "identity" and having the physically-secure device implement the trusted center. The converse is also true. A (t, N)-key-insulated encryption scheme with a fully trusted device may be viewed as a relaxation of ID-based encryption, where it is not insisted on $t = N - 1$. Even though the model of ID-based encryption assumes a fully trusted center, it was observered that the particular scheme in [5] -when viewed as an $(N-1, N)-KIE$- can be very easily modified so that the secure device no longer needs to be trusted. This almost immediately gives a fully secure KIE. However, these schemes are developed under random oracle and there is still a problem of how to build a ID-based scheme in a standard oracle. Some works have been researched without random oracle [3, 4], but the weaker security notion, called "selective ID secure" was used instead of the normal one. Water also gave a work of identity based encryption without random oracle [29].

Forward-secure encryption

Forward-secure encryption (FSE) is also one kind of key evolution. First idea was

given by [1] and the first efficient forward-secure public key encryption is provided by Canetti, Halevi and Katz [10]. This scheme builds on the hierarchical identity based encryption of Gentry and Silverberg [20], which in turn is based on Boneh and Franklin's identity-based encryption [5]. Besides, the forward-secure signature scheme was also presented by Bellare and Miner in 1999 [8] and by Abdalla and Reyzi in 2000 [2].

The main idea of forward-secure encryption is that changing the secret key during different time periods, while the public key is unchanged. It is different from KIE that the updating process is done without the use of a secret device.

Trade-off relations between the KIE and FSE are discussed in the paper [19]. According to this paper, there is a trade-off between lifespan of using the key certificate and simplicity of key updating process. While the updating process in FSE is very simple, it is very costly in KIE as the need of connecting with the helper device. However, with the help of helper device, once the secret key is exposed at one time period, it has no effect on the other time periods' keys. It is much different from the FSE, because since one key is exposed, the security of the system has been broken from that point of time.

Moreover, the FSE allows the past messages to be kept secret even in case of exposure, while the KIE does not. When the adversary can get the helper key and a secret key at the same time, he can compute all the messages sent at any time.

Last but not least, random-access key updates are impossible to achieve in the forward-security model.

Instrusion-resilient public key encryption

Coming next after the forward security and key insulation is intrusion-resilience, which was proposed as a means of mitigating the harmful effects that key exposure can have [13, 14]. Like the forward-secure scheme and key-insulated scheme, the public key is unchanged while the secret key is changed time after time.

Forward-secure schemes are advantageous in that the user is self-sufficient and need not interact with any other device. On the other hand, the security provided by key-insulated and intrusion-resilient schemes is better and these schemes might therefore be used when interacting with a server is feasible and does not represent a serious drawback. Finally, although the intrusion-resilient model offers stronger security guarantees than the key-insulated model, it is noted that solutions for the latter are much more efficient. The choice of which type of scheme to use therefore depends heavily on an assumption about the physical security of the server.

Hierarchical strongly key-insulated encryption

Hierarchical strongly key-insulated encryption is developed by Hanaoka et al. [18] in 2003. This scheme enhances the security of sKIPE by hierarchically structuring the helper key with added identity-based property.

In this scheme, like the original KIE, a private device, which stores the helper key, is not connected to the network except at each fixed time period when the decryption key is updated. When updating, the user connects with the private device and uses the helper key which is stored in it. All secret operations are done by the user alone. The private device is divided into multiple levels forming a hierarchical structure to improve its security.

Their proposed schemes are constructed by extending the hierarchical identity-based encryption schemes (HIBE) [20]. As they proved in the paper, straightforward extension of HIBE will be completely vulnerable for their attack model. They proposed two secure constructions of IBE that can renew and update the decryption key non-interactively: (1) a generic construction based on any HIBE, and (2) a specific construction based on Gentry-Silverberg HIBE. Although being more efficient than the generic scheme, the specific scheme is based on the bilinear Diffie-Hellman assumption and flexibility may become a concern when designing new constructions in terms of security.

Parallel key-insulated public key encryption

This is the scheme developed by Hanaoka et al. [17] in 2006. This work enhances the security of the system by using two parallel helper devices, which are used alternately to update the secret key. These helper devices are never used at the same time and never stored at the same places.

The security of this scheme is as follows:

- If none of the helper keys is compromised, similar to the original KIE, exposure of any of user secret keys does not compromise the security of the non-exposed stages.
- Even if one of helper keys is compromised in addition to the exposure of any of user secret keys, it still does not compromise the security of the non exposed stages except for the ones whose corresponding user secret keys can be trivially determined from the exposed keys.
- For the strong version, even if both helper keys are compromised, security of all stages remain secure as long as user secret key is not compromised in addition

to the helper keys.

Therefore, even in case of one helper device is attacked, another can be considered to be safe. Moreover, the damage after being attacked is reduced to the minimum.

The detail scheme is built based on the work of Boney and Franklin [5], thus the efficiency of this scheme can be said to be considerable with the scheme in [5]

Key-insulated signature

Besides of encryption schemes, key-insulation is also useful in developing the signature scheme, which is very important in transactions and e-commerce. In this global communication environment, signature computation will be frequently performed on a relatively insecure device (e.g., a mobile phone) that can not be trusted to completely maintain the secrecy of the secret key. In the effort to deal with the problem, key-insulated signature is a good solution. This work is done in [12].

Chapter 4

Model and Security Definition of KwAH

In this chapter, we will give the model of KIE with an auxiliary helper key (KwAH) and the security notion. We follow by showing some of the characteristics of bilinear maps. We then briefly review the related computational assumptions.

4.1 Model: KIE with an Auxiliary Helper Key(KwAH)

A KwAH scheme $\mathcal{E}$ consists of five efficient algorithms (**KeyGen, Δ-Gen, Update, Encrypt, Decrypt**).

KeyGen: Takes a security parameter k and returns mk, ak, usk_0 and pk. Public key pk includes a description of finite message space $\mathcal{M}$, and description of finite ciphertext space $\mathcal{C}$.

Δ-Gen: Takes as inputs, mk and i, and returns stage i helper key hsk_i if $\ell \not| i$, or $\perp$ otherwise, and takes as inputs, ak and i, and returns stage i helper key hsk_i if $\ell | i$, or $\perp$ otherwise

Update: Takes as inputs, usk_{i-1}, hsk_i and i, and returns stage i user secret key usk_i.

Encrypt: Takes as inputs, pk, i and $M \in \mathcal{M}$, and returns ciphertext $C \in \mathcal{C}$.

Decrypt: Takes as inputs, pk, usk_i and $C \in \mathcal{C}$, and returns $M \in \mathcal{M}$ or $\perp$.

These algorithms must satisfy the standard consistency constraint, namely,

$$\forall i \in \{1, 2, ..., N\},\ \forall M \in \mathcal{M} : \mathbf{Decrypt}(pk, usk_i, C) = M \quad \text{where } C = \mathbf{Encrypt}(pk, i, M).$$

4.2 Security Definition

Here, we define the notion of semantic security for KwAH. This is based on the security definition in the original KIE [11, 7]. It should be noticed that the definition in [7] looks simpler than in [11] but they are essentially the same.

We say that a KwAH scheme $\mathcal{E}$ is *semantically secure against an adaptive chosen ciphertext attack under an adaptive chosen key exposure attack* (IND-KE-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following IND-KE-CCA game:

Setup: The challenger takes a security parameter k and runs the **KeyGen** algorithm. He gives the adversary the public key pk and keeps usk_0, mk and ak to himself.

Phase 1: The adversary issues several queries $q_1, \cdots, q_\rho$ where each of the queries q_i is one of:

- Exposure query $\langle j, \texttt{class} \rangle$: If `class` = "user", the challenger responds by

running the algorithms **Δ-Gen** and **Update** to generate usk_j and sends it to the adversary. If `class` = "main helper" or "auxiliary helper", the challenger sends mk or ak to the adversary, respectively.

- Decryption query $\langle j, C\rangle$: The challenger responds by running the algorithms **Δ-Gen** and **Update** to generate usk_j. He then runs **Decrypt** to decrypt the ciphertext C using usk_j and sends the result to the adversary.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over, she outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and $j^* \in \{1, 2, ..., N\}$ on which she wishes to be challenged. The challenger picks a random bit $\beta \in \{0, 1\}$ and sets $C^* = \textbf{Encrypt}(pk, j^*, M_\beta)$. The challenger sends C^* as the challenge to the adversary.

Phase 2: The adversary issues additional queries $q_{\rho+1}, \cdots, q_{max}$ where each of the queries is one of:

- Exposure query $\langle j, \texttt{class}\rangle$: Challenger responds as in Phase 1.
- Decryption query $\langle j, C\rangle$: Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

Guess: Finally, the adversary outputs her guess $\beta' \in \{0, 1\}$. She wins the game if $\beta' = \beta$ and

1. $\langle j^*, C^*\rangle$ does not appear in Decryption queries,
2. $\langle j^*$, "user"$\rangle$ does not appear in Exposure queries,
3. both $\langle j$, "user"$\rangle$, such that $m \cdot \ell \leq j^* \leq (m+1) \cdot \ell - 1$ and $m \cdot \ell \leq j \leq (m+1) \cdot \ell - 1$ for some m $(0 \leq m \leq n-1)$, and $\langle \cdot$, "main helper"$\rangle$ do not simultaneously appear in Exposure queries,
4. both $\langle j$, "user"$\rangle$, such that $j^* = (m+1) \cdot \ell - 1$ or $(m+1) \cdot \ell$ and $j = (m+1) \cdot \ell - 1$ or $(m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, and $\langle \cdot$, "auxiliary helper"$\rangle$ do not simultaneously appear in Exposure queries,
5. both $\langle \cdot$, "main helper"$\rangle$ and $\langle \cdot$, "auxiliary helper"$\rangle$ do not simultaneously appear in Exposure queries.

We refer to such an adversary $\mathcal{A}$ as an IND-KE-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ as:

$$Adv_{\mathcal{E},\mathcal{A}} = \Pr[\beta' = \beta] - 1/2.$$

The probability is over the random bits used by the challenger and the adversary.

Definition 4. We say that a KwAH scheme $\mathcal{E}$ is (t, ϵ)-*adaptive chosen ciphertext*

secure under adaptive chosen key exposure attacks if for any t-time IND-KE-CCA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is IND-KE-CCA secure.

As usual, we can define chosen plaintext security similarly to the game above except that the adversary is not allowed to issue any Decryption queries. The adversary still can adaptively issue Exposure queries. We call this adversary IND-KE-CPA adversary.

Definition 5. We say that a KwAH scheme $\mathcal{E}$ is (t, ϵ)-*adaptive chosen plaintext secure under adaptive chosen key exposure attacks* if for any t-time IND-KE-CPA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is *IND-KE-CPA secure.*

IND-KE-CCA is already a strong security notion, but its security can be enhanced further to cover the compromise of both the helper keys. Concretely, as a constraint on the above adversary's Exposure query, we can modify 5. so that:

> 5′. $\langle \cdot$, "main helper"$\rangle$, $\langle \cdot$, "auxiliary helper"$\rangle$, and $\langle j$, "user"$\rangle$ do not simultaneously appear in Exposure queries for any $j \in \{1, 2, ..., N\}$.

Such modification allows the adversary $\mathcal{A}$ to obtain both mk and ak if $\mathcal{A}$ doesn't ask any of user secret keys. Let this adversary be a *strong IND-KE-CCA* adversary.

Definition 6. We say that a KwAH scheme $\mathcal{E}$ is (t, ϵ)-*adaptive chosen ciphertext secure under strongly adaptive chosen key exposure attacks* if for any t-time strong IND-KE-CCA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is *strongly IND-KE-CCA secure.*

Similarly, we can define *strong IND-KE-CPA adversary*, and here as well, she is not allowed to issue any Decryption queries.

Definition 7. We say that a KwAH scheme $\mathcal{E}$ is (t, ϵ)-*adaptive chosen plaintext secure under strongly adaptive chosen key exposure attacks* if for any t-time strong IND-KE-CPA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is *strongly IND-KE-CPA secure.*

A Remark on the Security Notion: Exposure of the Helper Keys. In the discussion we had so far, it may seem like we may have overlooked the exposure of stage i helper key, but actually, we haven't. It is obvious that if hsk_i can be computed from usk_{i-1} and usk_i for any stage i, then exposure of hsk_i can be emulated by using the responses to the Exposure queries. So, the security definition so far given is

sufficient as it is even against exposure of stage i helper keys for any i, if we assume that such property holds. As a matter of fact, all of our constructions satisfy this property.

Chapter 5

KIE with Auxiliary Helper Key

In this chapter, we propose two different KwAH schemes and prove their security under the CBDH assumption in the random oracle model. The trade-offs between public key length and ciphertext size, as well as the difference between the encryption and decryption algorithms' efficiency in these schemes give flexibility to choose an efficient one to implement.

The schemes in [5] are considered to be efficient by many researchers. Our schemes are reasonably efficient since efficiency of our KwAH schemes can said to be comparable to [5]. In our schemes, we let $N = O(\mathsf{poly}(k))$.

The results in this chapter were written in publication P1, P2 and P3.

5.1 IND-KE-CPA Schemes

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q of size k, and g be a generator of $\mathbb{G}_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. Let G, H be cryptographic hash functions $G : \mathbb{G}_2 \to \{0,1\}^n$ for some n, $H : \{0,1\}^* \to \mathbb{G}_1$, respectively.

5.1.1 KwAH1: IND-KE-CPA Construction

The first IND-KE-CPA scheme KwAH1 consists of the following algorithms:

KeyGen: Given a security parameter k, **KeyGen** algorithm:

1. generates $\mathbb{G}_1$, $\mathbb{G}_2$, g and e.
2. picks $s_1, s_2 \in \mathbb{Z}_q^*$ uniformly at random, and sets $h_1 = g^{s_1}$ and $h_2 = g^{s_2}$,
3. chooses cryptographic hash functions G and H,
4. computes $d_{-1} = H(-1)^{s_1}$ and $d_0 = H(0)^{s_2}$, and
5. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H\rangle$, $mk = s_1$, $ak = s_2$ and $usk_0 = d_{-1} \cdot d_0$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1^* \times \{0,1\}^n$.

Δ-Gen: For given mk and $i \in \{1, 2, ..., N\}$, **Δ-Gen** algorithm:

1. outputs $\perp$ if $i = 0 \bmod \ell$,
2. outputs $hsk_i = H(i{-}1)^{-s_1} \cdot H(i)^{s_1}$ if $i \neq m{\cdot}\ell{+}1$ for some m $(0 \leq m \leq n{-}1)$,
3. outputs $hsk_i = H(i{-}2)^{-s_1} \cdot H(i)^{s_1}$ if $i = m{\cdot}\ell{+}1$ for some m $(0 \leq m \leq n{-}1)$.

For given ak and $i \in \{1, 2, ..., N\}$, **Δ-Gen** algorithm:

1. outputs $\perp$ if $i \neq 0 \bmod \ell$,
2. outputs $hsk_i = H(i - \ell)^{-s_2} \cdot H(i)^{s_2}$ otherwise.

Update: For given usk_{i-1}, hsk_i and i, **Update** algorithm:

1. computes $usk_i = usk_{i-1} \cdot hsk_i$,

2. deletes usk_{i-1} and hsk_i, and
3. outputs usk_i.

Encrypt: For given pk, i, and a message $M \in \{0,1\}^n$, assuming that $m \cdot \ell + 1 \leq i \leq (m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, **Encrypt** algorithm:

1. chooses random $r \in \mathbb{Z}_q^*$,
2. computes $W = (e(h_1, H(i)) \cdot e(h_2, H(m \cdot \ell)))^r$ if $i \neq (m+1) \cdot \ell$,
3. computes $W = (e(h_1, H(i-1)) \cdot e(h_2, H((m + 1) \cdot \ell)))^r$ if $i = (m+1) \cdot \ell$,
4. sets $C = \langle i,\ g^r,\ G(W) \oplus M \rangle$, and
5. outputs C as a ciphertext.

Decrypt: For given pk, usk_i and $C = \langle i, c_0, c_1 \rangle$, **Decrypt** algorithm:

1. computes $W' = e(c_0, usk_i)$,
2. computes $M' = c_1 \oplus G(W')$, and
3. outputs M' as a plaintext.

We show the correctness of this scheme. Through **Δ-Gen** and **Update**, the secret key usk_i can be computed to be

$$usk_i = \begin{cases} H(i)^{s_1} \cdot H(m \cdot \ell)^{s_2} & (\text{if } i \neq (m+1) \cdot \ell), \\ H(i-1)^{s_1} \cdot H((m+1) \cdot \ell)^{s_2} & (\text{if } i = (m+1) \cdot \ell) \end{cases}$$

Therefore,

$$\begin{aligned} e(c_0, usk_i) &= \begin{cases} e(g^r, H(i)^{s_1} \cdot H(m \cdot \ell)^{s_2}) & (\text{if } i \neq (m+1) \cdot \ell), \\ e(g^r, H(i-1)^{s_1} \cdot H((m+1) \cdot \ell)^{s_2}) & (\text{if } i = (m+1) \cdot \ell) \end{cases} \\ &= \begin{cases} e(g, H(i)^{s_1} \cdot H(m \cdot \ell)^{s_2})^r & (\text{if } i \neq (m+1) \cdot \ell), \\ e(g, H(i-1)^{s_1} \cdot H((m+1) \cdot \ell)^{s_2})^r & (\text{if } i = (m+1) \cdot \ell) \end{cases} \\ &= \begin{cases} (e(g^{s_1}, H(i)) \cdot e(g^{s_2}, H(m \cdot \ell)))^r & (\text{if } i \neq (m+1) \cdot \ell), \\ (e(g^{s_1}, H(i-1)) \cdot e(g^{s_2}, H((m+1) \cdot \ell)))^r & (\text{if } i = (m+1) \cdot \ell) \end{cases} \end{aligned}$$

$$e(c_0, usk_i) = \begin{cases} (e(h_1, H(i)) \cdot e(h_2, H(m \cdot \ell)))^r & (\text{if } i \neq (m+1) \cdot \ell), \\ (e(h_1, H(i-1)) \cdot e(h_2, H((m+1) \cdot \ell)))^r & (\text{if } i = (m+1) \cdot \ell) \end{cases}$$

This means that, applying decryption after encryption produces the original message M as required.

5.1.2 KwAH2: IND-KE-CPA Construction

The second IND-KE-CPA scheme KwAH2 consists of the following algorithms:

KeyGen: Same as that of KwAH1 except that it:

2. Picks $s_1, s_2 \in \mathbb{Z}_q^*$ uniformly at random, and sets $h = g^{s_1+s_2}$,
5. Outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h, G, H\rangle$, $mk = s_1,\ ak = s_2,\ usk' = g^{s_2}$ and $usk'_0 = d_{-1} \cdot d_0$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1^* \times \{0,1\}^n$.

Δ-Gen: Same as that of KwAH1 scheme.

Update: For given $usk_{i-1} = (usk'_{i-1},\ usk')$, hsk_i and i, **Update** algorithm:

1. computes $usk'_i = usk'_{i-1} \cdot hsk_i$,
2. deletes usk'_{i-1} and hsk_i, and
3. outputs $usk_i = (usk'_i,\ usk')$.

Encrypt: For given pk, i, and a message $M \in \{0,1\}^n$, assuming that $m \cdot \ell + 1 \leq i \leq (m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, **Encrypt** algorithm:

1. chooses random $r \in \mathbb{Z}_q^*$,
2. computes $W = (e(h, H(i)))^r$
3. computes $h_3 = (H(i) \cdot H(m \cdot \ell)^{-1})^r$ if $i \neq (m+1) \cdot \ell$,
4. computes $h_3 = (H(i) \cdot H(i-1)^{-1})^r$ if $i = (m+1) \cdot \ell$,
5. sets $C = \langle i,\ h_3,\ g^r,\ G(W) \oplus M\rangle$, and
6. outputs C as a ciphertext.

Decrypt: For given pk, usk'_i, usk' and $C = \langle i, h_3, c_0, c_1\rangle$, assuming that $m \cdot \ell + 1 \leq i \leq (m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, **Decrypt** algorithm:

1. computes $W' = e(c_0, usk'_i) \cdot e(usk', h_3)$ if $i \neq (m+1) \cdot \ell$,
2. computes $W' = e(c_0, usk'_i) \cdot e(h \cdot usk'^{-1}, h_3)$ if $i = (m+1) \cdot \ell$,
3. computes $M' = c_1 \oplus G(W')$, and
4. outputs M' as a plaintext.

The correctness of KwAH2 can be shown in the same way as that of KwAH1. Through **Δ-Gen** and **Update**, the secret key usk_i can be computed to be

$$usk'_i = \begin{cases} H(i)^{s_1} \cdot H(m \cdot \ell)^{s_2} & (\text{if } i \neq (m+1) \cdot \ell), \\ H(i-1)^{s_1} \cdot H(i)^{s_2} & (\text{if } i = (m+1) \cdot \ell) \end{cases}$$

Therefore,

$$
\begin{aligned}
e(h, H(i))^r &= e(g^{s_1+s_2}, H(i))^r \\
&= e(g^r, H(i)^{s_1+s_2}) \\
&= \begin{cases} e(g^r, H(i)^{s_1} \cdot H(m \cdot \ell)^{s_2}) \cdot e(g^{s_2}, (H(i) \cdot H(m \cdot \ell)^{-1})^r) \\ \qquad\qquad\qquad\qquad\qquad\qquad (\text{if } i \neq (m+1) \cdot \ell), \\ e(g^r, H(i-1)^{s_1} \cdot H(i)^{s_2}) \cdot e(g^{s_1}, (H(i) \cdot H(i-1)^{-1})^r) \\ \qquad\qquad\qquad\qquad\qquad\qquad (\text{if } i = (m+1) \cdot \ell) \end{cases} \\
&= \begin{cases} e(c_0, usk'_i) \cdot e(usk', h_3) & (\text{if } i \neq (m+1) \cdot \ell), \\ e(c_0, usk'_i) \cdot e(h \cdot usk'^{-1}, h_3) & (\text{if } i = (m+1) \cdot \ell) \end{cases}
\end{aligned}
$$

This fact shows that decryption after encryption will produce the message as required.

5.1.3 Trade-offs between Two Schemes

Public Key Size vs. Ciphertext Size. As can be seen from the KwAH2 above, usk' is used only in **Decrypt** algorithm and does not play any role in the **Encrypt** algorithm. Moreover, even in case that the attacker can get usk', he does not have any more advantage in computing the plaintext itself, as it is difficult to get the usk'_i from the public key, ciphertext and usk' (CBDH assumption). Therefore, in fact, the part usk' of the private key usk_i does not need to be either kept secret or given publicly. *We can save it in a place and do not need to take care about its security.* It also means that the size of private key in KwAH2 can be considered to be the same as in the KwAH1 scheme. However, if we let usk' be a part of private key, then we can see the trade-off between private key size and public key size of these two schemes.

For example, in case of the application we showed in the introduction, we can save usk' in the computer and write it in the smart card, so whenever we have the secret key updated, we can get the information of usk' very conveniently. Another way is to keep this information in a card (any card is possible, as long as we have that card when updating). Of course, we do not need to worry if the card is exposed to the attackers. Therefore, it is very flexible for the users in keeping the information of usk'.

The size of private key in KwAH2 is still the same with KwAH1's, while the size of public key is shorter. That makes it cost less when sending the information about public key to the user after doing **KeyGen** algorithm. However, the size of ciphertext in KwAH2 is longer than in KwAH1, so we can say that there is a trade-off between

ciphertext size and public key length.

Cost for Encrytion vs. Cost for Decryption. It can be seen from the above schemes that there is a difference between the efficiency of **Encrypt** and **Decrypt** algorithms in these two schemes. By looking in the details we can see that the encryption algorithm in KwAH1 is not as efficient as in the KwAH2, while the decryption algorithm is different. Obviously, it takes more time to compute two bilinear maps than one bilinear map, so in the KwAH1 schemes the sender needs to do much more than the receiver. It means that it costs more for the sender in encrypting the plaintext than the receiver in decrypting the ciphertext. In case of KwAH2, things are different, because the encryption algorithm is much easier than the decryption algorithm. Thus, there is another trade-off between the efficiency of encryption algorithm and decryption algorithm in these two schemes. Depending on the computational ability of receiver and sender, the system designers can choose which efficient scheme to use.

5.1.4 Security and Proofs

Now, we prove that KwAH1 and KwAH2 are IND-KE-CPA under the CBDH assumption in the random oracle model. Here, we briefly mention the technical hurdle for the security proof. Since we consider adaptively chosen key exposure adversary, the simulator has to deal with various types of key exposures, i.e. mixture of mk, ak, and user secret keys, and moreover, it does not know the adversary's strategy before the simulation. Nevertheless, the simulator must provide successful simulation. This makes the proof complicated.

Theorem 1. *Suppose $(t_{cbdh}, \epsilon_{cbdh})$-CBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ and hash functions G and H are random oracles. Then, KwAH1 and KwAH2 are $(t_{kwah}, \epsilon_{kwah})$-IND-KE-CPA secure as long as:*

$$\epsilon_{kwah} \leq \frac{3q_G N}{2} \epsilon_{cbdh}$$
$$t_{kwah} \leq t_{cbdh} + \Theta(\tau(2q_H + 3q_E)),$$

where IND-KE-CPA adversary $\mathcal{A}_{kwah}$ issues at most q_H H-queries, q_G G-queries and q_E Exposure queries. Here, τ is the maximum time for computing an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$, and pairing e.

Proof:

Proof for KwAH1. We show that we can construct an algorithm $\mathcal{A}_{cbdh}$ that can

solve the CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ by using an adversary $\mathcal{A}_{kwah}$ that breaks IND-KE-CPA security of our scheme. The algorithm $\mathcal{A}_{cbdh}$ is given an instance $\langle g, g^a, g^b, g^c \rangle$ in $\mathbb{G}_1^4$ from the challenger and tries to output $e(g,g)^{abc}$ using $\mathcal{A}_{kwah}$. Let $g_1 = g^a, g_2 = g^b, g_3 = g^c$. The algorithm $\mathcal{A}_{cbdh}$ works by interacting with $\mathcal{A}_{kwah}$ in an IND-KE-CPA game as follows:

Before we start the simulation, we let $\mathcal{A}_{cbdh}$ flip a coin $\mathcal{COIN} \in \{0,1\}$ such that we have $\Pr[\mathcal{COIN} = 0] = \delta$ for some δ which we will determine later. If $\mathcal{COIN} = 0$, $\mathcal{A}_{cbdh}$ simulates the responses to $\mathcal{A}_{kwah}$'s queries expecting that $\mathcal{A}_{kwah}$ will never submit $\langle \cdot$, "main helper"$\rangle$ nor $\langle \cdot$, "auxiliary helper"$\rangle$ as Exposure query. If $\mathcal{COIN} = 1$, $\mathcal{A}_{cbdh}$ carries out the simulation expecting that $\mathcal{A}_{kwah}$ will submit $\langle \cdot$, "main helper"$\rangle$ or $\langle \cdot$, "auxiliary helper"$\rangle$.

If $\mathcal{COIN} = 0$, $\mathcal{A}_{cbdh}$ responses to $\mathcal{A}_{kwah}$'s queries will be as follows:

Setup: $\mathcal{A}_{cbdh}$ picks a random $s \in \mathbb{Z}_q^*$. Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{kwah}$ the system parameter

$$pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H \rangle,$$

where $h_1 = g_1$ and $h_2 = g_1^s$, and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below.

G-queries: $\mathcal{A}_{kwah}$ issues up to q_G queries to the random oracle G. To respond to these queries algorithm, $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle W, x \rangle$ as explained below. We call this list G_{list}. The list is initially empty. When $\mathcal{A}_{kwah}$ gives $\mathcal{A}_{cbdh}$ a query W to the oracle G, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query W already appears on the G_{list} in a tuple $\langle W, x \rangle$, then $\mathcal{A}_{cbdh}$ outputs $G(W) = x$.
2. $\mathcal{A}_{cbdh}$ chooses a random $x \in \{0,1\}^n$.
3. $\mathcal{A}_{cbdh}$ adds the tuple $\langle W, x \rangle$ to the G_{list} and outputs $G(W) = x$.

H-queries: $\mathcal{A}_{cbdh}$ picks a random $\alpha \in \{1, ..., N\}$ in advance. $\mathcal{A}_{kwah}$ issues up to q_H queries to the random oracle H. To respond to these queries algorithm, $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle i, u_i, r_i \rangle$ as explained below. We call the list H_{list}. The list is initially empty. When $\mathcal{A}_{kwah}$ gives $\mathcal{A}_{cbdh}$ a query i to the oracle H, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query i already appears on the H_{list} in a tuple $\langle i, u_i, r_i \rangle$, then $\mathcal{A}_{cbdh}$ outputs $H(i) = u_i$.
2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_\alpha = 0$.
3. If $i < \alpha$, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g^{r_i}$.
4. If $i > \alpha$, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g_2^z \cdot g^{r_i}$, where

- $z = 1$ if $\alpha = 0 \bmod \ell$ and $i = 0 \bmod \ell$,
- $z = -s$ if $\alpha = 0 \bmod \ell$ and $i \neq 0 \bmod \ell$,
- $z = 1$ if $\alpha = -1 \bmod \ell$ and $i \neq 0 \bmod \ell$,
- $z = -s^{-1}$ (s^{-1} is the inverse of $s \bmod q$) if $\alpha = -1 \bmod \ell$ and $i = 0 \bmod \ell$,
- $z = 0$ otherwise,

5. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, u_i, r_i \rangle$ to the H_{list} and outputs $H(i) = u_i$.

Challenge: Once algorithm $\mathcal{A}_{kwah}$ decides that Phase 1 is over, it outputs a target stage i^* and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{A}_{cbdh}$ responds as follows:

1. $\mathcal{A}_{cbdh}$ sets $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as:

$$c_0^* = g_3$$
$$c_1^* = \mu$$

where $\mu \in_R \{0,1\}^n$.

2. $\mathcal{A}_{cbdh}$ gives $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as the challenge ciphertext to $\mathcal{A}_{kwah}$.

Exposure queries: $\mathcal{A}_{kwah}$ issues up to q_E Exposure queries. When $\mathcal{A}_{kwah}$ gives a query $\langle i, \texttt{class} \rangle$, $\mathcal{A}_{cbdh}$ responds as follows:

1. If `class` = "main helper" or "auxiliary helper", $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ aborts the simulation.
3. $\mathcal{A}_{cbdh}$ runs the algorithm for responding to H-queries to obtain $\langle i, u_i, r_i \rangle$ and $\langle j, u_j, r_j \rangle$, where $j = i - 1$ if $i = 0 \bmod \ell$, or $j = L$ such that $i - \ell < L < i$, $L = 0 \bmod \ell$ otherwise.
4. $\mathcal{A}_{cbdh}$ sets $usk_i = h_1^{r_{i-1}} \cdot h_2^{r_i}$ if $i = 0 \bmod \ell$, or $usk_i = h_1^{r_i} \cdot h_2^{r_L}$ otherwise. Observe that usk_i is the user secret key corresponding to the stage i. Especially, when $i > \alpha$,

$$u_{i-1}^{\log_g h_1} \cdot u_i^{\log_g h_2} = \begin{cases} (g_2^{-s} \cdot g^{r_{i-1}})^a \cdot (g_2 \cdot g^{r_i})^{s \cdot a} \\ \quad (\text{if } i = 0 \bmod \ell,\ \alpha = 0 \bmod \ell) \\ (g_2 \cdot g^{r_{i-1}})^a \cdot (g_2^{-s^{-1}} \cdot g^{r_i})^{s \cdot a} \\ \quad (\text{if } i = 0 \bmod \ell,\ \alpha = -1 \bmod \ell) \end{cases}$$

$$u_{i-1}^{\log_g h_1} \cdot u_i^{\log_g h_2} = \begin{cases} g^{a \cdot r_{i-1}} \cdot g^{r_i \cdot s \cdot a} = h_1^{r_{i-1}} h_2^{r_i} \\ \quad (\text{if } i = 0 \bmod \ell,\ \alpha = 0 \bmod \ell) \\ g^{a \cdot r_{i-1}} \cdot g^{r_i \cdot s \cdot a} = h_1^{r_{i-1}} h_2^{r_i} \\ \quad (\text{if } i = 0 \bmod \ell,\ \alpha = -1 \bmod \ell) \end{cases}$$

$$u_L^{\log_g h_2} \cdot u_i^{\log_g h_1} = \begin{cases} (g_2 \cdot g^{r_L})^{s \cdot a} \cdot (g_2^{-s} \cdot g^{r_i})^a \\ \quad (\text{if } i \neq 0 \bmod \ell,\ \alpha = 0 \bmod \ell) \\ (g_2^{-s^{-1}} \cdot g^{r_L})^{s \cdot a} \cdot (g_2 \cdot g^{r_i})^a \\ \quad (\text{if } i \neq 0 \bmod \ell,\ \alpha = -1 \bmod \ell) \end{cases}$$

$$= \begin{cases} g^{s \cdot a \cdot r_L} \cdot g^{r_i \cdot a} = h_2^{r_L} h_1^{r_i} \\ \quad (\text{if } i \neq 0 \bmod \ell,\ \alpha = 0 \bmod \ell) \\ g^{s \cdot a \cdot r_L} \cdot g^{r_i \cdot a} = h_2^{r_L} h_1^{r_i} \\ \quad (\text{if } i \neq 0 \bmod \ell,\ \alpha = -1 \bmod \ell) \end{cases}$$

5. $\mathcal{A}_{cbdh}$ outputs usk_i to $\mathcal{A}_{kwah}$.

Guess: When $\mathcal{A}_{kwah}$ decides that Phase 2 is over, $\mathcal{A}_{kwah}$ outputs its guess bit $\beta' \in \{0, 1\}$. At the same time, algorithm $\mathcal{A}_{cbdh}$ terminates the simulation. Then, $\mathcal{A}_{cbdh}$ picks a tuple $\langle W, x \rangle$ uniformly at random from the G_{list}, and computes

$$T = \begin{cases} (\frac{W}{e(g_1, g_3)^{r_{\alpha-1}}})^{s^{-1}} & \text{if } \alpha = 0 \bmod \ell, \\ \\ (\frac{W}{e(g_1, g_3)^{s \cdot r_A}}) & \text{otherwise.} (\alpha - \ell < A < \alpha,\ A = 0 \bmod \ell) \end{cases}$$

Finally, $\mathcal{A}_{cbdh}$ outputs T.

Claim 1. *If $i^* = \alpha$ and $\mathcal{A}_{cbdh}$ does not abort, then $\mathcal{A}_{kwah}$'s view is identical to its view in the real attack until $\mathcal{A}_{kwah}$ submits W^* as a G-query, where*

$$W^* = \begin{cases} e(g_1, g_3)^{r_{\alpha-1}} \cdot e(g, g)^{s \cdot abc} & \textit{if } \alpha = 0 \bmod \ell, \\ e(g_1, g_3)^{s \cdot r_A} \cdot e(g, g)^{abc} & \textit{otherwise.} \end{cases}$$

We note that if $i^* = \alpha$,

$$e(g, g)^{abc} = \begin{cases} (\frac{W^*}{e(g_1, g_3)^{r_{\alpha-1}}})^{s^{-1}} & \textit{if } \alpha = 0 \bmod \ell, \\ \\ (\frac{W^*}{e(g_1, g_3)^{s \cdot r_A}}) & \textit{otherwise.} \end{cases}$$

Proof. It is obvious that the responses to G are perfect. The responses to H are also as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1$. Interestingly, the responses to Exposure queries are perfect if $\mathcal{A}_{cbdh}$ does not abort. Finally, we show that the response to Challenge is indistinguishable from the real attack until $\mathcal{A}_{kwah}$ submits W^*. Let the response to Challenge be $C^* = \langle \alpha, c_0^*, c_1^* \rangle$. Then, c_0^* is uniformly distributed in $\mathbb{G}_1$ due to random $\log_g g_3 (= c)$, and therefore are as in the real attack. Also, since $c_1^* = M_\beta \oplus G(W^*)$, it is information-theoretically impossible to obtain any information on M_β unless $\mathcal{A}_{kwah}$ asks $G(W^*)$. □

Next, let us define by E_1, an event assigned to be true if and only if $i^* = \alpha$. Similarly, let us define by E_2, an event assigned to be true if and only if a G-query coincides with W^*, and by E_{msk}, an event assigned to be true if and only if an Exposure query coincides with $\langle \cdot$, "main helper"$\rangle$ or $\langle \cdot$, "auxiliary helper"$\rangle$.

Claim 2. *We have that* $\Pr[\beta' = \beta | E_1, \neg E_{msk}] \geq \Pr[\beta' = \beta | \neg E_{msk}]$.

Proof. Since α is uniformly chosen from $\{1, \cdots, N\}$ at random, E_1 is independent to $\mathcal{A}_{kwah}$'s view in the real world. Hence, we have $\Pr[\beta' = \beta | E_1, \neg E_{msk}] \geq \Pr[\beta' = \beta | \neg E_{msk}]$. □

Claim 3. *We have that* $\Pr[\beta' = \beta | E_1, \neg E_2, \neg E_{msk}] = 1/2$.

Proof. Let the response to Challenge be $C^* = \langle \alpha, c_0^*, c_1^* \rangle$. Since $c_1^* = M_\beta \oplus G(W^*)$, it is information-theoretically impossible to obtain any information on M_β without submitting W^* as a G-query. This implies that $\mathcal{A}_{kwah}$'s best strategy becomes a random guess if E_2 is false. Hence, we have $\Pr[\beta' = \beta | E_1, \neg E_2, \neg E_{msk}] = 1/2$. □

Claim 4. *We have that*

$$\begin{aligned} &\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 0] \\ &\geq \frac{1}{q_G N} \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}]. \end{aligned}$$

Proof. If $i^* = \alpha$, then $e(g,g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_2]$. Obviously, we have $\Pr[E_2] \geq \Pr[E_2, E_1, \neg E_{msk}] = \Pr[E_2 | E_1, \neg E_{msk}] \Pr[E_1 | \neg E_{msk}]$
$\Pr[\neg E_{msk}]$ and $\Pr[E_1 | \neg E_{msk}] = 1/N$. Hence, by choosing a tuple from G_{list} uniformly at random, $\mathcal{A}_{cbdh}$ can correctly output $e(g,g)^{abc}$ with probability of at least $1/q_G \cdot 1/N \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}]$. □

Finally, we calculate $p_0 := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 0]$ from the above claims. Letting $\lambda := \Pr[\beta' = \beta | E_{msk}] - 1/2$, from Claims 1, 2 and 3, we have

$$\begin{aligned}
&\Pr[\beta' = \beta] - \frac{1}{2} \\
&= \Pr[\beta' = \beta | \neg E_{msk}] \Pr[\neg E_{msk}] + \\
&\qquad \Pr[\beta' = \beta | E_{msk}] \Pr[E_{msk}] - \frac{1}{2} \\
&= \Pr[\beta' = \beta | \neg E_{msk}](1 - \Pr[E_{msk}]) + \\
&\qquad (\frac{1}{2} + \lambda) \Pr[E_{msk}] - \frac{1}{2} \\
&\leq \Pr[\beta' = \beta | E_1, \neg E_{msk}](1 - \Pr[E_{msk}]) + \\
&\qquad (\frac{1}{2} + \lambda) \Pr[E_{msk}] - \frac{1}{2} \\
&= (\Pr[\beta' = \beta | E_1, E_2, \neg E_{msk}] \Pr[E_2 | E_1, \neg E_{msk}] \\
&+ \Pr[\beta' = \beta | E_1, \neg E_2, \neg E_{msk}] \Pr[\neg E_2 | E_1, \neg E_{msk}]) \\
&\quad \cdot (1 - \Pr[E_{msk}]) + (\frac{1}{2} + \lambda) \Pr[E_{msk}] - \frac{1}{2} \\
&\leq (\Pr[E_2 | E_1, \neg E_{msk}] + \frac{1}{2}(1 - \Pr[E_2 | E_1, \neg E_{msk}])) \\
&\qquad \cdot (1 - \Pr[E_{msk}]) + (\frac{1}{2} + \lambda) \Pr[E_{msk}] - \frac{1}{2} \\
&= \frac{1}{2} \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}] + \lambda \Pr[E_{msk}].
\end{aligned}$$

From Claim 4, we have

$$p_0 \geq \frac{2}{q_G N}(\epsilon_{kwah} - \lambda \Pr[E_{msk}]).$$

Next, we discuss for the $\mathcal{COIN} = 1$ case. If $\mathcal{COIN} = 1$, $\mathcal{A}_{cbdh}$ responses to $\mathcal{A}_{kwah}$'s queries as follows:

Setup: $\mathcal{A}_{cbdh}$ picks random $s \in \mathbb{Z}_q^*$ and $\mathbf{b} \in \{1, 2\}$. Let $\bar{\mathbf{b}}$ be 1 (resp. 2) if $\mathbf{b} = 2$ (resp. 1). Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{kwah}$ the system parameter

$$pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H \rangle,$$

where $h_{\mathbf{b}} = g_1$ and $h_{\bar{\mathbf{b}}} = g^s$ (we expect that $\mathcal{A}_{kwah}$ asks ak if $\mathbf{b} = 1$, or mk otherwise), and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below.

G-queries: $\mathcal{A}_{kwah}$ issues up to q_G queries to the random oracle G. To respond to these queries algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle W, x\rangle$ as explained below. We call this list G_{list}. The list is initially empty. When $\mathcal{A}_{kwah}$ gives $\mathcal{A}_{cbdh}$ a query W to the oracle G, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query W already appears on the G_{list} in a tuple $\langle W, x\rangle$, then $\mathcal{A}_{cbdh}$ outputs $G(W) = x$.
2. $\mathcal{A}_{cbdh}$ chooses a random $x \in \{0,1\}^n$.
3. $\mathcal{A}_{cbdh}$ adds the tuple $\langle W, x\rangle$ to the G_{list} and outputs $G(W) = x$.

H-queries: $\mathcal{A}_{cbdh}$ picks a random $\alpha \in \{1, ..., N\}$ in advance. $\mathcal{A}_{kwah}$ issues up to q_H queries to the random oracle H. To respond to these queries algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle i, u_i, r_i\rangle$ as explained below. We call the list H_{list}. The list is initially empty. When $\mathcal{A}_{kwah}$ gives $\mathcal{A}_{cbdh}$ a query i to the oracle H, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query i already appears on the H_{list} in a tuple $\langle i, u_i, r_i\rangle$, then $\mathcal{A}_{cbdh}$ outputs $H(i) = u_i$.
2. If $i = \alpha - 1$, $\mathbf{b} = 1$, and $\alpha = 0 \bmod \ell$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
3. If $i = \alpha$, $\mathbf{b} = 1$, and $\alpha = -1 \bmod \ell$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
4. If $i = \alpha$, $\mathbf{b} = 1$, $\alpha \neq 0 \bmod \ell$, and $\alpha \neq -1 \bmod \ell$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
5. If $i = \alpha$, and $\mathbf{b} = 2$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
6. Else, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g^{r_i}$.
7. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, u_i, r_i\rangle$ to the H_{list} and outputs $H(i) = u_i$.

Challenge: Once algorithm $\mathcal{A}_{kwah}$ decides that Phase 1 is over, it outputs a target stage i^* and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{A}_{cbdh}$ responds as follows:

1. $\mathcal{A}_{cbdh}$ sets $C^* = \langle i^*, c_0^*, c_1^*\rangle$ as:

$$c_0^* = g_3$$
$$c_1^* = \mu$$

   where $\mu \in_R \{0,1\}^n$.
2. $\mathcal{A}_{cbdh}$ gives $C^* = \langle i^*, c_0^*, c_1^*\rangle$ as the challenge ciphertext to $\mathcal{A}_{kwah}$.

Exposure queries: $\mathcal{A}_{kwah}$ issues up to q_E Exposure queries. When $\mathcal{A}_{kwah}$ gives a query $\langle i, \texttt{class}\rangle$, $\mathcal{A}_{cbdh}$ responds as follows:

1. If $\mathbf{b} = 1$ and $\texttt{class}$ = "main helper", $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If $\mathbf{b} = 1$ and $\texttt{class}$ = "auxiliary helper", $\mathcal{A}_{cbdh}$ returns s to $\mathcal{A}_{kwah}$.
3. If $\mathbf{b} = 2$ and $\texttt{class}$ = "main helper", $\mathcal{A}_{cbdh}$ returns s to $\mathcal{A}_{kwah}$.

4. If $\mathbf{b} = 2$ and `class` = "auxiliary helper", $\mathcal{A}_{cbdh}$ aborts the simulation.
5. If $i = \alpha$ and `class` = "user", $\mathcal{A}_{cbdh}$ aborts the simulation.
6. If $i = \alpha - 1$, `class` = "user", $\mathbf{b} = 1$ and $\alpha = 0 \bmod \ell$, $\mathcal{A}_{cbdh}$ aborts the simulation.
7. If $A \leq i < A + \ell$, `class` = "user", and $\mathbf{b} = 2$, where $A \leq \alpha < A + \ell$ and $A = 0 \bmod \ell$, $\mathcal{A}_{cbdh}$ aborts the simulation.
8. Else[*1], $\mathcal{A}_{cbdh}$ runs the algorithm for responding to H-queries to obtain $\langle i, u_i, r_i \rangle$ and $\langle j, u_j, r_j \rangle$, where $j = i - 1$ if $i = 0 \bmod \ell$, or $j = L$ such that $i - \ell < L < i$, $L = 0 \bmod \ell$ otherwise.
9. $\mathcal{A}_{cbdh}$ sets $usk_i = h_1^{r_{i-1}} \cdot h_2^{r_i}$ if $i = 0 \bmod \ell$, or $usk_i = h_1^{r_i} \cdot h_2^{r_L}$ otherwise.
10. $\mathcal{A}_{cbdh}$ outputs usk_i to $\mathcal{A}_{kwah}$.

Guess: When $\mathcal{A}_{kwah}$ decides that Phase 2 is over, $\mathcal{A}_{kwah}$ outputs the guess bit $\beta' \in \{0, 1\}$. At the same time, algorithm $\mathcal{A}_{cbdh}$ terminates the simulation. Then, $\mathcal{A}_{cbdh}$ picks a tuple $\langle W, x \rangle$ uniformly at random from the G_{list}, and computes

$$T = \begin{cases} W \cdot e(g, g_3)^{-s \cdot r_\alpha} & \text{if } \mathbf{b} = 1 \text{ and } \alpha = 0 \bmod \ell, \\ W \cdot e(g, g_3)^{-s \cdot r_{\alpha-1}} & \text{if } \mathbf{b} = 2 \text{ and } \alpha = 0 \bmod \ell, \\ W \cdot e(g, g_3)^{-s \cdot r_A} & \text{otherwise.} \end{cases}$$

Finally, $\mathcal{A}_{cbdh}$ outputs T.

Claim 5. *If $i^* = \alpha$ and $\mathcal{A}_{cbdh}$ does not abort, then $\mathcal{A}_{kwah}$'s view is identical to its view in the real attack until $\mathcal{A}_{kwah}$ submits W^* as a G-query, where*

$$W^* = \begin{cases} e(g, g_3)^{s \cdot r_\alpha} \cdot e(g, g)^{abc} & \textit{if } \mathbf{b} = 1,\ \alpha = 0 \bmod \ell, \\ e(g, g_3)^{s \cdot r_{\alpha-1}} \cdot e(g, g)^{abc} & \textit{if } \mathbf{b} = 2,\ \alpha = 0 \bmod \ell, \\ e(g, g_3)^{s \cdot r_A} \cdot e(g, g)^{abc} & \textit{otherwise.} \end{cases}$$

We note that if $i^* = \alpha$,

$$e(g, g)^{abc} = \begin{cases} W^* \cdot e(g, g_3)^{-s \cdot r_\alpha} & \textit{if } \mathbf{b} = 1,\ \alpha = 0 \bmod \ell, \\ W^* \cdot e(g, g_3)^{-s \cdot r_{\alpha-1}} & \textit{if } \mathbf{b} = 2,\ \alpha = 0 \bmod \ell, \\ W^* \cdot e(g, g_3)^{-s \cdot r_A} & \textit{otherwise.} \end{cases}$$

Proof. It is obvious that the responses to G are perfect. The responses to H are also as in the real attack since each response is uniformly and independently distributed

[*1] Notice that in this case, `class` is always "user".

in $\mathbb{G}_1$. The responses to Exposure queries are perfect if $\mathcal{A}_{cbdh}$ does not abort. Finally, we show that the response to Challenge is indistinguishable from the real attack until $\mathcal{A}_{kwah}$ submits W^*. Let the response to Challenge be $C^* = \langle \alpha, c_0^*, c_1^* \rangle$. Then, c_0^* is uniformly distributed in $\mathbb{G}_1$ due to random $\log_g g_3 (= c)$, and therefore are as in the real attack. Also, since $c_1^* = M_\beta \oplus G(W^*)$, it is information-theoretically impossible to obtain any information on M_β unless $\mathcal{A}_{kwah}$ asks $G(W^*)$. □

Next, let us define by E_3, an event assigned to be true if and only if $i^* = \alpha$. Similarly, let us define by E_4, an event assigned to be true if and only if a G-query coincides with W^*, by E_5, an event assigned to be true if and only if an Exposure query coincides with $\langle \cdot$, "main helper"$\rangle$ if $\mathbf{b} = 1$ or $\langle \cdot$, "auxiliary helper"$\rangle$ if $\mathbf{b} = 2$, and by E_{msk}, an event assigned to be true if and only if an Exposure query coincides with $\langle \cdot$, "main helper"$\rangle$ or $\langle \cdot$, "auxiliary helper"$\rangle$. Notice that E_{msk} is identical to that in the case of $\mathcal{COIN} = 0$.

Claim 6. *We have that*
$\Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}] \geq \Pr[\beta' = \beta | E_{msk}]$.

Proof. Since α is uniformly chosen from $\{1, ..., N\}$ at random, E_3 is independent to $\mathcal{A}_{kwah}$'s view in the real world. Hence, we have $\Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}] \geq \Pr[\beta' = \beta | \neg E_5, E_{msk}]$. Due to symmetricity between E_5 and $\neg E_5$, we have $\Pr[E_5 | E_{msk}] = \Pr[\neg E_5 | E_{msk}] = 1/2$ and $\Pr[\beta' = \beta | E_5, E_{msk}] = \Pr[\beta' = \beta | \neg E_5, E_{msk}]$. Therefore,

$$\begin{aligned}
&\Pr[\beta' = \beta | E_{msk}] \\
&= \frac{1}{2} \Pr[\beta' = \beta | E_5, E_{msk}] + \frac{1}{2} \Pr[\beta' = \beta | \neg E_5, E_{msk}] \\
&= \Pr[\beta' = \beta | \neg E_5, E_{msk}],
\end{aligned}$$

and we finally have
$\Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}] \geq \Pr[\beta' = \beta | E_{msk}]$. □

Claim 7. *We have that* $\Pr[\beta' = \beta | E_3, \neg E_4, \neg E_5, E_{msk}] = 1/2$.

Proof. Let the response to Challenge be $C^* = \langle \alpha, c_0^*, c_1^* \rangle$. Since $c_1^* = M_\beta \oplus G(W^*)$, it is information-theoretically impossible to obtain any information on M_β without submitting W^* as a G-query. This implies that $\mathcal{A}_{kwah}$'s best strategy becomes a random guess if E_4 is false. Hence, we have

$$\Pr[\beta' = \beta | E_3, \neg E_4, \neg E_5, E_{msk}] = 1/2.$$

□

Claim 8. *We have that*

$$\begin{aligned}&\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 1] \\ &\geq \frac{1}{2q_G N} \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}].\end{aligned}$$

Proof. If $i^* = \alpha$, then $e(g,g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_4]$. Obviously, we have

$$\begin{aligned}&\Pr[E_4] \geq \Pr[E_4, E_3, \neg E_5, E_{msk}] \\ &= \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_3 | \neg E_5, E_{msk}] \Pr[\neg E_5, E_{msk}].\end{aligned}$$

Furthermore, we have $\Pr[E_3 | \neg E_5, E_{msk}] = 1/N$, and $\Pr[\neg E_5, E_{msk}] = 1/2 \cdot \Pr[E_{msk}]$. Hence, by choosing a tuple from G_{list} uniformly at random, $\mathcal{A}_{cbdh}$ can correctly output $e(g,g)^{abc}$ with probability of at least $1/q_G \cdot 1/N \cdot 1/2 \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}]$. □

Finally, we calculate $p_1 := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 1]$ from the above claims. Letting $\eta := \Pr[\beta' = \beta | \neg E_{msk}] - 1/2$, from Claims 5, 6 and 7, we have

$$\begin{aligned}&\Pr[\beta' = \beta] - \frac{1}{2} \\ &= \Pr[\beta' = \beta | \neg E_{msk}] \Pr[\neg E_{msk}] + \\ &\qquad \Pr[\beta' = \beta | E_{msk}] \Pr[E_{msk}] - \frac{1}{2} \\ &= (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] + \\ &\qquad \Pr[\beta' = \beta | E_{msk}](1 - \Pr[\neg E_{msk}]) - \frac{1}{2} \\ &\leq (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] + \\ &\qquad \Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}](1 - \Pr[\neg E_{msk}]) - \frac{1}{2}\end{aligned}$$

$$
\begin{aligned}
&\Pr[\beta' = \beta] - \frac{1}{2} \\
&= (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] \\
&+ (\Pr[\beta' = \beta | E_3, E_4, \neg E_5, E_{msk}] \Pr[E_4 | E_3, \neg E_5, E_{msk}] \\
&+ \Pr[\beta' = \beta | E_3, \neg E_4, \neg E_5, E_{msk}] \Pr[\neg E_4 | E_3, \neg E_5, E_{msk}]) \\
&\quad \cdot (1 - \Pr[\neg E_{msk}]) - \frac{1}{2} \\
&\leq (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] + (\Pr[E_4 | E_3, \neg E_5, E_{msk}] + \\
&\frac{1}{2}(1 - \Pr[E_4 | E_3, \neg E_5, E_{msk}])) \cdot (1 - \Pr[\neg E_{msk}]) - \frac{1}{2} \\
&= \frac{1}{2} \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}].
\end{aligned}
$$

From Claim 8, we have

$$p_1 \geq \frac{1}{q_G N} (\epsilon_{kwah} - \eta \Pr[\neg E_{msk}]).$$

Claim 9. *We have that* $\epsilon_{kwah} \geq \lambda \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}]$.

Proof. By the definitions of λ and η, we have $\lambda + 1/2 = \Pr[\beta' = \beta | E_{msk}]$ and $\eta + 1/2 = \Pr[\beta' = \beta | \neg E_{msk}]$, and consequently,

$$
\begin{aligned}
\epsilon_{kwah} + \frac{1}{2} &\geq \Pr[\beta' = \beta] \\
&= (\lambda + \frac{1}{2}) \Pr[E_{msk}] + (\eta + \frac{1}{2}) \Pr[\neg E_{msk}].
\end{aligned}
$$

Hence, we have $\epsilon_{kwah} \geq \lambda \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}]$, which proves the claim. □

We note that in both cases of $\mathcal{COIN} = 0$ and $\mathcal{COIN} = 1$, simulations are perfect until $\mathcal{A}_{kwah}$ submits W^* as a G-query if $i^* = \alpha$ and $\mathcal{A}_{cbdh}$ does not abort. Hence, the views of $\mathcal{A}_{kwah}$ are identical for these cases.

Now, we calculate $\epsilon_{cbdh} := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc}]$. From Claim 9, we

have

$$\begin{aligned}
\epsilon_{cbdh} &= \delta \cdot p_0 + (1-\delta) \cdot p_1 \\
&\geq \delta(\frac{2}{q_G N}(\epsilon_{kwah} - \lambda \Pr[E_{msk}])) + \\
&\qquad (1-\delta)(\frac{1}{q_G N}(\epsilon_{kwah} - \eta \Pr[\neg E_{msk}])) \\
&\geq \delta(\frac{2}{q_G N}(\epsilon_{kwah} - \lambda \Pr[E_{msk}])) + \\
&\qquad (1-\delta)(\frac{1}{q_G N}\lambda \Pr[E_{msk}]) \\
&\geq \frac{1}{q_G N}(2\delta\epsilon_{kwah} + (1-3\delta)\lambda \Pr[E_{msk}])
\end{aligned}$$

By letting $\delta = 1/3$, we finally have

$$\epsilon_{cbdh} \geq \frac{2}{3q_G N}\epsilon_{kwah}.$$

From the above discussions, we can see that the claimed bound of the running-time of $\mathcal{A}_{cbdh}$ holds. This completes the proof of the theorem 1 for KwAH1. □

Proof for KwAH2. The proof can be done in the same way with the proof for KwAH1. We construct the responses of $\mathcal{A}_{cbdh}$ as follows:

If $\mathcal{COIN} = 0$, the simulation is as follows.

Setup: $\mathcal{A}_{cbdh}$ picks a random $s \in \mathbb{Z}_q^*$. Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{kwah}$ the system parameter

$$pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h, G, H \rangle,$$

where $h = g_1^{s+1}$, and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below.

G, H-queries: same as in KwAH1

Challenge: same as in KwAH1, except that $C^* = (i^*, h_3^*, c_0^*, c_1^*)$, $h_3^* \in \{0,1\}^n$

Exposure queries: same as in KwAH1, except that:

5. $\mathcal{A}_{cbdh}$ outputs $(usk_i,\ g_1^s)$ to $\mathcal{A}_{kwah}$.

Guess: same as in KwAH1 except that

$$T = W^{(s+1)^{-1}}$$

If $\mathcal{COIN} = 1$, $\mathcal{A}_{cbdh}$ responses to $\mathcal{A}_{kwah}$'s queries as follows:

Setup: $\mathcal{A}_{cbdh}$ picks random $s \in \mathbb{Z}_q^*$ and $\mathbf{b} \in \{1, 2\}$. Let $\bar{\mathbf{b}}$ be 1 (resp. 2) if $\mathbf{b} = 2$ (resp. 1). Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{kwah}$ the system parameter

$$pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h, G, H \rangle,$$

where $h = g_1 \cdot g^s$ (we expect that $\mathcal{A}_{kwah}$ asks ak if $\mathbf{b} = 1$, or mk otherwise), and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below. Set $h_{\mathbf{b}} = g_1$ and $h_{\bar{\mathbf{b}}} = g^s$

G-queries : same as in KwAH1

H-queries :same as in KwAH1 but the H_{list} and responses are as follows:

1. If the query i already appears on the H_{list} in a tuple $\langle i, u_i, r_i \rangle$, then outputs $H(i) = u_i$.
2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_\alpha = 0$.
3. Else, $\mathcal{A}_{cbdh}$ choose a random $i \in \mathbb{Z}_q^*$ and sets $u_i = g^{r_i}$.
4. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, u_i, r_i \rangle$ to the H_{list} and outputs $H(i) = u_i$.

Challenge: same as in KwAH1, except that here

$C^* = (i^*, h_3^*, c_0^*, c_1^*)$

Exposure queries: same as in KwAH1, except that:

10. $\mathcal{A}_{cbdh}$ outputs $(usk_i,\ g_1^s)$ to $\mathcal{A}_{kwah}$.

Guess: same as in KwAH1 except that

$$T = W \cdot e(g_2, g_3)^{-s}$$

We can compute the advantage of $\mathcal{A}_{cbdh}$ in this simulation and show that theorem 1 is right for KwAH2 in the same way as for KwAH1. □

Security of KwAH1 and KwAH2 can also be proven under GBDH assumption with a tighter security reduction.

Theorem 2. *Suppose $(t_{gbdh}, \epsilon_{gbdh})$-GBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ and hash functions G and H are random oracles. Then, KwAH1,KwAH2 are $(t_{kwah}, \epsilon_{kwah})$-IND-KE-CPA secure as long as*

$$\epsilon_{kwah} \leq \frac{3N}{2} \epsilon_{gbdh}$$
$$t_{kwah} \leq t_{gbdh} + \Theta(\tau(2q_H + 3q_E)),$$

where IND-KE-CPA adversary $\mathcal{A}_{kwah}$ issues at most q_H H-queries and q_E Exposure queries. Here, τ is the maximum time for computing an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$, and pairing e.

Proof: The proof of Theorem 2 is similar to Theorem 1. We construct an algorithm $\mathcal{A}_{gbdh}$ that can solve the GBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ by using an adversary $\mathcal{A}_{kwah}$ that breaks IND-KE-CPA security of our scheme. The algorithm $\mathcal{A}_{gbdh}$ is given an instance $\langle g, g^a, g^b, g^c \rangle$ in $\mathbb{G}_1^4$ from the challenger and can guess if a random $T \in \mathbb{G}_2$ equals to $e(g,g)^{abc}$ or not by using decision BDH oracle. This $\mathcal{A}_{gbdh}$ will try to output $e(g,g)^{abc}$ using $\mathcal{A}_{kwah}$. We build two kinds of responses to $\mathcal{A}_{kwah}$, corresponding to the cases of $\mathcal{COIN} = 0$ and $\mathcal{COIN} = 1$ the same as in the proof of theorem The advantage of the adversary $\mathcal{A}_{kwah}$ is also computed in the same way as in the proof of Theorem 1, except that Claim 4 and Claim 8 will be changed as follows:

Claim 4′. *We have that*

$$\begin{aligned}&\Pr[\mathcal{A}_{gbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 0] \\ &\geq \frac{1}{N} \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}].\end{aligned}$$

Proof. If $i^* = \alpha$, then $e(g,g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_2]$. Obviously, we have

$$\begin{aligned}\Pr[E_2] &\geq \Pr[E_2, E_1, \neg E_{msk}] \\ &= \Pr[E_2 | E_1, \neg E_{msk}] \Pr[E_1 | \neg E_{msk}] \Pr[\neg E_{msk}]\end{aligned}$$

and $\Pr[E_1 | \neg E_{msk}] = 1/N$. With the help of a decision BDH oracle $\mathcal{O}$, $\mathcal{A}_{gbdh}$ can choose $e(g,g)^{abc}$ from G_{list} if E_2 is true. Hence, $\mathcal{A}_{cbdh}$ can correctly output $e(g,g)^{abc}$ with probability of at least $1/N \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}]$. □

Claim 8′. *We have that*

$$\begin{aligned}&\Pr[\mathcal{A}_{gbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = 1] \\ &\geq \frac{1}{2N} \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}]\end{aligned}$$

Proof. If $i^* = \alpha$, then $e(g,g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_4]$. Obviously, we have

$$\begin{aligned}&\Pr[E_4] \geq \Pr[E_4, E_3, \neg E_5, E_{msk}] \\ &= \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_3 | \neg E_5, E_{msk}] \Pr[\neg E_5, E_{msk}]\end{aligned}$$

Furthermore, we have $\Pr[E_3 | \neg E_5, E_{msk}] = 1/N$, and $\Pr[\neg E_5, E_{msk}] = 1/2 \cdot \Pr[E_{msk}]$. Hence, with the help of a decision BDH oracl $\mathcal{O}$, $\mathcal{A}_{gbdh}$ can correctly output $e(g,g)^{abc}$ with probability of at least $1/N \cdot 1/2 \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}]$. □

Using Claim 4′ and Claim 8′ instead of Claim 4 and Claim 8 in computing the adversary's advantage with the same way as in proof of theorem 1, we have

$$\epsilon_{kwah} \leq \frac{3N}{2}\epsilon_{gbdh}$$

The bound of running time is the same as in Theorem 2. □

5.2 Strongly IND-KE-CPA Schemes

We can build strongly IND-KE-CPA schemes KwAH1′ and KwAH2′ by only slightly modifying KwAH1 and KwAH2, respectively.

5.2.1 KwAH1′: Strongly IND-KE-CPA Construction

The first strongly IND-KE-CPA scheme KwAH1′ is based on KwAH1. It consists the following algorithms:

KeyGen: Given a security parameter k, **KeyGen** algorithm does the same as that of KwAH1 except that it:

2. picks random, $s_1, s_2, s_3 \in \mathbb{Z}_q^*$, and sets $h_1 = g^{s_1 s_3}$ and $h_2 = g^{s_2 s_3}$,
5. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H\rangle$, $mk = s_1$, $ak = s_2$ and $usk_0 = \langle d_{-1}^{s_3} \cdot d_0^{s_3}, s_3\rangle$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1^* \times \{0,1\}^n$.

Δ-Gen: Same as in KwAH1.

Update: For given $usk_{i-1} = \langle usk'_{i-1}, s_3\rangle$, hsk_i and i, **Update** algorithm:

1. computes $usk'_i = usk'_{i-1} \cdot hsk_i^{s_3}$,
2. deletes usk'_{i-1} and hsk_i,
3. outputs $usk_i = \langle usk'_i, s_3\rangle$.

Encrypt: Same as in KwAH1.

Decrypt: For given $usk_i = \langle usk'_i, s_3\rangle$ and $C = \langle i, c_0, c_1\rangle$, **Decrypt** algorithm does the same as that of KwAH1 except that it:

1. computes $W' = e(c_0, usk'_i)$.

The correctness of this scheme can be done in the same way as KwAH1.

5.2.2 KwAH2′: Strongly IND-KE-CPA Construction

The second strongly IND-KE-CPA scheme KwAH2′ is based on KwAH2. It consists the following algorithms:

KeyGen: Given a security parameter k, **KeyGen** algorithm does the same as that of KwAH1 except that it:

2. picks random, $s_1, s_2, s_3 \in \mathbb{Z}_q^*$, and sets $h = g^{(s_2+s_1)s_3}$,
5. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h, G, H\rangle$, $mk = s_1$, $ak = s_2$, $usk' = g^{s_2}$ and $usk_0 = \langle d_{-1}^{s_3} \cdot d_0^{s_3}, s_3\rangle$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1^* \times \{0,1\}^n$.

Δ-Gen: Same as in KwAH2

Update: For given $usk_{i-1} = \langle usk'_{i-1}, usk'^{s_3}, s_3\rangle$, hsk_i and i, **Update** algorithm:

1. computes $usk'_i = usk'_{i-1} \cdot hsk_i^{s_3}$,
2. deletes usk'_{i-1} and hsk_i,
3. outputs $usk_i = \langle usk'_i, usk'^{s_3}, s_3\rangle$.

Encrypt: Same as in KwAH2

Decrypt: Same as in KwAH2

The correctness of this scheme can be done in the same way as KwAH2.

5.2.3 Trade-offs between Two Strongly IND-KE-CPA schemes

Developed from the two IND-KE-CPA schemes, these two strongly IND-KE-CPA schemes still remain trade-offs between public key length and ciphertext size, as well as between the costs of encryption and decryption computations, as discussed above for the IND-KE-CPA schemes. This fact helps to provide the flexibility when implementing these schemes.

5.2.4 Security and Proofs

We prove that KwAH1′ and KwAH2′ are strong-IND-KE-CPA under CBDH assumption in the random oracle model. Even when two helper keys are exposed, KwAH1′ and KwAH2′ are still secure as long as there is none query of private key.

Theorem 3. *(Informal) Suppose CBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and hash functions G and H are random oracles. Then, KwAH1′ and KwAH2′ are strong-IND-KE-CPA secure.*

Proof: We give the basic idea of proof for this theorem. We can construct an algorithm $\mathcal{A}_{cbdh}$ that can solve the CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ by using an adversary $\mathcal{A}_{kwah}$ that breaks strong-IND-KE-CPA security of our scheme. The algorithm $\mathcal{A}_{cbdh}$ is given an instance $\langle g, g^a, g^b, g^c\rangle$ in $\mathbb{G}_1^4$ from the challenger and tries to output $e(g,g)^{abc}$ using $\mathcal{A}_{kwah}$. Let $g_1 = g^a$, $g_2 = g^b$, and $g_3 = g^c$. There are three cases

needed to be considered:

1. The adversary will never query for main helper key or auxiliary helper key. ($\mathcal{COIN}$=0)
2. The adversary will query just one of the two helper keys. ($\mathcal{COIN}$=1)
3. The adversary will query both helper keys (of course, he can not query any secret key in this case). ($\mathcal{COIN}$=2)

For each case, we build the simulation, responses for the adversary's queries. For the first two cases, we can do the same as in theorem 1. Here, we consider the case of two helper keys exposure. We construct the responses of $\mathcal{A}_{cbdh}$ for KwAH1′ as follows:

Setup: $\mathcal{A}_{cbdh}$ picks a random $s, s_1 \in \mathbb{Z}_q^*$. Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{kwah}$ the system parameter

$$pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H \rangle,$$

where $h_1 = g_1$, $h_2 = g_1^s$ and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below. Here, it is noticed that $h_1 = g^{s_1 \cdot s_3}$, $s_2 = s \cdot s_1$.

G-queries: Same as in case $\mathcal{COIN}$=0 in theorem 1.

H-queries: Same as in case $\mathcal{COIN}$=0 in theorem 1.

Challenge: Same as in case $\mathcal{COIN}$=0 in theorem 1.

Exposure queries: The $\mathcal{A}_{cbdh}$ responds are as follows:

1. If $\mathcal{A}_{kwah}$ issues a query of private key, then $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If $\mathcal{A}_{kwah}$ issues a query of "main helper key" then $\mathcal{A}_{cbdh}$ outputs s_1.
3. If $\mathcal{A}_{kwah}$ issues a query of "auxiliary helper key" then $\mathcal{A}_{cbdh}$ outputs $s \cdot s_1$.

Guess: Same as in case $\mathcal{COIN}$=0.

The advantage of $\mathcal{A}_{cbdh}$ in each simulation can be computed in the same way as the proof of Theorem 1.
As noted before, in cases $\mathcal{COIN}$=0, $\mathcal{COIN}$=1, and $\mathcal{COIN}$=2, simulations are perfect until $\mathcal{A}_{cbdh}$ submits W^* as G-query if $i^* = \alpha$ and $\mathcal{A}_{cbdh}$ does not abort. Hence the views of $\mathcal{A}_{kwah}$ are identical for these three cases.
Simulation for KwAH2′ can be done in the same way, using proof for KwAH2. □

Chapter 6

Chosen Ciphertext Secure Construction

In this chapter, we construct chosen ciphertext secure KwAH schemes by extending KwAH1 and KwAH2 with Fujisaki-Okamoto padding [15, 16]. It should be noticed that the proofs of security of our schemes cannot be straightforwardly done since the model of KwAH significantly differs from the standard public key encryption. The results of this chapter were written in publication P1, P2, P3.

6.1 Construction

We give constructions of the two IND-KE-CCA schemes, developed from KwAH1 and KwAH2.

6.1.1 KwAH3: IND-KE-CCA Construction

The first IND-KE-CCA scheme KwAH3 is developed from KwAH1. Let F, G, H be cryptographic hash functions that F: $\{1, ..., N\} \times \{0,1\}^n \times \{0,1\}^\lambda \longrightarrow \mathbb{Z}_q^*$ and G: $\mathbb{G}_2 \rightarrow \{0,1\}^{n+\lambda}$. H is the same as in KwAH1. The KwAH3 scheme consists of the following algorithms:

KeyGen: Same as that of KwAH1 except that it:

3. choose cryptographic hash functions F, G, H.
5. Outputs $pk = (q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, F, G, H)$, $mk = s_1$, $ak = s_2$ and $usk_0 = d_{-1} \cdot d_0$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1 \times \{0,1\}^{n+\lambda}$.

Δ-Gen: Same as in KwAH1.

Update: Same as in KwAH1.

Encrypt: For given pk, i, and a message $M \in \{0,1\}^n$, assuming that $m \cdot \ell + 1 \leq i \leq (m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, **Encrypt** algorithm:

1. chooses random $R \in \{0,1\}^\lambda$,
2. computes $\sigma = F(i, M, R)$,
3. computes $W = (e(h_1, H(i)) \cdot e(h_2, H(m \cdot \ell)))^\sigma$ if $i \neq (m+1) \cdot \ell$,
4. computes $W = (e(h_1, H(i-1)) \cdot e(h_2, H((m + 1) \cdot \ell)))^\sigma$ if $i = (m+1) \cdot \ell$,
5. sets $C = \langle i,\ g^\sigma,\ G(W) \oplus (M||R) \rangle$, and
6. outputs C as a ciphertext.

Decrypt: Given pk, usk_i and $C = \langle i, c_0, c_1 \rangle$, **Decrypt** algorithm:

1. output $\perp$ if $C \notin \mathbb{Z}_N \times \mathbb{G}_1 \times \{0,1\}^{n+\lambda}$,
2. computes $W' = e(c_0, usk_i)$,

3. computes $(M'||R) = c_1 \oplus G(W')$, and
4. outputs M' as a plaintext if $c_0 = g^{\sigma'}$, or $\perp$ otherwise, where $\sigma' = F(i, M', R')$.

The correctness of this scheme can be done in the same way as KwAH1.

6.1.2 KwAH4: IND-KE-CCA Construction

The second IND-KE-CCA scheme KwAH4 is developed from KwAH2. Here functions F, G, H are the same as in KwAH3. The scheme consists of the following algorithms:

KeyGen: Same as that of KwAH2 except that it:

2. Picks $s_1, s_2 \in \mathbb{Z}_q^*$ uniformly at random, and sets $h = g^{s_1+s_2}$,
3. choose cryptographic hash functions F, G, H.
5. Outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h, F, G, H\rangle$, $mk = s_1$, $ak = s_2$, $usk' = g^{s_2}$ and $usk_0 = d_{-1} \cdot d_0$. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{Z}_N \times \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{n+\lambda}$.

Δ-Gen: Same as in KwAH2.

Update: Same as in KwAH2.

Encrypt: For given pk, i, and a message $M \in \{0,1\}^n$, assuming that $m \cdot \ell + 1 \leq i \leq (m+1) \cdot \ell$ for some m $(0 \leq m \leq n-1)$, **Encrypt** algorithm:

1. chooses random $R \in \{0,1\}^\lambda$,
2. computes $\sigma = F(i, M, R)$,
3. computes $W = (e(h, H(i)))^\sigma$
4. computes $h_3 = (H(i) \cdot H(m \cdot \ell)^{-1})^\sigma$ if $i \neq (m+1) \cdot \ell$,
5. computes $h_3 = (H(i) \cdot H(i-1)^{-1})^\sigma$ if $i = (m+1) \cdot \ell$,
6. sets $C = \langle i,\ h_3,\ g^\sigma,\ G(W) \oplus (M||R)\rangle$, and
7. outputs C as a ciphertext.

Decrypt: Given pk, usk_i and $C = \langle i, h_3, c_0, c_1\rangle$, **Decrypt** algorithm:

1. output $\perp$ if $C \notin \mathbb{Z}_N \times \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{n+\lambda}$,
2. computes $W' = e(c_0, usk_i) \cdot e(usk', h_3)$ if $i \neq (m+1) \cdot \ell$,
3. computes $W' = e(c_0, usk_i) \cdot e(h \cdot usk'^{-1}, h_3)$ if $i = (m+1) \cdot \ell$,
4. computes $(M'||R) = c_1 \oplus G(W')$, and
5. outputs M' as a plaintext if $c_0 = g^{\sigma'}$, or $\perp$ otherwise, where $\sigma' = F(i, M', R')$.

The correctness of this scheme can be done in the same way as KwAH2.

6.1.3 Trade-offs between Two IND-KE-CCA Schemes

These two IND-KE-CCA schemes are developed from the two IND-KE-CPA schemes given in the previous section. As the discussion above for the IND-KE-CPA schemes, in IND-KE-CCA schemes, the trade-off between the public key length and ciphertext size remains unchanged, as well as the cost of decryption and encryption algorithms.

6.2 Security and Proofs

We can prove that KwAH3 and KwAH4 are IND-KE-CCA under the CBDH assumption in the random oracle model.

Theorem 4. *Suppose* $(t_{cbdh}, \epsilon_{cbdh})$*-CBDH assumption holds in* $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ *and hash functions* G *and* H *are random oracles. Then, KwAH3 and KwAH4 is* $(t_{kwah}, \epsilon_{kwah})$*-IND-KE-CCA secure as long as:*

$$\epsilon_{kwah} \leq \frac{3q_G N}{2}\epsilon_{cbdh} + \frac{2q_F}{2^\lambda} + \frac{2q_D}{q}$$
$$t_{kwah} \leq t_{cbdh} + \Theta(\tau(5q_F + q_H + 3q_E + 5q_D)),$$

where IND-KE-CCA adversary $\mathcal{A}_{kwah}$ *issues at most* q_H *H-queries,* q_G *G-queries,* q_F *F-queries,* q_D *Decryption queries and* q_E *Exposure queries. Here,* τ *is the maximum time for computing an exponentiation in* $\mathbb{G}_1, \mathbb{G}_2$*, and pairing* e*.*

Proof: The proof is almost identical to Theorem 1 except that here, $\mathcal{A}_{cbdh}$ has to simulate responses to Decryption queries as well. We show the general idea to prove KwAH3. The proof for KwAH4 can be done similarly. For both cases for $\mathcal{COIN}$=0 and 1, if $(i \neq \alpha)$ then it will be easy for $\mathcal{A}_{cbdh}$ to calculate usk_i on his own, so the decryption will be easily done as well. Therefore we only need to consider the case for $i = \alpha$. The simulation can be as follows:

F-queries: $\mathcal{A}_{kwah}$ picks a random $R^* \in \{0,1\}^\lambda$ in advance. It issues up to q_F queries to the random oracle F. To respond to these queries, algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $C = \langle i, M, R, \sigma \rangle$ as explained below. We call this list F_{list}. The list is initially empty. When $\mathcal{A}_{kwah}$ gives $\mathcal{A}_{cbdh}$ a query $\langle i, M, R \rangle$ to the oracle F, $\mathcal{A}_{cbdh}$ responds as follows:

1. If $R = R^*$, $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If the query $\langle i, M, R \rangle$ already appears on the F_{list} in a tuple $\langle i, M, R, \sigma \rangle$

then outputs
$F(i, M, R) = \sigma$.

3. $\mathcal{A}_{cbdh}$ chooses a random $\sigma \in \mathbb{Z}_q^*$.
4. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, M, R, \sigma \rangle$ to the F_{list} and outputs $F(i, M, R) = \sigma$.

Challenge: same as in the KwAH1 except that the challenge ciphertext $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as $c_0^* = g_3$ and $c_1^* = \mu$ where $\mu \in_R \{0,1\}^{n+\lambda}$

Decryption: $\mathcal{A}_{kwah}$ issues up to q_D Decryption queries. When $\mathcal{A}_{kwah}$ gives a query $C = (i, c_0, c_1)$ $\mathcal{A}_{cbdh}$ responds as follows:

1. if $i \neq \alpha$ $\mathcal{A}_{cbdh}$ runs the algorithm to respond to Exposure queries to obtain usk_i, decryptc C, and outputs the decryption results to $\mathcal{A}_{kwah}$.
2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ searches for a tuple (α, M, R, σ) from F_{list} such that

$$c_0 = g^{\sigma},$$
$$c_1 = \begin{cases} G((e(h_1, H_{\alpha-1}).e(h_2, H_\alpha))^{\sigma}) \oplus (M||R) \\ \qquad \text{if } \alpha \equiv 0 \bmod \ell \\ G((e(h_1, H_\alpha).e(h_2, H_{m\cdot\ell}))^{\sigma}) \oplus (M||R) \\ \qquad \text{if } \alpha \not\equiv 0 \bmod \ell \end{cases}$$

3. If there exists such a tuple, $\mathcal{A}_{cbdh}$ outputs M to $\mathcal{A}_{kwah}$. Otherwise outputs $\perp$.

Responses to G-queries, H-queries and Exposure queries can be simulated similarly to the proof in theorem 1. With the same way of computation in Theorem 1, we can have:

$$p_0 \geq \frac{2}{q_G N}(\epsilon_{kwah} - \lambda \Pr[E_{msk}] - \Pr[F\text{-}Fail] - \Pr[D\text{-}Fail])$$

$$p_1 \geq \frac{1}{q_G N}(\epsilon_{kwah} - \eta \Pr[\neg E_{msk}] - \Pr[F\text{-}Fail] - \Pr[D\text{-}Fail])$$

where the F-$Fail$ is an event assigned to be true if and only if there exists an F-query (i, M, R) such that $R = R^*$, D-$Fail$ is an event assigned to be true if and only if $\mathcal{A}_{cbdh}$ return $\perp$ for a Decryption query which should no be rejected.

Here we need to calculate $\Pr[F\text{-}Fail]$ and $\Pr[D\text{-}Fail]$. Since it is information-theoretically impossible to obtain any information on R^*, $\mathcal{A}_{kwah}$ submits R^* as in one of F-queries with probability at most $q_F/2^{\lambda}$. $\mathcal{A}_{cbdh}$ fails to respond to a Decryption query only when $\mathcal{A}_{kwah}$ succeeds to generate a ciphertext $C = Encrypt(pk, \alpha, M; R)$ without submitting an F-query (α, M, R). Hence, $\Pr[D\text{-}Fail]$ will be at most q_D/q. By letting $\delta = 1/3$ we have the bound of ϵ_{cbdh} is the same as written in Theorem 4.

The bound of $\mathcal{A}_{cbdh}$'s running time is easily seen from the proof. □

Chapter 7

Conclusions

In this thesis, we gave some cryptographic encrypion schemes of key insulated public key encryption, using an auxiliary helper keys. All of the schemes are proven to be secure, based on the ability to access the random oracles.

By using an auxiliary helper key besides of the normal helper key, the security of the system can be more protected, in the meaning of reducing the damage when the secret key is exposed. The damage, compared with the previous works, becomes much smaller. Nevetherless, the use of auxiliary helper key is easy, which allows the general users without the professional knowledge of security, can use the system conveniently.

We also gave an example of using our schemes in the real life.

We gave two different schemes to provide the flexibility for the system builder. Furthermore, we proposed the schemes of chosen ciphertext attacks which have have the strongest security. All the proposed schemes are proven to be secure under the CBDH assumption.

The security of our schemes are proven in the random oracles, which might make us worry to use them in the real life. There is an open problem to update our schemes to be secure in the standard model.

Bibliography

[1] R. Anderson, "Two remarks on public key cryptology," Invited Lecture, ACM CCCS'97, available at `http://www.cl.cam.ac.uk/users/rja14/`.

[2] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," Proc. of Asiacrypt'00, LNCS 1976, Springer-Verlag, pp. 116-129, 2000.

[3] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," Proc. of Eurocrypt'04, LNCS 3027, Springer-Verlag, pp.223-238, 2004.

[4] D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," Proc. of Crypto'04, LNCS 3152, Springer-Verlag, pp.443-459, 2004.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," Proc. of Crypto'01, LNCS 2139, Springer-Verlag, pp.213-229, 2001.

[6] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," SIAM J. of Computing, vol. 32, no. 3, pp.586-615, 2003 (full version of [5]).

[7] M. Bellare and A. Palacio, "Protecting against key exposure: strongly key-insulated encryption with optimal threshold," available at `http://eprint.iacr.org/2002/064/` .

[8] M. Bellare and S.K. Miner, "A forward-secure digital signature scheme," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp. 431-448, 1999.

[9] C. Cocks, "An identity based encryption scheme based on quadratic residues," Proc. of IMA Int. Conf. 2001, Coding and Cryptography, LNCS 2260, Springer-Verlag, pp. 360-363, 2001.

[10] R. Canetti, S. Halevi and J. Katz, "A forward secure public key encryption scheme," Proc. of Eurocrypt'03, LNCS 2656, Springer-Verlag, pp.255-271, 2003.

[11] Y. Dodis, J. Katz, S. Xu and M. Yung, "Key-insulated public key cryptosystems," Proc. of Eurocrypt'02, LNCS 2332, Springer-Verlag, pp.65-82, 2002.

[12] Y. Dodis, J. Katz, S. Xu and M. Yung, "Strong key-insulated signature schemes," Proc. of PKC'03, LNCS 2567, Springer-Verlag, pp.130-144, 2003.

[13] Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "Intrusion-resilient public-key encryption," Proc. of CT-RSA'03, LNCS 2612, Springer-Verlag, pp.19-32, 2003.

[14] Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "A generic construction

for intrusion-resilient public-key encryption," Proc. of CT-RSA'04, LNCS 2964, Springer-Verlag, pp.81-98, 2004.

[15] E. Fujisaki and T. Okamoto, "How to enhance the security of public-key encryption at minimum cost," Proc. of PKC'99, LNCS 1560, Springer-Verlag, pp.53-68, 1999.

[16] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.537-554, 1999.

[17] G. Hanaoka, Y. Hanaoka and H. Imai, "Parallel key-insulated public key encryption," Proc. of PKC'06, LNCS 3958, Springer-Verlag, pp.105-122, 2006.

[18] Y. Hanaoka, G. Hanaoka, J. Shikata and H. Imai, "Identity-based hierarchical strongly key-insulated encryption and its application," Proc. of Asiacrypt'05, LNCS 3958, Springer-Verlag, pp.495-514, 2005.

[19] Y. Hanaoka, G. Hanaoka and H. Imai,"Forward Secure Encryption, Key-Insulated Encryption and Their Hybrid Schemes," SITA2004.

[20] C. Gentry and A. Silverberg, "Hierarchical id-based cryptography," Proc. of Asiacrypt'02, LNCS 2501, Springer-Verlag, pp.548-566, 2002.

[21] G. Itkis and L. Reyzin, "SiBIR: signer-base intrusion-resilient signatures," Proc. of Crypto'02, LNCS 2442, Springer-Verlag, pp.499-514, 2002.

[22] J.Katz, "Lecture note on introduction to cryptography", available at www.cs.umd.edu/j̃katz/TEACHING/crypto_F02/lectures.html.

[23] A. Menezes, P.V. Oorschot, S. Vanstone, "Handbook of Applied Cryptography," CRC Press, 1997.

[24] T. Okamoto and D. Pointcheval, "The gap-problems: a new class of problems for the security of cryptographic schemes," Proc. of PKC'01, LNCS 1992, Springer-Verlag, pp.104-118, 2001.

[25] J. Pollard, "Theorems of factorization and primality testing", Proceedings of Cambridge Philosophical Society 76 (1974), 521-528

[26] M.J.B. Robshaw and Y.L. Yin, "Elliptic Curve Cryptosystems", Technical Note, RSA Laboratories, 1997.

[27] A. Shamir, "Identity-based cryptosystems and signature schemes," Proc. of Crypto'84, LNCS 196, Springer-Verlag, pp.47-53, 1984.

[28] S.R. Stinson, "Cryptography: Theory and Practice," 2nd Edition, CRC Press, Inc 2002.

[29] B. Waters, "Efficient identity based encryption without random oracles," Proc. of Eurocrypt'05, LNCS 3494, Springer-Verlag, pp.114-127, 2005.

Publications

With Reviews

P1. "Key-Insulated Public Key Encryption with Auxiliary Helper Key: Model, Constructions and Formal Security Proofs"; Thi Lan Anh Phan, Goichiro Hanaoka, Kanta Matsuura, Hideki Imai; IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Special Section on Information Theory and Its Application, September 2007; Vol.E90-A, No.9, p.1814-1829.

P2. "Reduce the spread of key exposure in Key-Insulated Public Key Encryption"; Thi Lan Anh Phan, Yumiko Hanaoka, Goichiro Hanaoka, Kanta Matsuura, Hideki Imai; First International Conference on Cryptography in Viet-Nam; September, 2006; HaNoi, VietNam; LNCS4341 p.366-384.

Without Reviews

P3. "A New Key-Insulated Public Key Encryption Scheme with Auxiliary Helper Key"; Thi Lan Anh Phan, Goichiro Hanaoka, Kanta Matsuura, Hideki Imai; Proceeding of the 29th Symposium of Information Theory and Its Application; October 2006, Hakodate, p.77-80.

P4. "Formal Security Proofs of Key-Insulated Public Key Encryption with Auxiliary Helper Key"; Thi Lan Anh Phan, Goichiro Hanaoka, Kanta Matsuura, Hideki Imai; Symposium on Cryptography and Information Security; January 2007; Nagasaki, CD-proceeding.