2016

Contribution to functional encryption through encodings

Jongkil Kim
University of Wollongong

Recommended Citation
Contribution to Functional Encryption through Encodings

Jongkil Kim

Supervisor:
Professor J. Seberry
Co-supervisors:
Professor W. Susilo & A. Professor M. H. Au

This thesis is presented as part of the requirements for the conferral of the degree:

Ph. D

The University of Wollongong
School of Computing and Information Technology

April 2016
Declaration

I, Jongkil Kim, declare that this thesis submitted in partial fulfilment of the requirements for the conferral of the degree Ph. D, from the University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualifications at any other academic institution.

__________________________

Jongkil Kim
September 14, 2016
Abstract

We introduce novel techniques to achieve a wide range of functional encryption schemes. For our work, we explore a number of functional encryption schemes and observe their techniques to achieve adaptive security. Based on our observations, we develop several techniques to enable functional encryption schemes. Particularly, we utilize encoding frameworks to generalize our contribution which is applicable to a number of functional encryption schemes. We utilize the pair encoding framework (Eurocrypt’14) for our work, but we also introduce a new encoding framework to achieve efficient functional encryption schemes.

Firstly, using the pair encoding framework, we show that functional encryption schemes can be converted into their dual conversion without any efficiency loss. The dual conversion means the conversion of any scheme into the symmetric scheme in which the roles are swapped between private keys and ciphertexts (e.g. ciphertext policy attribute based encryption and key policy attribute based encryption). Additionally, we provide a new construction in prime order groups for pair encodings. This construction improves the efficiency of previous constructions for pair encodings since it realizes functional encryption schemes which were only suggested in composite order groups into prime order groups.

As a new encoding framework, we introduce a tag based encoding which is more efficient than previous encodings when the size of predicate is large. It supports a number of functional encryptions. Also, generic constructions for tag based encodings are provided. In particular, they are adaptively secure in prime order groups under the standard assumption or static assumptions. Moreover, key policy attribute based encryption schemes are followed. These schemes share the technique of the tag based encoding but these schemes provide semi-adaptive security which is weaker than adaptive security. However, these schemes show many desirable properties such as multi-use of attribute, short ciphertexts and support large universe under the standard assumption in prime order groups. Therefore, those schemes complement tag based encoding.
Publications

This thesis is related following publications/manuscripts.


5. Jongkil Kim, Willy Susilo, Man Ho Au, Fuchun Guo: KP-ABE with Short Ciphertexts in Prime Order Groups under Standard Assumption (Without DPVS) (Submitted)

6. Jongkil Kim, Willy Susilo, Man Ho Au, Fuchun Guo, Jennifer Seberry: Masked Form of Pair Encodings: Refined Duality for Pair Encodings without An Efficiency Loss (Submitted)
Acknowledgments

I am thankful to my supervisors, Prof. Jennifer Seberry and Prof. Willy Susilo for the time and support you offered me. In particular, I am really appreciate to you since you gave me an opportunity to start my study in cryptography and always encouraged me to go forward academically. I am also thankful to A. Prof. Manho Au and Dr. Fuchun Guo. I learned many things which would be very valuable for my future from your advice and guidance. I also wish to thank all my colleagues and friends I met in the university of Wollongong. I wish to thank my wife, Sujung Lee, for her support and sacrifice.
# Contents

Abstract iii
Publications iv
Acknowledgments v

## 1 Introduction
1.1 Functional Encryption 1
1.2 Encodings for Functional Encryption 3
1.3 Contribution 5
  1.3.1 Applications 5
1.4 Thesis Organization 6
  1.4.1 Refined Duality of the Pair Encoding Framework 7
  1.4.2 The Pair Encoding in Prime Order Groups 8
  1.4.3 Tag Based Encoding 8
  1.4.4 KP-ABE with Short Ciphertexts 8

## 2 Background
2.1 Preliminaries 10
  2.1.1 Bilinear Maps 10
  2.1.2 Access Structures 10
  2.1.3 Complexity Assumptions 11
  2.1.4 Functional Encryption 13
2.2 Related Work 15
  2.2.1 Difficulty in Functional Encryption 15
  2.2.2 Dual System Encryption 16
  2.2.3 Doubly Selective Security 17
  2.2.4 Techniques for Prime Order Groups 19
2.3 Encoding Frameworks 21
  2.3.1 Overview of the Pair Encoding Framework 22
  2.3.2 Computationally $\alpha$ hiding 24
2.4 Notations 25
CONTENTS

3 Refined Duality of Pair Encoding Schemes 26
  3.1 Attrapaundg and Yamada’s Technique [AY15] 26
  3.2 A Masked Form 27
  3.3 Masked Forms of Existing Schemes 30

4 Functional Encryption in Prime order groups 34
  4.1 Our Construction (Compiler) 37
  4.2 Security Analysis 38
    4.2.1 Semi-functional Ciphertext Invariance 40
    4.2.2 Semi-functional Key Invariance 44
    4.2.3 Semi-functional Security 50

5 Tag Based Encoding 53
  5.1 Complexity Assumptions 56
  5.2 Tag based encoding 58
  5.3 Our Compiler 59
    5.3.1 The Construction 59
    5.3.2 Security Analysis 61
  5.4 Our Compiler with Symmetric Bilinear Maps 66
    5.4.1 The Construction 66
    5.4.2 Security Analysis 68
  5.5 Our Efficient Compiler under Static Assumptions 72
    5.5.1 Our Construction 72
    5.5.2 Security Analysis 74
  5.6 New Schemes 78
  5.7 Instances from Existing Schemes 80

6 KP-ABE with short ciphertexts 82
  6.1 KP-ABE in a small universe 84
    6.1.1 Construction 85
    6.1.2 Security Analysis 87
  6.2 KP-ABE with short ciphertexts in a large universe 92
    6.2.1 Construction 93
    6.2.2 Security Analysis 95

7 Conclusion and Future Works 104

Bibliography 107

Appendix A 118
  A.1 A Dedicated Construction for Masked Forms 118
List of Tables

1.1 Comparison of Encoding Frameworks ......................... 4
1.2 Comparison between Previous Works and Ours ............... 6

2.1 Summary of Dual System Encryption .......................... 16
2.2 Summary of Doubly Selective Security ....................... 18
2.3 Summary of Our Notations .................................. 25

3.1 Comparison between AY’s Conversion and Ours in Composite Order Groups ................................................................. 30
3.2 Comparison between AY’s Conversion and Ours in Prime Order Groups ................................................................. 31

4.1 Comparison between Previous Works and Ours .................. 36
4.2 Summary of Semi-Functional Ciphertext Invariance .......... 41
4.3 Summary of Semi-functional Key Invariance .................. 45

5.1 Efficiency Comparison between Our Compiler and CGW’s .... 54
5.2 Efficiency Comparison of IPE Schemes Based on Encodings .... 55
5.3 Efficiency Comparison of PA-IPE Schemes Based on Encodings .... 55

6.1 Comparison of KP-ABE Schemes Having Constant Sized Ciphertexts 85
6.2 Summary of Security Games ............................... 97
List of Figures

2.1 Concept of Encoding frameworks .................................. 21
4.1 Results of Our Compiler ............................................ 35
5.1 Encoding Frameworks for FE ........................................ 53
Chapter 1

Introduction

1.1 Functional Encryption

Functional encryption (FE) is a public key cryptographic system which offers a fine-grained access control. Functional encryption is rooted to Identity Based Encryption (IBE) [Sha84, BF01, Coc01] and achieves access controls based on various types of functions using public keys. Subsequently, more complicated functions, such as boolean functions [SW05, GPSW06, LOS10], inner products [KSW08, OT09, AFV11], deterministic finite automata [Wat12] and turing machine [AS16] are embedded into functional encryption to support complex systems. One of the applications of functional encryption is a large scale IT systems (e.g. cloud storage). In a large scale IT system, efficiency and flexibility remain as critical issues, which are difficult to achieve concurrently using traditional public key encryption.

In a functional encryption, a user does not own a unique key. Instead, multiple key elements are created based on the user’s properties. Then, ciphertexts can be decrypted using not a single key element but the combination of those key elements as inputs of the function which the encryption schemes aims for. Therefore, in a functional encryption, key elements can appear in both keys that the adversary possesses and the other keys that decrypt the challenge ciphertext at the same time. It means that key spaces to simulate compromised keys are not well separated from the keys which are secure. In detail, private keys which decrypt the challenge ciphertext must be excluded from the other keys that the adversary can obtain in traditional partitioning techniques [BB04a, CHK03, BF01, BBG05].

There exists functional encryption scheme which is secure from the adaptive adversary where the adaptive adversary implies the adversary who can collect keys before and after it decides the target ciphertext. Also, some of those adaptively secure functional encryptions are proved via these partitioning technique [BB04b, Wat05, GW09]. However, with the partitioning technique, selective security of func-
CHAPTER 1. INTRODUCTION

Traditional encryption was more often considered due to the difficulty of proving adaptive securities of functional encryption schemes. In a selective security model, the adversary declares its target only before a system sets up. In this model, since the simulator always knows the target of the adversary, it can separate a key space in which keys decrypt the target from the other key spaces when it creates public keys. This allows traditional partitioning techniques to be applicable to functional encryption schemes.

Nevertheless, when functions which encryption schemes employ are complicated, proving adaptive security of those encryptions using the traditional partitioning techniques becomes very daunting. A well-known example of functional encryption is Attribute Based Encryption (ABE) [SW05]. ABE has refined as two primitives Key Policy Attribute Based Encryption (KP-ABE) and Ciphertext Policy Attribute Based Encryption (CP-ABE) based on where an access policy is located in [GPSW06]. In KP-ABE, a user has a policy and ciphertexts are created for a set of attributes. In CP-ABE, a user’s key created based on his/her attributes and the ciphertext employs a policy. Attribute based encryption shows the difficulty of proving adaptive security of functional encryption. For example, in CP-ABE, keys are created based on a user’s attributes. Therefore, each user has multiple key elements which represent the user’s attributes. In CP-ABE, ciphertexts are created based on the policy allowing access to the encrypted message. Therefore, if a user has attributes satisfying a policy of some ciphertexts, the messages encrypted in the ciphertexts are revealed to the user. However, other users who cannot meet the policy also may have some attributes that appear in the ciphertexts. For instance, a user possesses attributes "A" and "B" and a ciphertext created for a boolean formula "A AND C". Although "A" appears both in the user’s key and the ciphertext, the ciphertext must not be revealed any information about the encrypted message to the user. Therefore, attribute "A" cannot be belonging to any key spaces when the simulator separates key space. This makes proving CP-ABE scheme based on traditional partitioning technique more difficult.

Waters [Wat09] introduced the dual system encryption which provides a breakthrough technique of proving adaptive security of functional encryption. An adaptive security model is more realistic than the selective security model. In the adaptive security model, the adversary does not declare its target in advance. To achieve adaptive security, the dual system encryption implements auxiliary types of keys and ciphertexts, namely semi-functional keys and semi-functional ciphertexts, appearing only in the security proof. Subsequently, it must be proved that a security game consisting only of semi-functional keys and semi-functional ciphertexts is indistinguishable from the original security game which uses only normal keys and normal ciphertexts. Since semi-functional keys cannot decrypt semi-functional ciphertexts,
the security analysis of the transformed game becomes much easier than that of
the original game. Waters showed that the dual system encryption is a powerful
tool in public key encryptions and signatures by introducing a number of adaptive
equation schemes.

More recently, Lewko and Waters [LW12] introduced another break-through
technique combining the traditional partitioning techniques and Waters’ dual system
equation [Wat09]. In the dual system encryption, all keys and ciphertexts have
two types, i.e. normal and semi-functional, and proving the invariance of both types
of keys is critical. In the proof of the invariance, a semi-functional key correlates
with the ciphertext to hide its type to the simulator, but at the same time, this
correlation must be hidden to the adversary. Originally, Waters solved this problem
by information theoretic arguments. Therefore, the correlation is hidden perfectly
to the adversary. However, Lewko and Waters showed that this paradox is also
achievable using two selective security proofs which are reduced to computational
assumptions. The technique is often known as doubly selective security\footnote{It also called as computational hiding [Att14a, Wee14] since it hides the correlation computationally} [Att14a,
AY15] because computational assumptions are originated from their selective and
co-selective security. The doubly selective security allows FE to have more desirable
properties and support to more complicated functions. For example, it achieves
adaptively secure ABE scheme allowing multi-use of attributes in [LW12].

1.2 Encodings for Functional Encryption

In this work, we present new techniques to achieve adaptive security of functional
encryption. In particular, we utilize encoding frameworks [Att14a, Wee14] to present
our contribution to functional encryption.

Encoding frameworks provide a new direction of proving security since one can
prove an adaptive security of functional encryption schemes by only showing that the
schemes satisfy the properties required by the encodings. Therefore, the encoding
frameworks reduce an effort of proving the securities of functional encryptions, and
also provide a new insight of properties leading to adaptively secure encryption
schemes. Simply speaking, checking a few properties of schemes can result in the
full proof of adaptive security of an encryption scheme when the encoding framework
used to construct a new scheme.

Moreover, utilizing encoding frameworks is a very efficient way to describe func-
tional encodings because a contribution to an encoding framework takes effects on
all functional encryption schemes described as encodings. For example, in Chapter
CHAPTER 1. INTRODUCTION

Table 1.1: Comparison of Encoding Frameworks

<table>
<thead>
<tr>
<th></th>
<th>Composite Order Groups</th>
<th>Prime Order Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect hiding*a</td>
<td>Wee [Wee14], CGW [CGW15]</td>
<td>CGW [CGW15], AC [AC16]</td>
</tr>
</tbody>
</table>

*a If the correlations are hidden by information theoretic arguments as Waters’ dual system encryption, it is known as perfect hiding since the correlation is hidden perfectly.

*b Encodings for computational hiding are also applicable to perfect hiding when computational assumptions can be replaced by information theoretic arguments such as pair-wise independence.

*c This work introduced concurrently and independently with out works.

4, we introduce a new compilerb for pair encodings. In this chapter, we suggest only one compiler but this compiler implies multiple new schemes which are listed in table 4.1.

The dual system encryption and the doubly selective security are utilized as main tools to construct a number of functional encryption schemes. To utilize those techniques, functional encryption schemes are enforced to some common structures and properties. This causes coupling of functional encryption schemes, naturally. Encoding frameworks are introduced to observe those requirements of the dual system encryption (and doubly selective security) and formalize them using encodings. Two independent works [Att14a, Wee14] have been proposed by formalizing those common features of functional encryptions introduced. In particular, the pair encoding framework [Att14a] includes functional encryption schemes of which the adaptive securities are proved using doubly selective security.

More recently, several works [CGW15, AC16, Att15] improved the efficiency of compilers of encodings. They suggested compilers for existing encodings in prime order groups. Because previous compilers of the encodings are only suggested in composite order groups and composite order groups were significantly inefficient compared with prime order groups [LW12, Fre10, Gui13], those encoding make functional encryption more efficient and practical. Table 1.1 is provided to explain current achievements of encoding frameworks. Computational hiding implies the doubly selective security in the table.

The efficiency of functional encryption is an important issue since functional encryption is developed to support large systems which have complicated access policies. To support the large systems, encryption and decryption process should be performed rapidly. Therefore, achieving more efficient functional encryption schemes is important on-going problems. In particular, improving efficiency of functional

b It should be noted that the term “compiler” implies generic construction of encodings [CGW15]. It is named because of the role of a generic construction in encoding frameworks. It compiles encodings to encryption schemes.
encryption schemes using encoding framework is more meaningful since it can affect a number of functional encoding schemes at the same time.

The compilers from CGW [CGW15] and AC [AC16] show good efficiency in prime order groups but applying their techniques for doubly selective security is not clear. Therefore, encodings utilizing doubly selective security remains feasible only in composite order groups. Also, those compilers in prime order groups for encodings commonly have a certain limitation in an efficiency. Particularly, they require at least two prime order group elements to feature a composite order group element in schemes of Wee [Wee14] and Attrapadung [Att14a]. Breaking this limitation is an interesting and meaningful problem.

1.3 Contribution

In this work, we introduce techniques to construct efficient functional encryption schemes via encoding frameworks. In Chapter 3, we suggest a way to convert one FE scheme to its dual scheme without an efficiency loss where a dual scheme is the symmetric conversion in which the locations of predicates and descriptions of an original scheme are exchanged between keys and ciphertexts (e.g. KP-ABE and CP-ABE). Since the previous technique [AY15] requires an efficiency loss for this conversion, our technique instantly improves the efficiencies of the schemes which are derived by the previous technique. In Chapter 4, we introduce a new technique which is applicable to an existing encoding framework for doubly selective security. Using this technique, several existing schemes in composite order groups are featured into prime order groups which are about 254 times faster in computations [Gui13]. In Chapter 5, we suggest a new efficient encoding framework which can works with compilers breaking the previous efficiency boundaries. In Chapter 6, we describe that the technique in Chapter 5 is also applicable to more wider class of functional encryption. Since the techniques in this thesis are based on encodings, they are applicable to a number of functional encryptions. We summarize the encryption schemes achievable through our works in Table 1.2.

1.3.1 Applications

Since functional encryption supports a fine-grained access control. It can be widely used for real applications. Cloud storage service is popular and practical application since it provides more reliable storage with affordable price and mobility. Data in cloud storages can be shared easily with the other. However, the cloud service provider is basically a third party and often untrusted. Relying on the security of

\footnote{There is an independent and concurrent work [Att15] in this claim.}
data to an untrusted third part causes a risk to personal users or organisations. One of promising ways to protect data in untrusted cloud is encryption. In particular, functional encryption (e.g. ABE) is used to protect the data in cloud storage, but still maintain conveniences of the cloud services [WMZV16, LYZ+13, BBS+09, ZM13, NJM+]. One of good examples is O-Auth [Har12, TG11] which is an widely used open source authentication platform. It internally uses a functional encryption scheme for a complex access control. Functional encryption is also used for an access control of cloud storages. In those practical works, the efficiency and capability of functional encryption schemes are considered important since computational burden of functional encryption is the biggest obstacle to make them in practical applications. Therefore, making efficient functional encryption is critical issue in those practical applications.

### 1.4 Thesis Organization

In this thesis, we improve functional encryption schemes in three ways. First, we enhance the efficiency of encodings by themselves in Chapter 3. Next, we provide a generic construction which features encoding schemes in composite order groups into
prime order groups in Chapter 4. Finally, we introduce a new encoding framework which breaks current theoretic boundary of efficiency in Chapters 5 and 6.

In Table 1.2, one can see that the instances given in Chapter 3 also appear in Chapter 4. This is because we target to improve the pair encoding instances given in [Att14a, LW12] but with the different ways. We directly reduce the length of each encoding schemes in Chapter 3, but in chapter 4 we suggests a new compiler in prime order groups although the previous compliers are secure only in composite order groups.

Instances in Chapter 5 are quite different from those in Chapters 3 and 4 since we are suggesting a new encoding framework rather than using a existing encoding framework as in Chapters 3 and 4. The compiler in our new encoding is more efficient than that we suggested in Chapter 4, but it is only applicable to the subclass of encoding instances where the technique in Chapter 4 is applicable. However, we show that our technique can be extended to more complicated scheme such as KP-ABE under a weaker security model in Chapter 6.

1.4.1 Refined Duality of the Pair Encoding Framework

In Chapter 3, we introduce a new symmetric conversion technique for functional encryption schemes. In particular, our technique does not incur any loss in efficiency while it is converted and is applicable when an adaptive security of a functional encryption scheme is proved by the doubly selective technique. Because functional encryption schemes that were proved by doubly selective technique were mostly introduced as pair encoding schemes, we also present our results using the pair encoding framework. To describe our conversion more clearly, we newly introduce a masked form of pair encoding frameworks. A masked form requires specific structures for pair encoding schemes, but all pair encodings of which the adaptive securities were proved using doubly selective security exhibit this structure. In other words, all of them have their own masked forms, to the best of our knowledge. Using a masked form, one pair encoding scheme can be converted into its dual scheme, easily. Compared with the best conversion technique of functional encryption [AY15], ours reduces one element in public keys, private keys and ciphertexts of converted schemes in composite order groups [Att14a] when we apply our result to the generic constructions of a pair encoding. Additionally, in prime order groups [Att15], our result reduces three group elements in public keys, private keys and ciphertexts. These savings are applicable for all functional encoding schemes proved by doubly selective technique.
1.4.2 The Pair Encoding in Prime Order Groups

In Chapter 4, a new construction for FE schemes is introduced in prime order groups. To show that our construction is widely applicable and supports computational hiding, we adopt the syntax of Attrapadung’s pair encoding framework [Att14a]. Therefore, our new construction can be considered as a new compiler of pair encodings. Particularly, previous compilers are either only equipped with composite order groups [Att14a, Wee14] or not supporting encoding schemes utilizing doubly selective security [CGW15, AC16]. Our compiler firstly, but concurrently, introduces a compiler of pair encoding schemes for doubly selective security in prime order groups. There is an independent work from Attrapadung [Att15]. Compared with [Att15], our compiler is constructed under different assumptions and requires smaller sized public keys although the size of other parameters is almost same. In this work, we utilize Lewko and Waters’ IBE scheme and expand it for more complicated function using a nested model of the dual system encryption.

1.4.3 Tag Based Encoding

In Chapter 5, we introduce a new encoding framework for functional encryption. We observed that common properties of functional encryptions [Wat09, AL12, CZF12, Wee14], and generalized them as tag based encoding using tag. Three adaptively secure generic constructions for the encoding are provided as compilers. Two constructions are secure under the standard decisional linear assumption. The other is secure under static assumptions. The construction under static assumption is most efficient but relies its security to stronger assumptions than the other constructions. Our constructions are adaptively secure with both in asymmetric and symmetric bilinear maps and practically improves the previous functional encryption presented by the encoding frameworks. To the best of our knowledge, our construction provides a first encoding framework with symmetric bilinear maps in prime order groups.

1.4.4 KP-ABE with Short Ciphertexts

In Chapter 6, we introduce new expressive semi-adaptively secure KP-ABE schemes in prime order groups under standard assumptions. The technique to achieve those schemes quite similar with our tag based encryption, but these schemes are semi-adaptively secure. In semi-adaptive model, the target of the adversary declare after system set-up. Therefore, a semi-adaptive security model is stronger than selective model, but weaker than the adaptive model. They are also considered as special cases of our tag based encoding achieving weaker security since the techniques resembles each other. Therefore, we obtain two semi-adaptively secure KP-ABE schemes
under the Decisional Linear Assumption (DLIN) as follows:

- KP-ABE for a small attribute universe; and
- KP-ABE with short ciphertexts for a large attribute universe.

We focus more on a semi-adaptive KP-ABE with short ciphertexts since it supports many desirable properties of ABE such as allowing multi-uses of attributes and a large attribute universe with short ciphertexts. However, we also introduce the first scheme a warming-up scheme of explaining our main technique.
Chapter 2

Background

2.1 Preliminaries

2.1.1 Bilinear Maps

We let $G_1$, $G_2$ and $G_T$ denote three multiplicative cyclic groups of prime order $p$. Also, we let $g_1$ and $g_2$ be generators of $G_1$ and $G_2$, resp., and $e$ be a bilinear map, $e: G_1 \times G_2 \rightarrow G_T$. The bilinear map $e$ has the following properties:

1. Bilinearity: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.

2. Non-degeneracy: $e(g_1, g_2) \neq 1$.

We say that $G_1$ and $G_2$ are bilinear groups if the group operation in $G_1$ and $G_2$ and the bilinear map $e: G_1 \times G_2 \rightarrow G_T$ are both efficiently computable. If $G_1 \neq G_2$, the map $e$ is an asymmetric bilinear map. Otherwise, we can simply denote $G_1$ and $G_2$ as $G$ and call $e: G \times G \rightarrow G_T$ as a symmetric bilinear map.

2.1.2 Access Structures

We adopt the definition of Access Structure and Linear Secret-Sharing Schemes from [Bei96].

**Definition 2.1 (Access Structure)** Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $A \subset 2^{\{P_1, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in A$ and $B \subset C$, then $C \in A$. An monotone access structure is a monotone collection $A$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $A \subset 2^{\{P_1, \ldots, P_n\}} \setminus \emptyset$. The sets in $A$ are called the authorized sets, and the sets not in $A$ are called the unauthorized sets.

**Definition 2.2 (Linear Secret-Sharing Schemes (LSSS))** A secret sharing scheme $\Pi$ over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$. 
2. There exists a matrix $A$ called the share-generating matrix for $\Pi$. The matrix $A$ has $m$ rows and $\ell$ columns. For all $i = 1, \ldots, m$, the $i$th row of $A$ is labelled by a party $\rho(x)$ ($\rho$ is a function from $\{1, \ldots, m\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \ldots, r_\ell)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_\ell \in \mathbb{Z}_p$ are randomly chosen, then $Av$ is the vector of $m$ shares of the secret $s$ according to $\Pi$. The share $(Av)_i$ belongs to party $\rho(x)$. 

LSSS often uses for attribute based encryption. If $\rho$ is injective, attribute based encryption scheme only allows one-use of attributes. It means that an attribute in an access policy can be appear only once. Otherwise, if $\rho$ is not necessary to be injective, attribute based encryption allows multi-uses of attributes.

### 2.1.3 Complexity Assumptions

**Decisional Linear Assumption (DLIN)** Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (p, G, G_T, e) \leftarrow \mathcal{G}, \quad g, f, \nu \leftarrow G, \quad c_1, c_2 \leftarrow \mathbb{Z}_p, \quad D = (G, g, f, \nu, g^{c_1}, f^{c_2}),$$

$$T_0 = \nu^{c_1+c_2}, \quad T_1 \leftarrow G$$

We define the advantage of an algorithm $A$ in breaking DLIN to be:

$$\text{Adv}_{\mathcal{G},A}^{\text{DLIN}}(\lambda) := |\Pr[A(D, T_0) = 1] - \Pr[A(D, T_1) = 1]|$$

**Decisional Bilinear Diffie-Hellman Assumption (DBDH)** Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (p, G, G_T, e) \leftarrow \mathcal{G}, \quad g \leftarrow G, \quad c_1, c_2, c_3 \leftarrow \mathbb{Z}_p, \quad D = (G, g, g^{c_1}, g^{c_2}, g^{c_3}),$$

$$T_0 = e(g, g)^{c_1 c_2 c_3}, \quad T_1 \leftarrow G_T$$

We define the advantage of an algorithm $A$ in breaking DBDH to be:

$$\text{Adv}_{\mathcal{G},A}^{\text{DBDH}}(\lambda) := |\Pr[A(D, T_0) = 1] - \Pr[A(D, T_1) = 1]|$$

It should be noted that DLIN implies DBDH [BW06b, CW14a].

**Symmetric External Diffie-Hellman Assumption (SXDH)** Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (p, G_1, G_2, G_T, e) \leftarrow \mathcal{G}, \quad g_1 \leftarrow G_1, \quad g_2 \leftarrow G_2, \quad c, d, z \leftarrow \mathbb{Z}_p$$

$$D = (g_1, g_1^d, g_1^z \in G_1, g_2, g_2^z \in G_2)$$
\[ T_0 = g_1^{dz}, \quad T_1 \overset{R}{\leftarrow} G_1 \]

We define the advantage of an algorithm \( \mathcal{A} \) in breaking \( SXDH \) to be:

\[
Adv_{\mathcal{G},\mathcal{A}}^{SXDH}(\lambda) := |\Pr[\mathcal{A}(D,T_0) = 1] - \Pr[\mathcal{A}(D,T_1) = 1]| 
\]

We introduce three assumptions \( LW1, LW2 \) and \( LW3 \) which originally appear in [LW09]. It is worth noting that following assumptions utilize asymmetric bilinear maps as the \( SXDH \) assumption although the other standard assumptions \( DLIN \) and \( DBDH \) use symmetric bilinear maps.

**Assumption 1.** (\( LW1 \)) Given a group generator \( \mathcal{G} \), we define the following distribution:

\[
\mathbb{G} = (p,G_1,G_2,G_T,e) \overset{R}{\leftarrow} \mathcal{G}, \quad f_1 \overset{R}{\leftarrow} G_1, \quad f_2 \overset{R}{\leftarrow} G_2, \quad a,c,d \overset{R}{\leftarrow} \mathbb{Z}_p \\
D = (f_1,f_1^a,f_1^{ac},f_1^c,f_1^d,f_1^{ad},f_1^{cd},f_1^{c3d} \in G_1, f_2,f_2^c \in G_2) \\
T_0 = f_1^{ac^d}, \quad T_1 \overset{R}{\leftarrow} G_1
\]

We define the advantage of an algorithm \( \mathcal{A} \) in breaking assumption 1 to be:

\[
Adv_{\mathbb{G},\mathcal{A}}^{LW1}(\lambda) := |\Pr[\mathcal{A}(D,T_0) = 1] - \Pr[\mathcal{A}(D,T_1) = 1]| 
\]

**Assumption 2.** (\( LW2 \)) Given a group generator \( \mathcal{G} \), we define the following distribution:

\[
\mathbb{G} = (p,G_1,G_2,G_T,e) \overset{R}{\leftarrow} \mathcal{G}, \quad f_1 \overset{R}{\leftarrow} G_1, \quad f_2 \overset{R}{\leftarrow} G_2, \quad c,d,t,w \overset{R}{\leftarrow} \mathbb{Z}_p \\
D = (f_1,f_1^d,f_1^{dp},f_1^{tw},f_1^{dwt},f_1^{dpt} \in G_1, f_2,f_2^{c},f_2^{d},f_2^{cw} \in G_2) \\
T_0 = f_2^{cw}, \quad T_1 \overset{R}{\leftarrow} G_2
\]

We define the advantage of an algorithm \( \mathcal{A} \) in breaking assumption 2 to be:

\[
Adv_{\mathbb{G},\mathcal{A}}^{LW2}(\lambda) := |\Pr[\mathcal{A}(D,T_0) = 1] - \Pr[\mathcal{A}(D,T_1) = 1]| 
\]

**Assumption 3.** (\( LW3 \)) Given a group generator \( \mathcal{G} \), we define the following distribution:

\[
\mathbb{G} = (p,G_1,G_2,G_T,e) \overset{R}{\leftarrow} \mathcal{G}, \quad f_1 \overset{R}{\leftarrow} G_1, \quad f_2 \overset{R}{\leftarrow} G_2, \quad a,b,c \overset{R}{\leftarrow} \mathbb{Z}_p \\
D = (f_1,f_1^a,f_1^b,f_1^c \in G_1, f_2,f_2^a,f_2^b,f_2^c \in G_2)
\]
\[ T_0 = e(f_1, f_2)^{abc}, \quad T_1 \overset{R}{\leftarrow} G_T \]

We define the advantage of an algorithm \( A \) in breaking assumption 3 to be:

\[ \text{Adv}_{G,A}^{LW_3}(\lambda) := | \Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1] | \]

### 2.1.4 Functional Encryption

We define functional encryption and its security models.

**Definition 2.3 (Functional Encryption)** [Att14a] A functional encryption for a predicate \( R \) consists of \texttt{Setup}, \texttt{Encrypt}, \texttt{KeyGen} and \texttt{Decrypt} as follows:

\texttt{Setup} \((1, \ell) \rightarrow (PK, MSK)\): takes as input a security parameter \( \lambda \) and an integer \( \ell \) allocated to a predicate. The output is a public parameter \( PK \) and a master secret key \( MSK \).

\texttt{KeyGen} \((x, MSK, PK) \rightarrow SK\): takes as input a predicate \( x \in X \), a master secret key \( MSK \) and a public parameter \( PK \). The output is a private key \( SK \).

\texttt{Encrypt} \((y, M, PK) \rightarrow CT\): takes as input a description \( y \in Y \), a public parameter \( PK \) and a plaintext \( M \). The output is a ciphertext \( CT \).

\texttt{Decrypt} \((SK, CT) \rightarrow M\): takes as input a secret key \( SK \) for \( x \) and a ciphertext \( CT \) for \( y \). If \( R(x, y) = 1 \), the output is either a message \( M \). Otherwise, \( \bot \).

**Correctness.** For all \( M, x \in X, y \in Y \) such that \( R(x, y) = 1 \), if \( SK \) is the output of \texttt{KeyGen}(\( x, MSK, PK \)) and \( CT \) is the output of \texttt{Encrypt}(\( y, M, PK \)) where \( PK \) and \( MSK \) are the outputs of \texttt{Setup}(\( 1, \kappa \)), then \texttt{Decrypt}(\( SK, CT \)) outputs \( M \).

**Definition 2.4 (Adaptive Security of Functional Encryption)** [Att14a] With \( q_t \) private key queries where \( q_t \) is polynomial, a functional encryption for a predicate \( R \) is adaptively secure if there is no PPT adversary \( \mathcal{A} \) which has a non-negligible advantage in the game between \( \mathcal{A} \) and the challenge \( \mathcal{C} \) defined below.

\texttt{Setup}: The challenger runs \texttt{Setup}(\( 1, \kappa \)) to create \((PK, MSK)\). PK is sent to \( \mathcal{A} \).

\texttt{Phase 1}: The adversary requests a private key for \( x_i \in X \) for \( i \in [1, q_t] \). For each \( x_i \), the challenger returns \( SK_i \) created by running \texttt{KeyGen}(\( x_i, MSK, PK \)).

\texttt{Challenge}: When the adversary requests the challenge ciphertext for \( y \in Y \) such that \( R(x_i, y) = 0 \ \forall i \in [1, q_t] \), and submits equal-length messages \( M_0 \) and \( M_1 \), the challenger randomly selects \( b \) from \( \{0, 1\} \) and returns the challenge ciphertext \( CT \) created by running \texttt{Encrypt}(\( y, M_b, PK \)).
Phase 2: This is identical with Phase 1 except the additional restriction that \( x_i \in \mathcal{X} \) for \( i \in [q_1 + 1, q_t] \) such that \( R(x_i, y) = 0; \forall i \in [q_1 + 1, q_t] \).

Guess: The adversary outputs \( b' \in \{0, 1\} \). If \( b = b' \), then the adversary wins.

We define the advantage of the adversary against a functional encryption FE as

\[
Adv_{\mathcal{A}}^{\text{FE}}(\lambda) := |\Pr[b = b'] - 1/2|.
\]

**Definition 2.5** *Semi-adaptively Secure Functional Encryption* [CW14c] With \( q_t \) private key quires where \( q_t \) is polynomial, a functional encryption for a predicate \( R \) is semi-adaptively secure if there is no PPT adversary \( \mathcal{A} \) which has a non-negligible advantage in the game between \( \mathcal{A} \) and the challenge \( \mathcal{C} \) defined below.

**Setup**: The challenger runs \( \text{Setup}(1^\lambda, \kappa) \) to create (PK, MSK). PK is sent to \( \mathcal{A} \).

**Init**: After all public parameters are published, the adversary chooses a target \( y \in \mathcal{Y} \) for the challenge ciphertext, and gives it to the challenger.

**Phase 1**: The adversary requests a private key for \( x_i \in \mathcal{X} \) for \( i \in [1, q_1] \). For each \( x_i \), the challenger returns \( SK_i \) created by running \( \text{KeyGen}(x_i, MSK, PK) \).

**Challenge**: When the adversary requests the challenge ciphertext for \( y \in \mathcal{Y} \) such that \( R(x_i, y) = 0 \forall i \in [1, q_1] \), and submits equal-length messages \( M_0 \) and \( M_1 \), the challenger randomly selects \( b \) from \( \{0, 1\} \) and returns the challenge ciphertext \( CT \) created by running \( \text{Encrypt}(y, M_b, PK) \).

**Phase 2**: This is identical with Phase 1 except the additional restriction that \( x_i \in \mathcal{X} \) for \( i \in [q_1 + 1, q_t] \) such that \( R(x_i, y) = 0; \forall i \in [q_1 + 1, q_t] \).

Guess: The adversary outputs \( b' \in \{0, 1\} \). If \( b = b' \), then the adversary wins.

We define the advantage of the adversary against a functional encryption FE as

\[
Adv_{\mathcal{A}}^{\text{FE}}(\lambda) := |\Pr[b = b'] - 1/2|.
\]

**Definition 2.6** (*Selectively Secure Functional Encryption*) Selectively secure functional encryption is defined identically to semi-adaptively secure functional encryption except *Init*. In selectively secure functional encryption, *Init* is performed before *Setup*. Therefore, the adversary declares its target \( y \in \mathcal{Y} \) before it sees any public parameters.
CHAPTER 2. BACKGROUND

2.2 Related Work

2.2.1 Difficulty in Functional Encryption

Functional encryption rooted to identity based encryption [Sha84]. In an identity based encryption, an arbitrary identity can be used as a public key. Therefore, in an identity based encryption, there is no public key to allocated to each user. Identities such as email addresses or staff numbers replace public keys. Although the concept of identity based encryption was simple and straight-forward, achieving the adaptive security of identity based encryption in standard model was remained a difficult problem for a long time. The first identity based encryption in the standard model was introduced by Boneh and Boyen [BB04b]. Before this work, it requires a random oracle [BF01] or weaker security model [BB04a, CHK03] such as selective security.

The selective security model is considered an alternative of adaptive security model. In the selective security model, the ability of the adversary is limited since it must declare the target ciphertext before it sees any parameters in the system. This makes partitioning technique more suitable since the simulator can separate the target keys before it responds any query for the other keys. In addition, a semi-adaptive security model is also considered [CW14c, Tak14]. In semi-adaptive model, the adversary declares the target after it sees public keys but before it queries any private key.

Although those security models are useful to prove minimum security of the schemes, more desirable security model for functional encryption is adaptive security model. In the adaptive security model the adversary collects many secret keys before and after it sets the target ciphertext which it wants to decrypt and collude those keys to break the ciphertext as an adversary does in the real world. However, responding this adaptive adversary is difficult, in particular if the simulator does not know the target before the adversary declares it, because the traditional techniques require a separation between keys which are secure from the adversary and the other keys which are compromised to the adversary. Particularly, although the partitioning technique is still useful for adaptive security of simple functional encryption such as identity based encryption [BB04b, Wat05, Gen06] and hierarchical identity based encryption [GH09], the proving adaptive security become more significantly difficult if encryption scheme requires a more complicated functions such as identity based broadcast encryption, predicate encryption and attribute based encryption.

We may consider broadcast encryption as an example. Broadcast encryption [FN93] and Identity based broadcast encryption [Del07, SF07] are a public key encryption system where a sender encrypts a message for multiple receivers. Intended receivers share a ciphertext, but they can decrypt it using their own keys. In broad-
Table 2.1: Summary of Dual System Encryption

<table>
<thead>
<tr>
<th>Security Games</th>
<th>Private Keys</th>
<th>Ciphertext</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game_{Real}</td>
<td>Normal</td>
<td>Normal</td>
<td>Equivalent to Adaptive Security Model</td>
</tr>
<tr>
<td>\approx Game_0</td>
<td>Normal</td>
<td>Semi-functional</td>
<td>Semi-functional Cipher Text Invariance</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>Semi-functional Key Invariance</td>
</tr>
<tr>
<td>\approx Game_k</td>
<td>Semi-functional (\leq k)</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td></td>
</tr>
<tr>
<td>\approx Game_q</td>
<td>Semi-functional</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>\approx Game_{Final}</td>
<td>Semi-functional</td>
<td>Semi-functional + Random message</td>
<td>Semi-functional Security</td>
</tr>
</tbody>
</table>

q: The total number of key query

cast encryption systems, a small sized ciphertext is important. Otherwise, if the size of ciphertexts increases linearly with the number of receivers, there is no benefit that a sender simply uses identity based encryption for each receiver for broadcasting.

The first fully collision resistant broadcast encryption with a short ciphertext was introduced by Boneh, Gentry and Waters [BGW05], but it is only selectively secure. Fully collision resistant implies that any users outside the target recipient cannot decrypt the ciphertext even if all users except the users in the target ciphertext collude. The adaptive security was achieved by Boneh and Waters [BW06a]. The size of the ciphertext in their system sub-linear to the total number of users in the system.

For identity based broadcast encryption, the partitioning technique is utilized for selectively secure identity based encryption [Del07]. Later, Gentry and Waters [GW09] suggested the first adaptively secure identity-based broadcast encryption scheme but still utilized partitioning technique. The size of ciphertexts in their scheme is only sub-linear to the number of recipients. For sub-linear sized ciphertext, they first introduced an IBBE scheme which has a linear sized Tag in the ciphertext but allow exponentially many users in the system. Subsequently, they suggested a way to achieve sub-linear sized ciphertext by reusing Tag in the original scheme and increasing the size of other components in a ciphertext from constant to sublinear. However, there is no partitioning technique which leads adaptively secure identity based broadcast scheme with a constant sized ciphertext, to the best of our knowledge.

2.2.2 Dual System Encryption

Waters overcame the limitation of the traditional technique by introducing dual system encryption [Wat09]. In the dual system encryption, proving security of en-
encryption schemes is divided multiple relatively easy proofs. The objectives of these multiple proofs are converting keys and ciphertext in the adaptive security model to auxiliary types which are called semi-functional keys and semi-functional ciphertexts. After it converts all keys and the challenge ciphertext to semi-functional ones, proving security becomes much easier since semi-functional keys cannot decrypt semi-functional ciphertexts. We summarize the concept of the dual system encryption in table 2.1. In dual system encryption, the adaptive security model denoted as $\text{Game}_\text{Real}$. If $\text{Game}_\text{Real}$ is indistinguishable from $\text{Game}_\text{Final}$, the security is proved since the message encrypted is replaced to a random message in the final game. Therefore, it is proved that the challenge ciphertext does not reveal any information about the message encrypted.

The critical proof of the dual system encryption is showing the semi-functional key invariance which is the invariance between $\text{Game}_{k-1}$ and $\text{Game}_k$ in the table. In the semi-functional key invariance, the $k^{th}$ key is converted from a normal key to a semi-functional key which cannot decrypt the (semi-functional) challenge ciphertext. Therefore, it means that the key converted from a valid key to an invalid key for the semi-functional ciphertext. In this proof, both semi-functional parts of the challenge key (i.e. the $k^{th}$ key) and the challenge ciphertext are projected from the normal parts. Therefore, they are strongly correlated each other. Nevertheless, since the semi-functional parts must hinder the decryption, the correlation of the semi-functional parts must be hidden to the adversary. Generally, hiding this correlation is easier than the traditional partitioning proof. In the dual system encryption, the parameters used in semi-functional parts of the challenge key and the challenge ciphertext are isolated well from public keys and the other keys. It enables to reduce the scope of the critical point of the security analysis. In particular, in the semi-functional space, the relation between the challenge key and the challenge ciphertext is considered at a time although the traditional technique requires to simulate all keys and the challenge ciphertext and public key in the analysis.

Therefore, this breakthrough helps to overcome prior difficulty of achieving adaptively secure functional encryption. For example, Attrapadung and Libert [AL10] introduced the first IBBE scheme having a constant sized ciphertext using the dual system encryption. Moreover, the first fully secure expressive ABE scheme for a boolean function was introduced by Lewko, Okamoto, Sahai, Takashima and Waters [LOS+10] using the dual system encryption.

2.2.3 Doubly Selective Security

The dual system encryption is very versatile to construct functional encryption schemes but the relation between the dual system encryption and the traditional
Table 2.2: Summary of Doubly Selective Security

<table>
<thead>
<tr>
<th>Security Games</th>
<th>Private Keys</th>
<th>Ciphertext</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\approx_{\text{Game}_{\text{Real}}}$</td>
<td>Normal</td>
<td>Normal</td>
<td>Equivalent to Adaptive Security Model</td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{0}}$</td>
<td>Normal</td>
<td>Semi-functional</td>
<td>Semi-functional Ciphertext Invariance</td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{k}^{N}}$</td>
<td>Semi-functional ($&lt; k$) Nominally Semi-functional ($= k$) Normal ($&gt; k$)</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{T}}$</td>
<td>Semi-functional ($&lt; k$) Temporary Semi-functional ($= k$) Normal ($&gt; k$)</td>
<td>Semi-functional</td>
<td>Semi-functional Key Invariance (Co-selective Security)</td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{k}}$</td>
<td>Semi-functional ($&lt; k$) Normal ($&gt; q_{1}$)</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>$\approx_{\text{Game}<em>{q</em>{1}}}$</td>
<td>(if $\leq q_{1}$) Semi-functional (if $&gt; q_{1}$) Normal</td>
<td>Semi-functional</td>
<td>Semi-functional Key Invariance (Selective Security)</td>
</tr>
<tr>
<td>$\approx_{\text{Game}<em>{q</em>{1}+1}}$</td>
<td>(if $\leq q_{1} + 1$) Semi-functional (if $&gt; q_{1} + 1$) Normal</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{q}}$</td>
<td>Semi-functional</td>
<td>Semi-functional</td>
<td></td>
</tr>
<tr>
<td>$\approx_{\text{Game}_{\text{Final}}}$</td>
<td>Semi-functional</td>
<td>Semi-functional + Random message</td>
<td>Semi-functional Security</td>
</tr>
</tbody>
</table>

$q_{1}$: The number of key queries before the challenge ciphertext query (Phase I)
$q$: The total number of key queries (Phase I + Phase II)

partitioning technique is not obvious. The dual system encryption also needs some computational assumptions but this is only used to project normal parts of keys to semi-functional parts. The core part of the dual system encryption depends only on information theoretic arguments.

Lewko and Waters [LW12] introduced a new technique which replaced the information theoretic arguments to computational assumptions as do the partitioning techniques. This replacement made the dual system encryption even more powerful. For example, more desirable attribute based encryption schemes which allowed multi-use of attributes and a large universe of attributes were introduced using doubly selective security [Att14a]. Additionally, their technique shows that the achievements from the traditional partitioning proofs can be utilized in the dual system encryption. This is a versatile result since there exist a number of functional encryption schemes of which the selective securities are proved.

We add table 2.2 to summarize the proving sequences of the dual system encryption with Lewko and Waters’ doubly selective security. There are two more types of private keys in the table. In Waters’ dual system encryption (table 2.1), private keys are only either normal or semi-functional. In the doubly selective security, two more types which are nominally semi-functional keys and temporary semi-functional keys
are defined for private keys. A nominally semi-functional key has semi-functional parts, but these parts are correlated with the semi-functional parts of the challenge ciphertext. Hence, they can decrypt the semi-functional ciphertexts. This is possible since the semi-functional parts of keys and ciphertexts are projected from their the normal parts in the dual system encryption. A temporary semi-functional key work similarly to the semi-functional keys in the original dual system encryption. They retain some correlation as a nominally semi-functional keys, but it also contains a random parts in the semi-functional parts. Therefore, they cannot decrypt semi-functional ciphertexts.

The core idea of the doubly selective security is that the invariance between a nominally semi-functional key and a temporary semi-functional key can be proved computationally using two selective security proofs. As a realization of their technique, they provide CP-ABE for multi-use of attributes. In their work, the invariance between a nominally semi-functional key and a temporary semi-functional key was proved using the selective KP-ABE proof if the key that the simulator sought for its type is queried before the challenge ciphertext (i.e. co-selective security). Also, if the key was queried after the challenge ciphertext (i.e. selective security), the invariance for those two type keys was showed using the selective CP-ABE proof. They show that utilizing selectively secure KP-ABE to prove the co-selective security of CP-ABE is possible since the order of information delivered from the adversary to the simulator is identical in those schemes. In detail, in the co-selective proof of CP-ABE, a set of attribute for a private key given to the simulator before an access structure for challenge ciphertext. This order is exactly identical with that of KP-ABE.

2.2.4 Techniques for Prime Order Groups

Many functional encryption schemes were initially introduced in composite order groups. Moreover, compilers of encoding frameworks were also introduced initially in composite order groups. However, composite order groups require a significant efficiency loss to functional encryption schemes when they are implemented. According to Guillevic [Gui13], the minimum group orders for prime order and composite order bilinear group are 256 and 2,644 bits, resp in a 128 bits security level. Moreover, a pairing computation in composite order groups is about 254 times slower than that of prime order bilinear groups. Therefore, constructing an adaptively secure functional encryption schemes in prime order groups is essential to ensure that the schemes are adoptable in practice.

Dual Pairing Vector Spaces (DPVS) [OT09, OT10] are often used to feature composite order groups into prime order groups. In DPVS, orthogonal vectors which
consist of prime order group elements are utilized to realized composite order group elements. More recently, Lewko and Waters suggest an generic technique in [Lew12] to transform a construction in composite order groups into prime order groups by utilizing DPVS. However, DPVS still retains an efficiency loss caused by the size of vectors. For example, in [Lew12], the size of vector increase linearly to the size of predicate in a encryption scheme. This implies a significant loss in efficiency. The disadvantage of DPVS is partially eased by sparse DPVS [OT15, OT11] which utilizes vectors having many zeros as elements. However, it is not clear how sparse DPVS can be utilized for generic conversions for functional encryption.

Several techniques [Fre10, HHH+14, Seo12] which convert encryption systems from composite order groups to prime order groups have also been proposed. Nevertheless, the techniques in [Fre10, HHH+14, Seo12] are not applicable to dual system encryption since they do not hide the values of parameters used in semi-functional spaces information theoretically in prime order groups. It means that those techniques are not applicable to encoding frameworks since they are based on the dual system encryption.

Therefore, in order to construct functional encryption schemes in prime order groups using encoding frameworks, new compilers [CGW15, Att15, AC16] for encodings were introduced. Interestingly, all compilers were introduced using the Dual System Groups [CW13]. The dual system groups were introduced also to features composite order groups in prime order groups. They are observed the properties which make composite order groups be applicable to the dual system encryption. Then, they defined new domains for groups which satisfies those properties and showed that the groups elements from newly defined domains can be achievable using either prime order groups and composite order groups. Since the groups which are newly defined in the dual system groups must provide more desirable properties than ordinary bilinear prime order groups, each group elements in these defined domains consist of multiple prime order group elements. The purpose of the dual system groups is quite similar to that of the encoding frameworks. However, the dual system groups focused on the properties of groups to replace composite order groups to prime order groups. Therefore, it still needs to construct and prove an encryption scheme using properties which are similar to those of composite order groups. However, encoding frameworks took a modular approaches. Encoding frameworks formalize function parts as module (i.e. encodings). Since it has a generic compiler to interpret these modules and the security of compiler is already proved, the frameworks only need a relatively simple requirements for encodings.
2.3 Encoding Frameworks

There are specific properties required for the dual system encryption. These requirements result in those functional encryption schemes share common properties, naturally. For example, functional encryption schemes [LW09, LOS+10] in composite order groups share linear structures. In particular, private keys and ciphertexts are linear over their randomization parameters. This common property is essential to project the semi-functional parts of semi-functional keys from their normal parts in the dual system encryption.

Encoding frameworks [Att14a, Wee14] were introduced by observing this coupling of functional encryption schemes and formalize the common properties. Two encoding frameworks the pair encoding framework [Att14a] and the predicate encoding framework [Wee14] were introduced, concurrently. Those frameworks enable a modular approach for functional encryption schemes.

We draw a conceptual figure (Figure 2.1) which explains encoding frameworks. In the encoding frameworks, the parameters of a functional encryption such as public keys, private keys and ciphertexts are presented as encodings and those encodings must meet properties defined in the framework. Then, they provide a generic compiler of which the security is proved only using the properties of encodings to interpret those encodings to functional encryption schemes. Therefore, if a new encoding is shown that the suggested encoding scheme satisfies the properties that encoding framework requires, the corresponding encryption scheme which is interpreted by the compiler is also automatically secure. This means that one can suggest a new encryption scheme relatively easily by introducing a new encoding scheme. The security and efficiency of the resulting encryption schemes heir exactly those of compilers. Therefore, improving security and efficiency of a compiler takes an effect on the whole resulting encryption schemes without any modification of encodings. Due to this reason, introducing an improved compiler is important in the encoding framework.

The predicate encoding framework and the pair encoding framework are almost identical technically, but only the pair encoding framework includes Lewko and Waters’ doubly selective security technique. Therefore, we only review one of those encodings. Syntax and properties of the pair encoding framework [Att14a] will be described in following subsection.

\begin{figure}[h]
\centering
\begin{tabular}{ccc}
& & \\
Encodings & A compiler & Schemes \\
\( k\bar{E}(x \in X, \cdot), c\bar{E}(y \in Y, \cdot) \) & Compiler\((R, k\bar{E}, c\bar{E})\) & Encryption Schemes \\
\( R : X \times Y \rightarrow \{0,1\} \) & (Denoted by encoding schemes) & (ABE, PE, ...)
\end{tabular}
\caption{Concept of Encoding frameworks}
\end{figure}
2.3.1 Overview of the Pair Encoding Framework

We briefly summarize syntax of pair encodings [Att14a].

For a predicate \( R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \), a pair encoding for \( R(\mathcal{X}, \mathcal{Y}) \) consists of four algorithms \( P(R) := (\text{Param}, \text{Encode1}, \text{Encode2}, \text{Pair}) \).

**Param** \((\kappa) \rightarrow n : \)** An index \( \kappa \) is taken as input and an integer \( n \), where \( n \) is the length of common vector \( \vec{w} \) shared in Encode1 and Encode2, is output. Hence, the domain of \( \vec{w} \) is defined as \( \mathcal{W} := \mathbb{Z}_p^n \). Additionally, it sets up domains \( \mathcal{R} \) and \( \mathcal{D} \) to generate the random values and the master key.

**Encode1** \((\alpha, x, \vec{w}) \rightarrow \vec{k}(\alpha, x, \vec{w}; \vec{r}) : \)** It takes \((\alpha, x, \vec{w}) \in \mathcal{D} \times \mathcal{X} \times \mathcal{W} \) as input, creates randomness \( \vec{r} \in \mathcal{R} \), and outputs the vector \( \vec{k}(\alpha, x, \vec{w}; \vec{r}) \). We let \( k_i \) denote the \( i^{th} \) element of \( \vec{k}(\alpha, x, \vec{w}; \vec{r}) \). \( k_i \) is a linear combination of monomials of \( \alpha, r_j \) and \( w_j r_j \), where \( r_j \) and \( w_j' \) are the \( j^{th} \) coordinate of \( \vec{r} = (r_1, ..., r_{m_r}) \) and the \( j^{th} \) coordinate of \( \vec{w} = (w_1, ..., w_n) \), respectively. That is

\[
k_i(\alpha, x, \vec{w}; \vec{r}) = b_i \alpha + \left( \sum_{j \in [1,n]} b_{i,j',j} w_j r_j \right) + \left( \sum_{j \in [1,m_r]} b_{i,j} r_j \right)
\]

where \( b_i, b_{i,j',j}, b_{i,j} \) for all \( i \in [1,m_1], j' \in [1,n], j \in [1,m_r] \) are coefficients to represent the encoding if we set \( m_1 = |\vec{k}(\alpha, x, \vec{w}; \vec{r})| \).

**Encode2** \((y, \vec{w}) \rightarrow \vec{c}(y, \vec{w}; s, \vec{s}) : \)** It takes \((y, \vec{w}) \in \mathcal{Y} \times \mathcal{W} \) and randomness \( s \in \mathbb{Z}_p \) and \( \vec{s} \in \mathcal{R} \) as input and outputs the vector \( \vec{c}(y, \vec{w}; s, \vec{s}) \). \( c_i \) which denotes the \( i^{th} \) element of \( \vec{c}(y, \vec{w}; s, \vec{s}) \) is defined similarly as a linear combination of monomials of \( s, s_j, w_j s \) and \( w_j' s_j \) where \( s_j \) and \( w_j' \) are the \( j^{th} \) coordinate of \( \vec{s} = (s_1, ..., s_{m_s}) \) and the \( j^{th} \) coordinate of \( \vec{w} = (w_1, ..., w_n) \), respectively. That is

\[
c_i(y, \vec{w}; s, \vec{s}) = a_i s + \left( \sum_{j' \in [1,n]} a_{i,j',j} w_j' s \right) + \left( \sum_{j' \in [1,n]} a_{i,j'} w_j s_j \right) + \left( \sum_{j \in [1,m_s]} a_{i,j} s_j \right)
\]

where \( a_i, a_{i,j'}, a_{i,j',j}, a_{i,j} \) for all \( i \in [1,m_2], j' \in [1,n], j \in [1,m_s] \) are coefficients to represent the encoding if we set \( m_2 = |\vec{c}(y, \vec{w}; s, \vec{s})| \).

**Pair** \((x, y) \rightarrow M_{xy} : \)** It takes as input predicate \( x \) and description \( y \) and outputs reconstruction matrix \( M_{xy} \).
CHAPTER 2. BACKGROUND

Correctness If \( R(x, y) = 1 \), \( \text{Pair} \) outputs an \( m_1 \times m_2 \) reconstruction matrix \( M_{xy} \), where \( m_1 = |\bar{k}(\alpha, x, \bar{w}; \bar{r})| \) and \( m_2 = |\bar{c}(y, \bar{w}; s, \bar{s})| \), such that

\[
\bar{k}(\alpha, x, \bar{w}; \bar{r})M_{xy}\bar{c}(y, \bar{w}; s, \bar{s})^T = \sum_{i \in [1, m_1], j \in [1, m_2]} k_i M_{xy_{i,j}} c_j = \alpha s
\]

where \( k_i \) and \( c_j \) are the \( i^{th} \) coordinate of \( \bar{k}(\alpha, x, \bar{w}; \bar{r}) \) and the \( j^{th} \) coordinate of \( \bar{c}(y, \bar{w}; s, \bar{s}) \), respectively.

In the generic construction of the pair encoding framework provided in [Att14a], the outputs of \( \text{Encode}_1 \) and \( \text{Encode}_2 \) are used for private keys and ciphertexts, respectively. Intuitively, \( \bar{k}(\alpha, x, \bar{w}; \bar{r}) \) and \( \bar{c}(y, \bar{w}; s, \bar{s}) \) are the exponents of private keys and the ciphertexts in a generic construction; the reconstruction matrix implies that there is a proper decryption algorithm. The security of the generic construction of the pair encoding framework was proved only by following properties:

**Property 1. (Linearity)** For all \((\alpha, x, \bar{w}) \in D \times X \times W\),

\[
\bar{k}(\alpha, x, \bar{w}; \bar{r}) + \bar{k}(\alpha', x, \bar{w}; \bar{r}') = \bar{k}(\alpha + \alpha', x, \bar{w}; \bar{r} + \bar{r}')
\]

where \( \bar{r} \) and \( \bar{r}' \) are randomly selected from \( \mathcal{R} \). Also, for all \((y, \bar{w}) \in Y \times W\),

\[
\bar{c}(y, \bar{w}; s, \bar{s}) + \bar{c}(y, \bar{w}; s', \bar{s}') = \bar{c}(y, \bar{w}; s + s', \bar{s} + \bar{s}')
\]

when \( s \) and \( s' \) are randomly selected from \( \mathbb{Z}_p \) and \( \bar{s} \) and \( \bar{s}' \) are randomly selected from \( \mathcal{R} \).

**Property 2. (\( \alpha \) privacy)** For all \((x, y) \in X \times Y\) such that \( R(x, y) = 0 \), there exist two polynomially indistinguishable oracles \( O_{AH}^1 \) and \( O_{AH}^2 \) which are defined as follows

\( O_{AH}^{(1,2)} \): When the oracle is requested to output an initial instance, it randomly create \( g \) from \( G \) and \( \bar{w} \in W \). Then, it outputs \( \{g\} \).

When the oracle receives a type \( k \) query for \( x \in X \), it randomly generates \( \bar{r} \in \mathcal{R} \). If the oracle is \( O_{AH}^1 \), it sets \( \alpha' = 0 \). If the oracle is \( O_{AH}^2 \), it sets \( \alpha' \) as a random value from \( \mathbb{Z}_p \). Then, it returns as a type \( k \) response \( g^{\bar{k}(\alpha', x, \bar{w}; \bar{r})} \).

When a type \( c \) query is received for \( y \in Y \), the oracle randomly generates \( \bar{s} \in \mathcal{R} \) and \( s \in \mathbb{Z}_p \) and outputs as a type \( c \) response \( g^{\bar{c}(y, \bar{w}; s, \bar{s})} \). It should be noted that the oracle only responds if \( R(x, y) = 0 \) whether type \( k \) or type \( c \) queries are requested firstly.

\*\*We take \( \alpha \) privacy from the predicate encoding framework [Woe14], which is a very similar encoding framework to the pair encoding framework. In pair encoding, this property is defined as security. Hence, computational security of [Att14a] is identical to computational \( \alpha \) privacy in our terminology. Also, computationally \( \alpha \) hiding replaces doubly selective master key hiding of [Att14a].\*\*
Property 3. (Parameter vanishing) There exists some element $\vec{0} \in \mathcal{R}$ such that for all $(\alpha, x, \bar{w}) \in \mathcal{D} \times \mathcal{X} \times \mathcal{W}$, $\vec{k}(\alpha, x, w; \vec{0})$ is statistically independent of $\bar{w}$, that is, for all $\bar{w}' \in \mathcal{W}$:

$$\vec{k}(\alpha, x, \bar{w}; \vec{0}) = \vec{k}(\alpha, x, \bar{w}'; \vec{0}).$$

2.3.2 Computationally $\alpha$ hiding

In the pair encoding framework, $\alpha$ privacy is proved computationally using doubly selective security. We call this computational $\alpha$ privacy. Doubly selective security divides computational $\alpha$ privacy into selective $\alpha$ hiding and co-selective $\alpha$ hiding. We define oracles $O_{SAH}^1$ and $O_{SAH}^2$ for selective $\alpha$ hiding and $O_{CAH}^1$ and $O_{CAH}^2$ for co-selective $\alpha$ hiding. Each oracle returns a group generator as initial instance. Then, they respond to a type $k$ query by returning an output of $\text{Encode1}$, and a type $c$ query by returning an output of $\text{Encode2}$ for predicates $x$ and $y$ such that $R(x, y) = 0$.

$O_{SAH}^{[1,2]}$: When the oracle is requested to output initial instances, it randomly generates $\bar{w} \in \mathcal{W}$ and outputs $\{g\}$. It only responds if a type $c$ query is made before a type $k$ query. When a type $c$ query for $y \in \mathcal{Y}$ is received, the oracle randomly generates $\vec{s} \in \mathcal{R}$ and $s \in \mathbb{Z}_p$ and outputs these as a type $c$ response $g^{\vec{c}(y, \vec{w}; s, \vec{s})}$. When it receives a type $k$ query for $x \in \mathcal{X}$ such that $R(x, y) = 0$, it sets $\alpha' = 0$ if the oracle is $O_{SAH}^1$. Otherwise, if the oracle is $O_{SAH}^2$, $\alpha'$ is set as a random value from $\mathbb{Z}_p$. Then, it returns as a type $k$ response $g^{\hat{k}(\alpha', x, \vec{w}; \vec{r})}$.

$O_{CAH}^{[1,2]}$: When the oracle is requested to output initial instances, it randomly generates $\bar{w} \in \mathcal{W}$ and outputs $\{g\}$. It only responds if a type $k$ query is made before a type $c$ query. When it receives a type $k$ query for $x \in \mathcal{X}$, the oracle randomly generates $\vec{r} \in \mathcal{R}$. If the oracle is $O_{CAH}^1$, it sets $\alpha' = 0$. If the oracle is $O_{CAH}^2$, it sets $\alpha'$ as a random value from $\mathbb{Z}_p$, then, it returns a type $k$ response $g^{\hat{r}(\alpha', x, \vec{w}; \vec{r})}$. When a type $c$ query for $y \in \mathcal{Y}$ is received such that $R(x, y) = 0$, the oracle randomly generates $\vec{s} \in \mathcal{R}$ and $s \in \mathbb{Z}_p$ and outputs, as a type $c$ response, $g^{\vec{c}(y, \vec{w}; s, \vec{s})}$.

Selective $\alpha$ hiding If a type $c$ query is requested before a type $k$ query, the adversary $\mathcal{A}$ has an advantage for distinguishing between $O_{SAH}^1$ and $O_{SAH}^2$ as $\text{Adv}^{O_{SAH}}_{\mathcal{A}}(\lambda)$. Formally,

$$\text{Adv}^{O_{SAH}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(O_{SAH}^1) = 1] - \Pr[\mathcal{A}(O_{SAH}^2) = 1]|.$$

Co-selective $\alpha$ hiding If a type $k$ query is requested before a type $c$ query, we define the advantage of the adversary $\mathcal{A}$ to distinguish between $O_{CAH}^1$ and $O_{CAH}^2$ as $\text{Adv}^{O_{CAH}}_{\mathcal{A}}(\lambda)$. Formally,

$$\text{Adv}^{O_{CAH}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(O_{CAH}^1) = 1] - \Pr[\mathcal{A}(O_{CAH}^2) = 1]|.$$

In a doubly selective technique, selective and co-selective $\alpha$ hidings are used to
Table 2.3: Summary of Our Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g^a$</td>
<td>$(g^{a_1}, g^{a_2})$</td>
</tr>
<tr>
<td>$g^a b$ or $g^a b$</td>
<td>$(g^{a_1 b_1}, g^{a_2 b_2})$</td>
</tr>
<tr>
<td>$(g^{a_1, a_2})^r$</td>
<td>$(g^{r a_1, r a_2})$ where $r \in \mathbb{Z}_p$</td>
</tr>
<tr>
<td>$e(g, g^a)$</td>
<td>$e(g, g^{a_1}) e(g, g^{a_2})$ or $e(g, g^{a_1} g^{a_2}) = e(g, g^{a_1}) e(g, g^{a_2})$</td>
</tr>
</tbody>
</table>

prove the invariance of a challenge key of a functional encryption scheme where the challenge key is the key of which the simulator seeks for the type.

2.4 Notations

We briefly summarize the meaning of our notations. We let $\vec{a} = (a_1, a_2) \in \mathbb{Z}_p^2$ and $\vec{b} = (b_1, b_2) \in \mathbb{Z}_p^2$. Then, vector exponentiations of group elements imply vector group elements. For a group element $g$, $g^\vec{a}$ equals to $(g^{a_1}, g^{a_2})$. In addition, multiplication of vectors in exponents implies component-wise product of two vectors. For example, $g^\vec{a} \vec{b}$ implies $(g^{a_1 b_1}, g^{a_2 b_2})$. Similarly, a scalar exponentiation to a vector of group elements means a scalar multiplication to a vector in exponent. For example, $(g^{(a_1, a_2)})^r = (g^{(r a_1, r a_2)})$ where $r \in \mathbb{Z}_p$. Also, a multiplication of vector groups implies an addition of vectors in their exponents (e.g. $g^\vec{a} g^\vec{b} = g^{\vec{a} + \vec{b}}$). It should be noted that this multiplication is possible only if $|\vec{a}| = |\vec{b}|$. When it comes to a pairing operation, a pairing with vectors implies multiple pairing computations, that is, $e(g, g^\vec{a})$ requires two pairing computations $e(g, g^{a_1}) e(g, g^{a_2})$ where $\vec{a} = (a_1, a_2) \in \mathbb{Z}_p^2$, but the same result is achieved only by one pairing since $e(g, g^{a_1} g^{a_2}) = e(g, g^{a_1}) e(g, g^{a_2})$. 
Chapter 3

Refined Duality of Pair Encoding Schemes

In this chapter, we introduce a technique to convert functional encryption of which adaptive security is proved using doubly selective security [LW12]. Our technique refines the previous conversion technique suggested Attrapadung and Yamada [AY15]. Since our technique does not occur any efficiency loss, our technique improves the efficiency of the schemes converted by Attrapadung and Yamada’ technique which requires additional elements for the conversion.

In order to explain our conversion technique, we first introduce a masked form of functional encryption. This form requires a structural assumption, but this assumption is satisfied by all functional encryption using doubly selective security technique [Att14a, LW12]. To formalize our technique, we utilize the pair encoding framework which well describes functional encryptions.

3.1 Attrapaundg and Yamada’s Technique [AY15]

In the generic construction of pair encodings, $\tilde{k}(\alpha, x, \tilde{w}; \tilde{r})$ and $\tilde{c}(y, \tilde{w}; s, \tilde{s})$ are used for the key generation algorithm and the encryption, respectively. In a dual scheme, the structures of $\tilde{k}(\alpha, x, \tilde{w}; \tilde{r})$ and $\tilde{c}(y, \tilde{w}; s, \tilde{s})$ should be barely changed because the major change of those algorithms may cause the change of the selective proofs and assumptions. Nevertheless, because predicates $x$ and $y$ must be switched from the original scheme, those algorithms are also reversely used in dual scheme. In detail, the key generation algorithm of the dual scheme is constructed using $\tilde{c}(y, \tilde{w}; s, \tilde{s})$ and the encryption algorithm uses $\tilde{k}(\alpha, x, \tilde{w}; \tilde{r})$.

The difficulty of the conversion occurs from this contradiction. Since $\tilde{c}(y, \tilde{w}; s, \tilde{s})$ is used for the key generation in a dual scheme, $\alpha$-privacy of $\tilde{k}(\alpha, x, \tilde{w}; \tilde{r})$ must be transferred to $\tilde{c}(y, \tilde{w}; s, \tilde{s})$ without a change of the selective proofs. In [AY15],

26
Attrapadung and Yamada observed that adding new variables as links transplants \( \hat{\alpha} \) privacy from \( \vec{k}(\alpha, x, \vec{w}; r) \) to \( \vec{c}(y, \vec{w}; s, \vec{s}) \). We let denote those links as \( \hat{\phi} \) and \( \hat{s} \) and \( \hat{\alpha} \) as a new master secret. Then, in their technique, a pair encoding for the dual scheme consists of

\[
\vec{k}'(\hat{\alpha} + \hat{\phi}s, \vec{c}(y, \vec{w}; s, \vec{s})) \quad \text{and} \quad \vec{c}' = (\hat{s}, \vec{k}(\hat{\phi}s, x, \vec{w}; r))
\]

where

\[
\vec{k}(\alpha, x, \vec{w}; r) \quad \text{and} \quad \vec{c}(y, \vec{w}; s, \vec{s})
\]

are the outputs of \textit{Encode1} and \textit{Encode2} of a pair encoding for the dual scheme. Then, they show that hiding \( \hat{\alpha} \) is possible by \( \alpha \) hiding property of the original encodings even if \( \alpha \) of the original encodings is hiding computationally using doubly selective security.

### 3.2 A Masked Form

Functional encryption schemes utilizing doubly selective security shares common structures which can be represented as

\[
\vec{k}(\alpha, x, (\phi, y_a, ...); (u, t, ...)) = (\alpha + \phi u + y_a t, u, ...)
\]

and

\[
\vec{c}(y, (\phi, y_a, ...); s, \vec{s}) = (s, \phi s, ...).
\]

Here, \( \alpha \) represents a master secret and \( (\phi, y_a) \) acts as public parameters to protect \( \alpha \) and \( \phi \) does not appear anywhere else in the rest of \( \vec{k} \) and \( \vec{c} \). Also, \( (t, u) \) and \( s \) are randomization parameters. Informally, in the encodings, \( \phi \) is always isolated from the other elements of the encodings since it only exists to hide the value of \( \alpha \) in \( \alpha + \phi u + y_a t \) while computational \( \alpha \) privacy is proved. This common internal structure (i.e. the existence of \( \phi \)) allows us to propose a generic conversion without adopting additional parameters which was unavoidable in [AY15]. Because \( \phi \) is a well isolated parameter even inside \( \vec{k} \), we can use this existing parameter \( \phi \) to a link to transfer \( \alpha \) privacy from \( \vec{k} \) to \( \vec{c} \). Using this fact, we define a \textit{masked form} as follows

**Definition 3.1.** For a pair encoding scheme \( P(R) \), if there exists \( \vec{dk}(\phi u, x, \hat{\vec{h}}; \hat{r}), \vec{dc}(y, \tilde{\vec{h}}; s, \vec{s}), \phi \) and \( u \) such that

\[
g^{\vec{k}(0, x, \vec{h}; r)} = (g^u, g^{\vec{dk}(\phi u, x, \hat{\vec{h}}; \hat{r})}) \quad \text{and} \quad g^{\vec{c}(y, \tilde{\vec{h}}; s, \vec{s})} = (g^{\phi s}, g^{\vec{dc}(y, \tilde{\vec{h}}; s, \vec{s})})
\]

where \( \vec{h} = (\phi, \tilde{\vec{h}}) \) and \( \vec{r} = (u, \tilde{r}) \), we define \( (\vec{dk}, \vec{dc}, \phi, \hat{\vec{h}}, u, \tilde{r}) \) as a masked form of \( P(R) \).

**Theorem 3.1.** Suppose there is a mask form \( (\vec{dk}, \vec{dc}, \phi, \hat{\vec{h}}, u, \tilde{r}) \) of a pair encoding scheme \( P(R) \) for \( x \in \mathcal{X}, \ y \in \mathcal{Y} \) and \( R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \). Then, Dual-P(R) which consists of \( \vec{k}'(\alpha', y, (\phi, \hat{\vec{h}}); \hat{r}') = (g^u, g^{\phi s}, g^{\vec{dc}(y, \tilde{\vec{h}}; s, \vec{s})}) \) and \( \vec{c}'(x, (\phi, \hat{\vec{h}}); u, \vec{s}') = (g^u, g^{\vec{dk}(\phi u, x, \hat{\vec{h}}; \hat{r})}) \) is also a pair encoding scheme with \( \vec{r}' = (s, \vec{s}) \) and \( \vec{s}' = \tilde{r} \).
CHAPTER 3. REFINED DUALITY OF PAIR ENCODING SCHEMES

Proof: Linearity of Dual-P(R) holds trivially since \( \bar{d}k \) and \( \bar{d}c \) are also elements of \( \bar{k} \) and \( \bar{c} \) which have linearity. Also, \( \alpha' + \phi s \) and \( u \) which are the exponents of \( \bar{k'} \) and \( \bar{c'} \) are linear over \( \alpha' \), \( s \) and \( u \). The parameter vanishing of Dual-P(R) also holds since all elements of \( \bar{d}c \) are linear combinations of monomials \( h_i s, s_j \) and \( h_i s_j \) where \( s_j \) and \( h_j \) are elements of \( \bar{s} \) and \( (\phi, \tilde{h}) \), respectively by the definition of a pair encoding. The critical part of the proof is showing computational \( \alpha \) privacy. It will be proved by lemmas 3.1. and 3.2. \( \square \)

To prove that computational \( \alpha \) privacy of Dual-P(R) holds, we will show that there exist oracles \( O^1_{CAH}, O^2_{CAH}, \hat{O}^1_{SAH} \) and \( \hat{O}^2_{SAH} \) of Dual-P(R). This will be proved using P(R) which is the dual scheme of Dual-P(R). Since P(R) is a pair encoding scheme and satisfies computational \( \alpha \) privacy, there exist the oracles simulating computational \( \alpha \) privacy of P(R). We let denote \( \hat{O}^1_{CAH}, \hat{O}^2_{CAH}, \hat{O}^1_{SAH} \) and \( \hat{O}^2_{SAH} \) those oracles of P(R). Then, we will first show that \( O^1_{CAH} \) and \( O^2_{CAH} \) of Dual-P(R) can be simulated using \( \hat{O}^1_{SAH} \) and \( \hat{O}^2_{SAH} \) of P(R) in lemma 3.1. Similarly, \( \hat{O}^1_{CAH} \) and \( \hat{O}^2_{CAH} \) of P(R) will be utilized to simulate \( O^1_{SAH}, O^2_{SAH} \) of Dual-P(R) in lemma 3.2.

Lemma 3.1. Suppose there is PPT adversary \( \mathcal{A} \) who can distinguish \( O^1_{CAH} \) and \( O^2_{CAH} \) of Dual-P(R) with non-negligible advantage \( \epsilon \). Then, there exists PPT algorithm \( \mathcal{B} \) who can distinguish \( \hat{O}^1_{SAH} \) and \( \hat{O}^2_{SAH} \) of P(R) using \( \mathcal{A} \).

Proof: In this simulation, algorithm \( \mathcal{B} \) will simulate \( O^1_{CAH} \) and \( O^2_{CAH} \) of Dual-P(R) to distinguish between \( \hat{O}^1_{SAH} \) and \( \hat{O}^2_{SAH} \) of P(R).

When the adversary asks an initial instance, \( \mathcal{B} \) requests the initial instance to the oracle that it works with. If it receive \( \{g\} \) from the oracle, it sends the output to \( \mathcal{A} \) as an initial response. It should be noted that \( \mathcal{B} \) works either \( \hat{O}^1_{SAH} \) or \( \hat{O}^2_{SAH} \), but it does not know the type of the oracle which it actually works together.

When the adversary asks a type \( k \) response for \( y \in \mathcal{Y} \) to the algorithm. To get \( g^{\bar{c}(y, \tilde{h}; s, \tilde{s})} \), \( \mathcal{B} \) requests the type \( c \) response for \( y \) to the oracle that it works with. It sends the output to \( \mathcal{A} \).

When the adversary requests a type \( c \) response for \( x \in \mathcal{X} \), it requests a type \( k \) response for \( x \) to the oracle it works with to receive \( g^{\bar{c}(\alpha, x, \tilde{k}; \tilde{r})} \). It sends the output to \( \mathcal{A} \).

If it works with \( \hat{O}^1_{SAH}, \alpha = 0 \). It implies that \( \mathcal{B} \) sends

\[
g^{\bar{c}(\alpha, x, \tilde{k}; \tilde{r})} = (g^u, g^{\bar{d}k(\phi u, x, \tilde{h}; \tilde{r})}) = g^{\bar{c}(x, (\phi, \tilde{h}); u, \bar{r})}
\]

to \( \mathcal{A} \) as the type \( c \) response where \( \bar{r} = \tilde{r} \) and \( \tilde{h} = (\phi, \tilde{h}) \).

Also, as type \( k \) response, it sends to \( \mathcal{A} \)

\[
g^{\bar{c}(y, \tilde{h}; s, \tilde{s})} = (g^{\phi s}, g^{\bar{d}c(y, \tilde{h}; s, \bar{s})}) = g^{\bar{c}(0, x, (\phi, \tilde{h}); \bar{r})}
\]
CHAPTER 3. REFINED DUALITY OF PAIR ENCODING SCHEMES

where \( s' = (s, \tilde{s}) \).

If \( B \) works with \( \tilde{O}^2_{SAH} \), then \( \alpha \) is a random value. The algorithm implicitly sets \( \phi' = \phi + \alpha/u \). This is possible because \( \phi' \) was not needed to be given to \( A \), deterministically. In details, it implies that the type \( c \) response to the \( A \) is

\[
g^{\tilde{k}(\alpha, x, \tilde{h}; \tilde{r})} = (g^u, g^{\tilde{k}(\alpha + \phi, u, x, \tilde{h}; \tilde{r})}) = (g^u, g^{\tilde{k}(\phi', u, x, \tilde{h}; \tilde{r})}) = (g^u, g^{\phi'(x, \tilde{h}); u, x'})
\]

where \( s' = \tilde{r} \).

Also, the type \( k \) response implies

\[
g^{\tilde{c}(y, \tilde{h}, s, \tilde{s})} = (g^{\phi s}, g^{\tilde{c}(y, \tilde{h}, s, \tilde{s})}) = (g^{\phi' s}, g^{\tilde{c}(y, \tilde{h}, s, \tilde{s})})
\]

\[
= (g^{\phi' s}, g^{\tilde{c}(y, \tilde{h}, s, \tilde{s})}) = g^{\tilde{c}(\alpha', x, (\phi', \tilde{h}); r')}
\]

where \( r' = (s, \tilde{s}) \). These implicitly set \( \alpha' = -\alpha s/u \). \( \alpha \) commonly appears in \( \phi' \) and \( \alpha' \), but the value of \( \alpha \) is not revealed since \( \phi \) is uniquely allocated to \( \phi' \). Hence, \( \alpha' \) is also uniformly random.

Therefore, if \( B \) works with \( \tilde{O}^1_{SAH} \), it has properly simulated \( O^1_{CAH} \) with a common vector \( (\phi, \tilde{h}) \). Also, if \( B \) works with \( \tilde{O}^2_{SAH} \), it has properly simulated \( O^2_{CAH} \) with a common vector \( \tilde{h}' = (\phi + \alpha/u, \tilde{h}) \). The inconsistency of the common vectors is acceptable since \( B \) responds only once to \( A \) for each type of queries and the values of a common vector are not given explicitly to \( A \).

Lemma 3.2. Suppose there is a PPT adversary \( A \) who distinguish between \( O^1_{SAH} \) and \( O^2_{SAH} \) of Dual-P(\( R \)) with non-negligible advantage \( \epsilon \). Then, there exists a PPT algorithm \( B \) who distinguish between \( \tilde{O}^1_{CAH} \) and \( \tilde{O}^2_{CAH} \) of \( P(\( R \)) \) using \( A \).

Proof: In this simulation, the algorithm \( B \) will simulate \( O^1_{SAH} \) and \( O^2_{SAH} \) of Dual-P(\( R \)) to distinguish between \( \tilde{O}^1_{CAH} \) and \( \tilde{O}^2_{CAH} \) of \( P(\( R \)) \).

When the adversary asks an initial instance, \( B \) requests the initial instance to the oracle that it works with. If it receive \( \{g\} \) from the oracle, it sends the output to \( A \) as an initial response. It should be noted that \( B \) works either \( \tilde{O}^1_{CAH} \) or \( \tilde{O}^2_{CAH} \), but it does not know the type of the oracle which it actually works together.

When the adversary requests a type \( c \) response for \( x \in X \), it requests a type \( k \) response for \( x \) to the oracle it works with to receive \( g^{\tilde{k}(\alpha, x, \tilde{h}; \tilde{r})} \). It sends the output to \( A \).

When the adversary asks a type \( k \) response for \( y \in Y \) to the algorithm. To get \( g^{\tilde{c}(y, \tilde{h}, s, \tilde{s})} \), \( B \) requests the type \( c \) response for \( y \) to the oracle that it works with. It sends the output to \( A \).

The rest of the proof is similar to the proof of lemma 1. If \( B \) works with \( \tilde{O}^1_{CAH} \), it has simulated well \( O^1_{SAH} \) with a common vector \( (\phi, \tilde{h}) \) where \( \tilde{h} = (\phi, \tilde{h}) \).
 CHAPTER 3. Refined Duality of Pair Encoding Schemes

Table 3.1: Comparison between AY’s Conversion and Ours in Composite Order Groups

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Param.</th>
<th>AY [AY15]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphertext Policy Double</td>
<td>Key</td>
<td>$</td>
<td>G_T</td>
</tr>
<tr>
<td>Spatial Encryption [Att14a] CT</td>
<td>$</td>
<td>G_T</td>
<td>+ (3 + m + \sum_{i=1}^{m}d_i)</td>
</tr>
<tr>
<td>CP-ABE</td>
<td>Key</td>
<td>$</td>
<td>G_N</td>
</tr>
<tr>
<td>with short private keys [Att14a] CT</td>
<td>$</td>
<td>G_T</td>
<td>+ (4 + (1 + 3m)</td>
</tr>
<tr>
<td>Dual Functional Encryption for Regular Language [Att14a] CT</td>
<td>$</td>
<td>G_T</td>
<td>+ (6 + 3\ell)</td>
</tr>
<tr>
<td>KP-ABE in a small universe [LW12] CT</td>
<td>$</td>
<td>G_T</td>
<td>+ (4 + \ell)</td>
</tr>
<tr>
<td>Unbounded CP-ABE [Att14a] Key</td>
<td>$(5 + 2\ell)</td>
<td>G_N</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>CT</td>
<td>$</td>
<td>G_T</td>
</tr>
</tbody>
</table>

All works are realized using the generic construction of [Att14a] in composite order groups. $G_T, G_N$ are domains of composite order group elements of a order $N = p_1p_2p_3$ for $e : G_N^2 \rightarrow G_T$ where $p_1, p_2$ and $p_3$ are distinct primes.

$m$ and $\ell$ are the sizes of a predicate and a description, resp.

$T$ is the maximum number of attributes that a ciphertext can include.

$f_j$ and $d_i$ are the number of columns of affine matrices used in CP-DSE.

Also, if $B$ works with $\tilde{O}_{CAH}^2$, it has properly simulated $O_{SAH}^2$ with a common vector $(\phi + \alpha/u, \tilde{h})$. Therefore, if $A$ distinguishes between $O_{SAH}^1$ and $O_{SAH}^2$ with non-negligible advantage $\epsilon$, then, $B$ can use this output to distinguish between $\tilde{O}_{CAH}^1$ and $\tilde{O}_{CAH}^2$ using it.

3.3 Masked Forms of Existing Schemes

In this section, we provide the masked forms of the pair encoding schemes which are introduced in [Att14a, LW12] to assist the construction of their dual schemes. The original construction of CP-ABE with small universe is in [LW12]. The other masked forms are extracted from pair encoding schemes in [Att14a]. Additionally, we compare the efficiency of the dual schemes which are derived using our masked forms with previous dual schemes from Attrapadung and Yamada (AY) [AY15]. We compare those in both composite order groups (table 3.1) and prime order groups (table 3.2). Our schemes exhibit savings of one element in composite order groups and three elements in prime order groups in both private keys and ciphertexts, when the schemes are converted via the compilers in [Att14a] (for composite order constructions) and [Att15] (for prime order constructions). It should be noted that names of the masked forms are based on their original schemes (e.g. Unbounded KP-ABE), but they appear as their dual schemes in table 3.1 and 3.2 (e.g. Unbounded CP-ABE).

Unbounded KP-ABE [Att14a]

- $(\phi, u, s, \tilde{h}) = (\eta, u, s, (h_0, h_1, \phi_1, \phi_2, \phi_3))$
- $\tilde{d}k :$ Let $A$ is an $m \times \ell$ access matrix. $\pi$ is a map from each row of $A$ to an
Table 3.2: Comparison between AY’s Conversion and Ours in Prime Order Groups

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Param.</th>
<th>AY [AY15]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphertext Policy Doubly Spatial Encryption [Att14a]</td>
<td>CT</td>
<td>$(15 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
<td>$(12 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
</tr>
<tr>
<td>CP-ABE with short private keys [Att14a]</td>
<td>CT</td>
<td>$(15 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
<td>$(12 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
</tr>
<tr>
<td>Dual Functional Encryption for Regular Language [Att14a]</td>
<td>CT</td>
<td>$(15 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
<td>$(12 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
</tr>
<tr>
<td>KP-ABE in a small universe [LW12]</td>
<td>CT</td>
<td>$(15 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
<td>$(12 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
</tr>
<tr>
<td>Unbounded CP-ABE [Att14a]</td>
<td>CT</td>
<td>$(15 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
<td>$(12 + 3ℓ + 3\sum_{j=1}^cf_j)G_2$</td>
</tr>
</tbody>
</table>

All works are realised using the generic construction of [Att15] in prime order groups. $G_T, G_1$ and $G_2$ are domains of prime order group elements of a prime order $p$ for $e : G_1 \times G_2 \rightarrow G_T$. $m$ and $\ell$ are the sizes of a predicate and a description, resp. $T$ is the maximum number of attributes that a ciphertext can include. $f_j$ and $d_i$ are the number of columns of affine matrices used in CP-DSE.

attribute $π(x)$, and not necessary to be injective. Then,

$$d\overline{k}(qu, (A, π), (h_0, h_1, φ_1, φ_2, φ_3); \overrightarrow{r}) = (k_0, k_1, \{k_{2,x}, k_{3,x}, k_{4,x}; \forall x \in [1, m]\})$$

where

$$k_0 = \alpha + rφ_1, \quad k_1 = r, \quad k_{2,x} = A_0 + r\overrightarrow{v}, \quad k_{3,x} = r_x, \quad k_{4,x} = r_x(h_0 + h_1π(x)); \forall x \in [1, m].$$

This implies that $\overrightarrow{r} = (r, r, ..., r_m, v_2, ..., v_k)$ and $\overrightarrow{v} = (rφ_2, v_2, ..., v_k)$.

- $d\overline{c}$: Let $S = \{Att_1, ..., Att_q\}$. Then,

$$d\overline{c}(S, (h_0, h_1, φ_1, φ_2, φ_3); s, \overrightarrow{s}) = (c_0, c_1, c_2, \{c_{3,y}, c_{4,y}; \forall i \in [1, q]\})$$

where

$$c_0 = s, \quad c_1 = sφ_1 + wφ_2, \quad c_2 = w, \quad c_{3,y} = wφ_3 + s_y(h_0 + h_1y), \quad c_{4,y} = s_y; \forall i \in [1, q].$$

This implies that $\overrightarrow{s} = (w, s_1, ..., s_m)$.

**CP-ABE with small universe [LW12]**

- $(φ, u, s, \overrightarrow{h}) = (y_k, u, s, (y_a, a_1, ..., a_n))$

- $d\overline{k}$: Let $S = \{Att_1, ..., Att_q\}$. Then,

$$d\overline{k}(y_ku, S, (y_a, a_1, ..., a_n); \overrightarrow{r}) = (k_0, k_1, \{k_{2,Att_i}; \forall i \in [1, q]\})$$

where

$$k_0 = \alpha + y_at, \quad k_1 = t, \quad k_{2,Att_i} = a_{Att_i}t; \forall i \in [1, q].$$
CHAPTER 3. REFINED DUALITY OF PAIR ENCODING SCHEMES

This implies $\tilde{r} = t$.  

- $\vec{d}c$ : Let $A$ is an $m \times \ell$ access matrix. $\rho$ is a map from $A_x$, the $x^{th}$ row of $A$ to an attribute $\rho(x)$ for all $x \in [1, m]$, and not necessary to be injective. Then,

$$\vec{d}c((A, \rho), (y_n, a_1, ..., a_n); s, \vec{s}) = (c_0, \{c_1, c_2, \ldots \}; \forall x \in [1, m])$$

where

$$c_0 = s, \{c_1, y_nA_x\mu - a_{\rho(x)}s_x, c_2, x \in [1, m] \}.$$  

Hence, $\vec{s} = (\mu_2, \ldots, \mu_\ell, s_1, \ldots, s_m)$ and $\mu = (s, \mu_2, \ldots, \mu_\ell)$.  

KP-DSE [Att14a]

- $(\phi, u, s, \tilde{h}) = (\eta, u, s, (\phi_1, \phi_2, \phi_3, h'))$

- $\vec{d}k$ : For $(A; X^{(1)}, \ldots, X^{(m)})$ where $A \in \mathbb{Z}_p^{n \times k}$ and $X^{(i)} \in \text{AffM}(\mathbb{Z}_p^{n \times d_i})$. Then,

$$\vec{d}k(\eta u, (A; X^{(1)}, \ldots, X^{(m)}), (\phi_1, \phi_2, \phi_3, h'); \vec{r}) = (k_1, k_2, \{k_3, k_4, k_5, k_6; \forall i \in [1, m] \})$$

where

$$k_1 = \alpha + r\phi_1, k_2 = r, k_3 = A_u\vec{r} + r_i\phi_3, \{k_4, r_i, k_5, r_i(h'X^{(i)}); \forall i \in [1, m] \}.$$  

Hence, $\vec{r} = (r, r_1, \ldots, r_m, v_2, \ldots, v_k)$ and $\vec{r} = (r\phi_2, v_2, \ldots, v_k)$.

- $\vec{d}c$ : Let $\Omega = \{Y^{(1)}, \ldots, Y^{(t)}\}$ where $Y^{(j)} \in \text{AffM}(\mathbb{Z}_p^{n \times d_j})$. Then,

$$\vec{d}c(\Omega, (\phi_1, \phi_2, \phi_3, h'); s, \vec{s}) = (c_1, c_2, c_3, \{c_{4, j}, c_{5, j}; \forall j \in [1, \ell] \})$$

where

$$c_1 = s, c_2 = s\phi_1 + w\phi_2, c_3 = w, \{c_{4, j} = (w\phi_3, \rho) + s_j(h'Y^{(j)}), c_{5, j} = s_j; \forall i \in [1, q] \}$$

where $\vec{s} = (w, \{s_j; j \in [1, \ell] \})$.

KP-ABE with short private keys [Att14a]

- $(\phi, u, s, \tilde{h}) = (\eta, u, s, (\phi_1, \phi_2, \phi_3, h_0, \ldots, h_{T+1}))$ where $T$ is the maximum number of attributes which can be included a ciphertext.

- $\vec{d}k$ : Let $A$ is an $m \times \ell$ LSSS access matrix. $\pi$ is a map from each row of $A$ to an attribute $\pi(x)$, and not necessary to be injective. Then,

$$\vec{d}k(\eta u, (A, \pi), (\phi_1, \phi_2, \phi_3, h_0, \ldots, h_{T+1}); \vec{r}) = (k_1, k_2, \{k_3, k_4, k_5, k_6; \forall i \in [1, m] \})$$
CHAPTER 3. REFINED DUALITY OF PAIR ENCODING SCHEMES

where
\[ k_1 = \alpha + r_1, \quad k_2 = r, \quad \{k_{3,i} = A_i \vec{v}^T + r_i \phi_3, \]
\[ k_{4,i} = r_i, \quad k_{5,i} = r_i(h_0, h_2 - h_1 \pi(x), \ldots, h_{T+1} - h_1 \pi(i)^T); \forall i \in [1, m] \}

Hence, \( \vec{r} = (r, r_1, \ldots, r_m, v_2, \ldots, v_\ell) \) and \( \vec{v} = (r_\phi, v_2, \ldots, v_\ell). \)

• \( \vec{d}c \) : Let \( S = \{\text{Att}_1, \ldots, \text{Att}_q\} \). Then,

\[ \vec{d}c(S, (\phi_1, \phi_2, \phi_3, h_0, \ldots, h_{T+1}); s, \vec{s}) = (c_1, c_2, c_3, c_4, c_5) \]

where
\[ c_1 = s, \quad c_2 = s\phi_1 + w\phi_2, \quad c_3 = w, \quad c_4 = w\phi_3 + \tilde{s}(h_0 + h_1a_0 + \ldots + h_{T+1}a_T), \quad c_5 = \tilde{s} \]

where \( a_i \) is an coefficient of \( z^i \) in \( p(x) = \prod_{y \in S}(z - y) \). Also, this implies \( \vec{s} = (w, \tilde{s}) \).

Functional Encryption for Regular Languages [Att14a]

• \((\phi, u, s, \tilde{h}) = (\phi, u, s, (h_0, h_1, h_2, h_3, h_4, \phi_1, \phi_2))\)

• \( \vec{d}k \) : For the description of any DFA \( M = (Q, \Sigma_p, T, q_0, q_{n-1}) \) where \( n = |Q| \), let \( m = |T| \), and parse \( T = \{(q_{x_i}, q_{y_i}, \sigma_i)|t \in [1, m]\} \). Then,

\[ \vec{d}k(\phi u, M, (h_0, h_1, h_2, h_3, h_4, \phi_1, \phi_2); \vec{r}) = (k_1, k_2, k_3, k_4, \{k_{5,t}, k_{6,t}, k_{7,t}; \forall t \in [1, m]\}) \]

where
\[ k_1 = \alpha + r_1, \quad k_2 = r, \quad k_3 = r_0, \quad k_4 = -u_0 + r_0h_0, \]
\[ \{k_{5,t} = r_t, k_{6,t} = u_{x_t} + r_t(h_1 + h_2\sigma_t), k_{7,t} = -u_{y_t} + r_t(h_3 + h_4\sigma_t); \forall t \in [1, m]\}. \]

This implies \( u_{n-1} := \phi_2r \) and \( \vec{r} = (r, r_0, \ldots, r_m, \{u_{q_x}\}_{q_x \epsilon Q\backslash (q_{n-1})}). \)

• \( \vec{d}c \) : For \( w \in (\Sigma_p)^\ast \), let \( \ell = |w| \), and parse \( w = (w_1, \ldots, w_\ell) \).

\[ \vec{d}c(w, (h_0, h_1, h_2, h_3, h_4, \phi_1, \phi_2); s, \vec{s}) = (c_1, c_2, c_3, \{c_4,i, c_5,i; \forall i \in [1, \ell]\}) \]

where
\[ c_1 = s, \quad c_2 = -s\phi_1 + s_\ell\phi_2, \quad c_3 = s_0h_0, \]
\[ \{c_4,i = s_i, c_5,i = s_{i-1}(h_1 + h_2w_i) + s_i(h_3 + h_4w_i); \forall i \in [1, \ell]\}. \]

This implies \( \vec{s} = (s_0, \ldots, s_\ell) \).
Chapter 4

Functional Encryption in Prime order groups

In this chapter, we introduce a new compiler for the pair encoding. The new compiler implies a new generic construction of the pair encoding framework. Our new compiler is in prime order groups and adaptively secure under static assumptions. Some structure of our new compiler resembles Lewko and Waters’ IBE [LW10], but it is expanded for complicated functions using nested dual system encryption.

**New Schemes in Prime Order Groups.** Our compiler shares same encodings with those of [Att14a, AY15] but the outputs different schemes in prime order groups (figure 4.1). Therefore, with our construction (i.e. compiler), we realize FE schemes of [LW12, Att14a, AY15] which were introduced in composite order groups as the schemes in prime order groups. We provide the list of FE schemes achievable via our construction in table 4.1. Our construction achieves adaptive security of following functional encryption schemes in prime order groups for the first time\(^a\). They include Doubly Spatial Encryption, FE for regular language, KP-ABE with short ciphertexts, CP-ABE with short keys and unbounded ABE supporting multi-use of attribute.

**Refining the Notation of Pair Encoding.** The existing properties of pair encodings in [Att14a] are insufficient to construct adaptively secure FE in prime order groups. Therefore, we additionally define linearity over common parameters as a new property. To support this property, we refine the notation of the pair encoding framework. In our notation, common parameters are denoted as \((1, \vec{h})\) where \(\vec{h}\) is a vector of common parameters of the previous notation of pair encodings. It should be noted that this additional property does not harm the generality of the pair encoding framework because by definition, the pair encoding framework already

\(^a\)We notice that there is a concurrent work for this claim at present. We shall describe it in Section 1.2.
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Pair Encodings</th>
<th>Compilers</th>
<th>Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (x, y) \in \mathcal{X} \times \mathcal{Y} )</td>
<td>( k\hat{E}(x, \cdot) ), ( c\hat{E}(y, \cdot) )</td>
<td>Our Compiler</td>
<td>Unbounded ABE, FE, ... in prime order groups</td>
</tr>
<tr>
<td>( R : \mathcal{X} \times \mathcal{Y} \to {0, 1} )</td>
<td></td>
<td>Compiler [Att14a]</td>
<td>Unbounded ABE, FE, ... in composite order groups</td>
</tr>
</tbody>
</table>

Figure 4.1: Results of Our Compiler

requires a linear structure.

**Nested Dual System Encryption.** In Waters’ dual system encryption [Wat09], ciphertexts and private keys have two types which are normal and semi-functional (SF). Using these types, proving adaptive security of FE is divided into several relatively easy problems such as proving the invariance between a normal key and a semi-functional key. Later, Lewko and Waters [LW12] and Attrapadung [Att14a] separated this key invariance problem more specifically to apply the computational hiding. They additionally define a Nominally Semi-functional (NSF) key and a Temporary Semi-functional key (TSF) key and change a key from normal to SF in sequence of

\[
\text{Normal key} \xrightarrow{\text{(Step 1)}} \text{NSF key} \xrightarrow{\text{(computational assump.)}} \text{TSF key} \xrightarrow{\text{(Step 2)}} \text{SF key.}
\]

Then, they reduce the invariance between NSF and TSF to computational assumptions such as \(q\)-type assumptions. To apply the computational assumptions, the semi-functional elements of NSF and TSF keys must be uncorrelated with their normal elements as well as the other private keys. This independence is achieved by the Chinese Remainder Theorem in a composite order group. Due to different primes of composite order, the values in one subgroup do not correlate to those in another subgroup. Therefore, in steps 1 and 2 in the above transition, multiple random values are projected to different subgroup without correlation by a single transition.

We do not use either DPVS or dual system groups which are often used to feature composite order groups to prime order groups for our construction. Instead, we divide the transitions of a key (steps 1 and 2) into more steps by each random value to construct a generic construction for pair encodings in a prime order groups. We additionally define an NE
\[j\] key and a TE
\[j\] key to prove the key invariance. Those keys are special cases of NSF keys and TSF keys. An NE
\[j\] key is identical to a NSF key having \((r_1, \ldots, r_j, 0, \ldots, 0)\) as a random value. That is, the first \(j\) random values are selected randomly, but other random values are being set to 0. A TE
\[j\] key is defined similarly to an NE
\[j\] key, but using a TSF key. Using these types, we additionally localize each random value within the key and project it as semi-
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

Table 4.1: Comparison between Previous Works and Ours

<table>
<thead>
<tr>
<th>FEs</th>
<th>Policy</th>
<th>Multi-use</th>
<th>Universe</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doubly Spatial Encryption</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A [Att14a]</td>
<td>✔</td>
<td>✔</td>
<td>N/A</td>
<td>N/A Composite</td>
</tr>
<tr>
<td>AY [AY15]</td>
<td>✔ ✔</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A Prime</td>
</tr>
<tr>
<td>FE for regular languages</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A [Att14a]</td>
<td>✔ ✔</td>
<td>N/A</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>AY [AY15]</td>
<td>✔ ✔</td>
<td>N/A</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>N/A</td>
<td>Large</td>
<td>Prime</td>
</tr>
<tr>
<td>Unbounded ABE</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OT [OT12]</td>
<td>✔ ✔</td>
<td>No</td>
<td>Large</td>
<td>Prime</td>
</tr>
<tr>
<td>A [Att14a]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>AY [AY15]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Prime</td>
</tr>
<tr>
<td>ABE in a small universe</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOSTW [LOS+10]</td>
<td>✔ ✔</td>
<td>No</td>
<td>Small</td>
<td>Composite</td>
</tr>
<tr>
<td>A [Att14a]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Small</td>
<td>Composite</td>
</tr>
<tr>
<td>LW [LW12]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Small</td>
<td>Prime</td>
</tr>
<tr>
<td>AY [AY15]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Small</td>
<td>Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Small</td>
<td>Prime</td>
</tr>
<tr>
<td>ABE with short ciphertexts</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A [Att14a]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Prime</td>
</tr>
<tr>
<td>ABE with short keys</td>
<td></td>
<td>CT Key Dual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AY [AY15]</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Composite</td>
</tr>
<tr>
<td>Ours</td>
<td>✔ ✔</td>
<td>Yes</td>
<td>Large</td>
<td>Prime</td>
</tr>
</tbody>
</table>

Unbounded ABE implies ABE schemes supporting a large universe of attributes and unbounded sized policy. No in the column of Multi-use implies that each predicate (e.g., an attribute) can appear only once in a policy.

functional elements. For example, in step 1, we prove the invariance of $\text{NE}_{k-1}$ key and $\text{NE}_k$ to show the invariance of a normal key to a NSF key. Also, we change the TSF key to SF key using $\text{TE}_j$. If we denote $m_r$ as the number of random values in a key, a normal key and a TSF key change to a NSF key and an SF key, resp.

Step 1 : Normal key $\rightarrow \text{NE}_1 \rightarrow \ldots \rightarrow \text{NE}_{m_r-1} \rightarrow$ NSF key

Step 2 : TSF key $\rightarrow \text{TE}_{m_r-1} \rightarrow \ldots \rightarrow \text{TE}_1 \rightarrow$ SF key.

Lewko and Waters’ IBE. Since we use the nested dual system encryption, the key invariance problem can be broken down into simpler problems by each random value. To solve those simplified problems, we utilize the assumptions in Lewko and Waters’ IBE [LW10]. Primarily, in their IBE, the technique to provide the key invariance between a normal key and a semi-functional key is only applicable if FEs have one random value to randomize their function parts. This works for IBE well because IBE is one of the simplest form of FE. However, the way of applying their technique to FEs having more complicated structures (i.e., multiple random values for function parts and/or computational hiding) is unknown. In our work, since we successfully simplify the key invariance of pair encodings as we previously described, we can apply their technique to pair encoding framework. Informally, to prove the
invariance between NE key and NE+1 key in our nested model, we consider NE key and NE+1 key as a normal key and an SF key of Lewko and Waters’ IBE scheme. Except the i-th random value, other random values are generated randomly or fixed as 0, we fit our problem to the semi-functional key invariance of their IBE.

4.1 Our Construction (Compiler)

For a pair encoding $P(R, p)$, a functional encryption $FE(P)$ comprises four randomized algorithms employing prime order groups $G_1, G_2$ and $G_T$ of order $p$. In the construction, we use the subscripts of group elements to denote the group generators to be used to generate those elements (e.g. $g_1, f_1 \in G_1$ and $g_2, f_2 \in G_2$).

- **Setup($\lambda$) → $PK, MSK$**: The setup algorithm selects a bilinear group $G_1, G_2, G_T$ of order $p$. The algorithm randomly selects $g_1 \in G_1$ and $g_2 \in G_2$. Then, it runs $\text{Param}(\kappa)$ to derive $n$ where $\kappa$ is the index allocated to the function $R$. It randomly generates $\alpha, \alpha, b, y_u, y_v, h_1, ..., h_n \in \mathbb{Z}_p$ and sets $\tau = y_v + a \cdot y_u$ and $\vec{h} = (h_1, ..., h_n)$. It publishes public parameters as

$$\{g_1, g_1^\vec{h}, g_1^a, g_1^\alpha, g_1^\tau, g_1^\vec{h}, e(g_1, g_2)^\alpha\}.$$  

It also sets MSK as $\{g_2, g_2^a, g_2^\vec{h}, v_2 = g_2^{b\cdot y_v}, u_2 = g_2^{b\cdot y_u}, f_2 = g_2^b\}$.

- **Encrypt($M, y, PK$) → $CT_y$**: The encryption algorithm randomly chooses $(s, \vec{s}) \in \mathbb{Z}_p \times \mathcal{R}_s$ and runs $\text{Enc2}$ to get $\vec{c}(y, (1, \vec{h})); s, \vec{s})^b$. It sets the ciphertext as

$$C = M \cdot e(g_1, g_2)^{\alpha s}, \vec{C}_0 = g_1^{\vec{c}(y, (1, \vec{h}); s, \vec{s})}, \vec{C}_1 = g_1^{a \cdot \vec{c}(y, (1, \vec{h}); s, \vec{s})}, \vec{C}_2 = g_1^{\tau \cdot \vec{c}(y, (1, \vec{h}); s, \vec{s})}.$$  

- **KeyGen($x, MSK, PK$) → $SK_x$**: The key generation algorithm chooses a random vector $\vec{r} \in \mathcal{R}_r$ and runs $\text{Enc1}$ to get $\vec{k}(\alpha, x, (1, \vec{h}); \vec{r})^c$. Then, it randomly selects $\vec{z} \in \mathbb{Z}_p^{km}$ where $k_m$ is $|\vec{k}|$. Finally, it generates the private key following

$$\vec{K}_0 = g_2^{\vec{k}(\alpha, x, (1, \vec{h}); \vec{r})} v_2^{\vec{z}}, \quad \vec{K}_1 = u_2^{\vec{z}}, \quad \vec{K}_2 = f_2^{\vec{z}}.$$  

- **Decrypt($PK, x, y, SK_x, CT_y$) → $M$**: If $R(x, y) = 1$, the algorithm computes a reconstruction matrix $M_{xy}$ such that $\vec{k}(\alpha, x, (1, \vec{h}); \vec{r}) M_{xy} \vec{c}(y, (1, \vec{h}); s) = \alpha s$ by

\[\text{The algorithm only knows the values of } x \text{ and } \vec{r}. \text{ Therefore, } \vec{c}(y, (1, \vec{h}); s, \vec{s}) \text{ is a multivariate linear function of } \alpha \text{ and } \vec{h}. \text{ However, due to the linearity, all elements in a ciphertext can be calculated because } g_1^\vec{h}, g_1^\alpha \text{ and } g_1^{\vec{h}} \text{ are given.}\]

\[\text{Similar to Encrypt, } \vec{K}_0 \text{ can be calculated using } g_1^\vec{h}, g_1^\alpha \text{ and } g_1^{\vec{h}}.\]
Proof. Theorem 4.1 is proved by Lemmas 4.1 to 4.3.

\[ e(g_1, g_2)^{\alpha s} = e(C_0, (K_0)^{M_{xy}}) e(C_1, (K_1)^{M_{xy}}) e(C_2, (K_2)^{M_{xy}}). \]

Finally, the message can be recovered as \( C/e(g_1, g_2)^{\alpha s} \).

Correctness We let \( A_1 := e(C_0, (K_0)^{M_{xy}}), A_2 := e(C_1, (K_1)^{M_{xy}}) \) and \( A_3 := e(C_2, (K_2)^{M_{xy}}) \).

Then, we can calculate

\[
\begin{align*}
A_1 &= e(C_0, (K_0)^{M_{xy}}) = e(g_1^{\bar{c}_{y,(1,\tilde{h});s,s}}, (g_2^{\bar{c}_{(\alpha x,(1,\tilde{h});f),v_2^s})^{M_{xy}}) \\
&= e(g_1^{\bar{c}_{y,(1,\tilde{h});s,s}}, (g_2^{\bar{c}_{(\alpha x,(1,\tilde{h});f),v_2^s})^{M_{xy}}) \\
&= e(g_1, g_2^{\bar{c}_{y,(1,\tilde{h});s,s}}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) \\
&= e(g_1, g_2^{\alpha s}) e(g_1, g_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) \\
A_2 &= e(C_1, (K_1)^{M_{xy}}) = e(g_1^{\bar{c}_{y,(1,\tilde{h});s,s}}, (v_2^s)^{M_{xy}}) \\
&= e(g_1, g_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) \\
A_3 &= e(C_2, (K_2)^{M_{xy}}) = e(g_1^{\bar{c}_{y,(1,\tilde{h});s,s}}, f_2^z) \\
&= e(g_1, g_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s}) e(g_1, v_2^{\alpha s})
\end{align*}
\]

Therefore, the correctness of decryption algorithm holds since

\[ e(g_1, g_2)^{\alpha s} = A_1 \cdot A_2 \cdot A_3. \]

### 4.2 Security Analysis

We prove adaptive security of our construction in the following theorem.

**Theorem 4.1.** (Informal) For all pair encodings \( P(R, p) \) where \( R \) is a function and \( p \) is a prime. our construction \( FE(P) \) is adaptive secure.

**Proof:** Theorem 4.1 is proved by Lemmas 4.1 to 4.3. \( \square \)

To prove adaptive security of our construction, we define several types of keys and ciphertexts, which are only used in security analysis. In following definitions, \((K_0', K_1', K_2')\) and \((C', \tilde{C}_0', \tilde{C}_1', \tilde{C}_2')\) denote a normal key for \( x \in \mathcal{X} \) and a ciphertext for \( y \in \mathcal{Y} \) which are generated using \texttt{KeyGen} and \texttt{Encrypt}.

**Semi-functional (SF) Ciphertext** The algorithm randomly selects \((\tilde{h}', s', s') \in \mathcal{H} \times \mathbb{Z}_p \times \mathcal{R}_s\). It sets \( f_1 = g_1^{\tilde{h}'} \). For a predicate \( y \in \mathcal{Y} \), it outputs a semi-functional
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

CT as

\[ C = C, \quad \vec{C}_0 = \vec{C}_0', \quad \vec{C}_1 = \vec{C}_1' \cdot f_1^{(y, (1, \vec{h}'), s', \vec{x}')}, \quad \vec{C}_2 = \vec{C}_2' \cdot u_1^{(y, (1, \vec{h}'); s', \vec{x}')} \]

Semi-functional (SF) Key The algorithm randomly selects \( \alpha \in \mathbb{Z}_p \). For a predicate \( x \in \mathcal{X} \), it outputs a semi-functional key as

\[ \vec{K}_0 = \vec{K}_0' \cdot f_2^{-a\vec{k}(\alpha', x, (0, \vec{0})); \vec{0}}, \quad \vec{K}_1 = \vec{K}_1' \cdot f_2^{\vec{k}(\alpha', x, (0, \vec{0})); \vec{0}}, \quad \vec{K}_2 = \vec{K}_2' \]

\( \text{E}_j \) Ciphertext The algorithm creates a normal ciphertext \( \{C, \vec{C}_0', \vec{C}_1', \vec{C}_2'\} \) and selects \( s' \in \mathbb{Z}_p \). If \( j = 0 \), the algorithm set \( s'_0 = \vec{0} \). Otherwise, the algorithm randomly selects \( s'_1, \ldots, s'_j \) from \( \mathbb{Z}_p \) and it sets \( s'_j = (s'_1, \ldots, s'_j, 0, ..0) \) and \( f_1 = g_1^b \). It outputs the \( \text{E}_j \) ciphertext as

\[ C = C, \quad \vec{C}_0 = \vec{C}_0', \quad \vec{C}_1 = \vec{C}_1' \cdot f_1^{(y, (1, \vec{h}'); s', \vec{x}')}, \quad \vec{C}_2 = \vec{C}_2' \cdot u_1^{(y, (1, \vec{h}'); s', \vec{x}')} \]

It should be noted that \( \text{E}_{m_s} \) ciphertext is identical to the semi-functional ciphertext by definitions if we write \( \mathcal{R}_s = \mathbb{Z}_p^{m_s} \).

\( \text{NE}_j \) key The algorithm generates a normal key \( \{\vec{K}_0, \vec{K}_1, \vec{K}_2\} \) using \text{KeyGen}. Then it randomly selects \( r'_1, \ldots, r'_j \) from \( \mathbb{Z}_p \) and sets \( r'_j = (r'_1, \ldots, r'_j, 0, ..0) \in \mathcal{R}_r \). Finally, it randomly generates \( \vec{h}' \) and outputs an \( \text{NE}_j \) key as

\[ \vec{K}_0 = \vec{K}_0' \cdot f_2^{-a\vec{k}(0, x, (1, \vec{h}'); r'_j)}, \quad \vec{K}_1 = \vec{K}_1' \cdot f_2^{\vec{k}(0, x, (1, \vec{h}'); r'_j)}, \quad \vec{K}_2 = \vec{K}_2' \]

An \( \text{NE}_0 \) key is identical to a normal key since \( \vec{k}(0, x, (1, \vec{h}'); (0, ..0)) = \vec{0} \) by the definition of pair encodings. Additionally, we call \( \text{NE}_{m_r} \) key as a Nominally Semi-functional (NSF) Key if we write \( \mathcal{R}_r = \mathbb{Z}_p^{m_r} \).

\( \text{TE}_j \) Key The algorithm generates a normal key \( \{\vec{K}_0, \vec{K}_1, \vec{K}_2\} \) using \text{KeyGen}. Then, it randomly selects \( \alpha \in \mathcal{D}, \ r'_1, \ldots, r'_j \in \mathbb{Z}_p \) and \( \vec{h}' \in \mathcal{H} \). Finally, it sets \( r'_j = (r'_1, \ldots, r'_j, 0, ..0) \in \mathcal{R}_r \) and outputs a \( \text{TE}_j \) key as

\[ \vec{K}_0 = \vec{K}_0' \cdot f_2^{-a\vec{k}(\alpha', x, (1, \vec{h}'); r'_j)}, \quad \vec{K}_1 = \vec{K}_1' \cdot f_2^{\vec{k}(\alpha', x, (1, \vec{h}'); r'_j)}, \quad \vec{K}_2 = \vec{K}_2' \]

A \( \text{TE}_0 \) key is identical to an SF key since \( \vec{k}(\alpha', x, (1, \vec{h}'); \vec{0}) = \vec{k}(\alpha', x, (0, \vec{0}); \vec{0}) \) by parameter vanishing. Additionally, we call \( \text{TE}_{m_s} \) key as a Temporary Semi-functional (TSF) Key where \( \vec{0} = (0, \ldots, 0) \).

All \( \text{NE}_j \) key including an NSF key can decrypt both normal and semi-functional ciphertexts by sharing \( \vec{h}' \) in semi-functional elements with semi-functional ciphertexts. However, \( \text{TE}_j \) keys only decrypt normal ciphertexts. If they decrypt semi-
functional ciphertexts, $e(f_1, f_2)^{\alpha s}$ prevents the decryption even if they share $\vec{h}'$ with semi-functional ciphertexts.

We define security games consisting of different types of keys and ciphertexts. We will prove that all games are indistinguishable through our analysis.

**Game$_{\text{Real}}$** This game is identical with the adaptive security model. All keys and the challenge ciphertext are normal in this game.

**Game$_i^N$** This game is identical with **Game$_{\text{Real}}$** except the types of the first $i$ keys and the challenge ciphertext. In this game, the first $i - 1$ keys and the challenge ciphertext are semi-functional and the $i^{th}$ key is an NSF key.

**Game$_i^T$** This game is identical with **Game$_i^N$** except the type of the $i^{th}$ key. In this game, the $i^{th}$ key is a TSF key.

**Game$_i$** This game is identical with **Game$_i^T$** except the type of the $k^{th}$ key. In this game, the $i^{th}$ key is semi-functional.

**Game$_{\text{Final}}$** This game is identical with **Game$_{q_t}$** except the message encrypted in the challenge ciphertext where $q_t$ is a total number of queries from the adversary. In this game, the message encrypted is replaced a random.

### 4.2.1 Semi-functional Ciphertext Invariance

First, to prove the invariance between **Game$_{\text{Real}}$** and **Game$_0$** (i.e. semi-functional ciphertext invariance), we additionally define **Game$_{\text{Real}, j}$**; $\forall j \in [0, ms]$ using $E_j$ ciphertext if we write $\mathcal{R}_s = \mathbb{Z}_p^{ms}$.

**Game$_{\text{Real}, j}$** This game is identical with **Game$_{\text{Real}}$** except the type of the challenge ciphertext. In this game, the challenge ciphertext is $E_j$. It should be noted that **Game$_{\text{Real}, ms}$** is identical with **Game$_0$** since a $E_{ms}$ ciphertext is identical with a semi-functional ciphertext by the definitions.

We first show that **Game$_{\text{Real}}$** and **Game$_{\text{Real}, 0}$** are invariant in lemma 4.1.1. Then, in lemma 4.1.2, it is proved that **Game$_{\text{Real}, i-1}$** and **Game$_{\text{Real}, i}$** is indistinguishable for all $i \in [1, ms]$. We provide table 4.2 to summarize these processes.

**Lemma 4.1.** Suppose there exists a PPT adversary who distinguishes **Game$_{\text{Real}}$** and **Game$_0$** with a non-negligible advantage $\epsilon$, then an algorithm which breaks Assumption 1 (LW1) can be built with the advantage $\epsilon$ using the adversary.

**Proof:** This lemma is proved by lemmas 4.1.1. and 4.1.2. \hfill \Box
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

Table 4.2: Summary of Semi-Functional Ciphertext Invariance

<table>
<thead>
<tr>
<th>Exponents of semi-functional elements of the challenge CT for y</th>
<th>Type of the challenge CT</th>
<th>Games</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c(y, (1, h')); 0; (0, ..., 0))$</td>
<td>Normal</td>
<td>Game$_{Real}$</td>
<td>Game$<em>{Real} \approx$ Game$</em>{Real, 0}$ (by lemma 4.1.1)</td>
</tr>
<tr>
<td>$c(y, (1, h'); s'_1; (0, ..., 0))$</td>
<td>E$_0$</td>
<td>Game$_{Real, 0}$</td>
<td></td>
</tr>
<tr>
<td>$c(y, (1, h'); s'_1, (s'_1, 0, 0, ..., 0))$</td>
<td>E$_1$</td>
<td>Game$_{Real, 1}$</td>
<td>Game$<em>{Real, i} \approx$ Game$</em>{Real, i+1}$</td>
</tr>
<tr>
<td>$c(y, (1, h'); s'_1, s'_2, s'<em>3, ..., [s'</em>{m_y}])$</td>
<td>NSF (=E$_m$)</td>
<td>Game$_0$ *</td>
<td>by lemma 4.1.2</td>
</tr>
</tbody>
</table>

*: Game$_0$ is equal to Game$_{Real, m_y}$ by the definitions.

Lemma 4.1.1. Suppose there exists a PPT adversary $A$ who distinguishes Game$_{Real}$ and Game$_{Real, 0}$ with a non-negligible advantage $\epsilon$, then an algorithm which breaks Assumption 1 (LW1) can be built with the advantage $\epsilon$ using $A$.

Proof: In this proof, we will show that the hardness to distinguish Game$_{Real}$ and Game$_{Real, 0}$ is reduced to LW1 assumption. Therefore, using the given instance from LW1

$$\{f_1, f_1, f_1c^2, f_1c, f_1^2, f_1c^3, f_1d, f_1cd, f_1c^2d, f_1c^3d, T \in G_1, f_2, f_2^2 \in G_2\}$$

$B$ will simulate Game$_{Real}$ and Game$_{Real, 0}$ depending on the value of $T$ using $A$ to break the assumption.

Setup: $B$ randomly selects $\alpha, y_g, y_u \in \mathbb{Z}_p$, $\vec{h}', \vec{h}'' \in \mathbb{Z}_p^m$ and it sets

$$g_1 = f_1c^2 f_1^{y_g}, g_1^a = (f_1c^2)^{\vec{h}} f_1^{\vec{h}''}, g_1^a = f_1c^2 (f_1d)^{y_g}, g_1^a = (f_1c^2)^{h} (f_1d)^{h''}$$

$$g_2 = (f_2c)^{\alpha} f_2^{y_g}, g_2 = f_2c^2 f_2^{y_g}, g_2 = (f_2c)^{h} f_2^{h''}, g_2 = f_2c, f_2^c, u_2 = f_2^{y_g}.$$ 

The values of $\vec{h}$ are set implicitly since the simulator does not know the value of $c^2$. Also, it implies that $\tau = c + ay_u$. It publishes public parameters as follows:

$$g_1, g_1^a, g_1^a, g_1^{1-h}, g_1^{1-h}, g_1^{c} = f_1c^3 (f_1c^2)^{y_g} (f_1c)^{y_g} (f_1d)^{y_g}, g_1^{1-h} = (f_1c^3)^{h} (f_1c^2)^{h''} (f_1d)^{h''} (f_1d)^{h''}$$

$$e(g_1, g_2)^{\alpha} = e(f_1c^3, f_2c)^{\alpha} e(f_1c^2, f_2)^{2\alpha y_g} e(f_1, f_2)^{\alpha y_g^2}.$$ 

Phase I and II: $B$ randomly selects $\vec{z} \in \mathbb{Z}_p^{m_y}$ and $\vec{r} \in \mathcal{R}$, where $m_y = |\vec{k}(\alpha, x, (1, \vec{h}'); \vec{r})|.$ Then, it implicitly sets

$$\vec{z} = \vec{z} - k(\alpha c, x, (c, c \cdot \vec{h}'); \vec{r}).$$
\[ \vec{z} \text{ is randomly distributed due to } \vec{y}. \text{ To create normal keys, } \mathcal{B} \text{ sets} \]
\[ \vec{K}_0 = f_2^{\vec{x}(\alpha y_g \cdot x, (y_g \vec{h}''_r)^r)} \cdot (f_2^c)^{\vec{z}} \]
\[ \vec{K}_1 = f_2^{y_g \vec{z}} (f_2^c)^{-y_g \vec{k}(\alpha, x, (1, \vec{h}'), \vec{f})}, \quad \vec{K}_2 = f_2^{-\vec{z}} (f_2^c)^{\vec{k}(\alpha, x, (1, \vec{h}'), \vec{f})} \]

Using \( f_2^c, \vec{K}_0, \vec{K}_1 \) and \( \vec{K}_2 \) can be calculated. \( \vec{K}_0 \) is properly distributed since
\[ \vec{K}_0 = f_2^{\vec{x}(\alpha y_g \cdot x, (y_g \vec{h}''_r)^r)} \cdot f_2^{\vec{z}} \]
\[ = f_2^{\vec{k}(\alpha y_g + \alpha c^2, x, (y_g + c^2, \vec{h}''_r + c^2 \vec{h}''_r)^r)} \cdot f_2^{-\vec{z}} (f_2^c)^{\vec{k}(\alpha, x, (1, \vec{h}'), \vec{f})} f_2^{\vec{z}} \]
\[ = g_2^\vec{e}(\alpha, x, (1, \vec{h}'; \vec{f}) \cdot f_2^{\vec{z} - \vec{c}} (f_2^c)^{\vec{k}(\alpha, c, c \vec{h}'). \vec{f})} \]
\[ = g_2^\vec{e}(\alpha, x, (1, \vec{h}'; \vec{f}) \cdot \vec{v}_2. \]

The equalities (4.1) and (4.2) hold because of linearity over common parameters.

**Challenge:** When the adversary asks the challenge ciphertext with messages \( M_0 \) and \( M_1 \). \( \mathcal{B} \) randomly chooses \( \beta \) from \( \{0, 1\} \) and \( \vec{s} \) from \( \mathcal{R}_s \) and sets \( s = d \). The value of \( d \) does not used anywhere else. Therefore, this setting is hidden to the adversary.

The algorithm calculates the challenge ciphertexts as follows:

\[ C = M_\beta \cdot e(f_1^{|x|^d}, f_2^{|y|^d})^a e(f_1^{|x|^d}, f_2^{|y|^d})^{2\alpha |y| d} e(f_1^{|x|^d}, f_2^{|y|^d})^{\alpha |y| d^2}; \]
\[ \vec{C}_0 = (f_1^{|x|^d}) \vec{c}(y_1, (1, \vec{h}''_r); 1, 0) (f_1^{|y|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 1, 0) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s})), \]
\[ \vec{C}_1 = T \vec{c}(y_1, (1, \vec{h}''_r); 1, 0) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) (f_1^{|ad|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 1, 0) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}), \]
\[ \vec{C}_2 = C_0^\vec{e} \cdot C_1^\vec{y}. \]

\( \vec{C}_2 \) is calculable since \( f_1^{|x|^d}, f_1^{|x|^d}, f_1^{|x|^d} \) and \( f_1^{|ad|^d} \) are given. If \( T = f_1^{|ad|^d} \), the challenge ciphertext is the normal challenge ciphertext because

\[ \vec{C}_1 = (f_1^{|x|^d}) \vec{c}(y_1, (1, \vec{h}''_r); 1, 0) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) (f_1^{|ad|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 1, 0) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) \]
\[ = f_1^{|x|^d} \vec{c}(y_1, (1, \vec{h}''_r); 0, \vec{s}) f_1^{|ad|^d} \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) \]
\[ = f_1^{|x|^d} \vec{c}(y_1, (1, \vec{h}''_r); 0, \vec{s}) f_1^{|ad|^d} \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) (f_1^{|x|^d}) \vec{c}(y_1, (y_g \vec{h}''_r); 0, \vec{s}) \]
\[ = g_1^\vec{a} \vec{c}(y_1, (1, \vec{h}''_r); 0, \vec{s}). \]
The equalities (4.4) and (4.6) hold due to linearity over common parameters. Also, the equalities of (4.3) and (4.5) hold due to linearity over random values.

If $T$ is a random tuple and we let $T = f_1^{ac^2d} f_1^c$, then $f_1^{a_c(y(1,h);γ,0)}$ is multiplied to $\vec{C}_1$ above. It results in

$$\vec{C}_1 = g_1^{a_c(y(1,h);d,γ,0)} f_1^{a_c(y(1,h);γ,0)}, \quad \vec{C}_2 = g_1^{(c+ay_a)c(y(1,h);d,γ,0)} f_1^{y_a c(y(1,h);γ,0)}.$$

Therefore, $\vec{C}_1$ is either a normal ciphertext or an $E_0$ ciphertext depending on the value of $T$. Therefore, if $T = f_1^{ac^2d}$, the algorithm has properly simulated $\text{Game}_{\text{Real}}$. Otherwise, it has simulated $\text{Game}_{\text{Real},0}$.

In lemma 4.1.2, we will show that the invariance of $\text{Game}_{\text{Real},0}$ and $\text{Game}_0$. $\text{Game}_0$ is identical with $\text{Game}_{\text{Real},m_s}$ because $E_{m_s}$ ciphertext is identical with a semi-functional ciphertext by the definitions if we write $R_s = \mathbb{Z}_p^{m_s}$. Therefore, we can show the invariance of two games by showing $\text{Game}_{\text{Real},i-1}$ and $\text{Game}_i$ for all $i \in [1, m_s]$.

**Lemma 4.1.2.** Suppose there exists a PPT adversary $A$ who distinguishes $\text{Game}_{\text{Real},k-1}$ and $\text{Game}_k$ with a non-negligible advantage $ε$, then an algorithm $B$ which breaks Assumption 1 (LW1) can be built with the advantage $ε$ using $A$.

**Proof:** Using the given instances from LW1 assumption, $B$ will simulate $\text{Game}_{\text{Real},k-1}$ and $\text{Game}_{\text{Real},k}$ depending on the value of $T$ using $A$ who breaks the assumption with non-negligible advantage.

**Setup and Phase I and II** are simulated identically to those of lemma 4.1.1.

**Challenge:** When the adversary asks the challenge ciphertext with messages $M_0$ and $M_1$, $B$ randomly chooses $β \in \{0, 1\}$ and $s, s_1, \ldots, s_{k-1}, s_{k+1}, \ldots s_{m_s}, s', s'_1, \ldots, s'_{k-1} \in \mathbb{Z}_p$. It sets $\vec{s} = (s_1, \ldots, s_{k-1}, d, s_{k+1}, \ldots, s_{m_s})$ and $\vec{s}_{k-1} = (s'_1, \ldots, s'_{k-1}, 0, \ldots, 0)$. The value of $d$ has never been revealed. Therefore, $\vec{s}$ is uniformly random to the adversary. It calculates the challenge ciphertext as

$$\vec{C} = M_β \cdot e(f_1^{a_c(1)}, f_2^{α_c}) e(f_1^{a_c}, f_2^{α_c}) e(f_1^{a_c}, f_2^{α_c}) e(f_1^{a_c}, f_2^{α_c})$$

$$\vec{C}_0 = (f_1^{a_c} c(y(1,h);s,\vec{s}−d \cdot \vec{I}_k)) (f_1^{a_c} c(y(1,h);0,\vec{I}_k)) f_1^{a_c(y(1,h);s,\vec{s}−d \cdot \vec{I}_k)} f_1^{a_c(y(1,h);0,\vec{I}_k)} f_1^{a_c(y(1,h);0,\vec{I}_k)}$$

$$\vec{C}_1 = (f_1^{a_c} c(y(1,h);s,\vec{s}−d \cdot \vec{I}_k)) T c(y(1,h);0,\vec{I}_k) (f_1^{a_c} c(y(1,h);s,\vec{s}−d \cdot \vec{I}_k))$$

and sets $\vec{C}_2 = \vec{C}_0 \cdot \vec{C}_1^{γ_2}$ where $\vec{I}_k$ is a vector of which only the $k^{th}$ coordinate is 1 and all other coordinates are 0. It should be noted that $\vec{s}−d \cdot \vec{I}_k$ is equal to $(s_1, \ldots, s_{k-1}, 0,$
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

44

$s_{k+1}, ..., s_{m_r}$). Hence, it does not have $d$ as a coordinate. Therefore, $C_0$ and $C_1$ can be calculated by the given instances. Also, $C_2$ is calculable since $f_1^{c_1}, f_1^{c_2}, f_1^{a_1}$ and $f_1^{a_2}$ are given in the instance.

If $T = f_1^{ac_2d}$, the challenge ciphertext is the normal challenge ciphertext since

$$C_1 = f_1^{ac_2d}c(y_1, (1, \hat{h}'); \gamma, 0, \gamma - r_k) f_1^{ac_2d}c(y_1, (1, \tilde{h}'); 0, 1_k) f_1^{ac_2d}c(y_1, (1, \tilde{h}'); 0, 1_k)$$

$$= f_1^{ac_2d}c(y_1, (1, \hat{h}'); \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'); 0, \gamma - r_k)$$

$$= f_1^{ac_2d}c(y_1, (c^2 + y_1, c^2 \tilde{h}'; 0, s, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, s, \gamma)$$

$$= f_1^{ac_2d}c(y_1, (1, \hat{h}'); \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, s, \gamma)$$

$$= f_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, s, \gamma)$$

$$= f_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, s, \gamma)$$

Otherwise, we let $T = f_1^{ac_2d}f_1^{\gamma}$. Then, $f_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma)$ is multiplied to $C_1$. Hence,

$$C_1 = g_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma)$$

$$= g_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma)$$

$$= g_1^{ac_2d}c(y_1, (1, \hat{h}'; \gamma, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma) f_1^{ac_2d}c(y_1, (1, \tilde{h}'; 0, \gamma)$$

where $s'_k = (s'_1, ..., s'_{k-1}, \gamma, 0, ..., 0)$.

If $T = f_1^{ac_2d}$, the type of the challenge ciphertext is $E_{k-1}$ and the algorithm has properly simulated $\text{Game}_{\text{Real}, k-1}$. Otherwise, the type is $E_k$ and $\text{Game}_{\text{Real}, k}$ has been simulated.

\[\Box\]

4.2.2 Semi-functional Key Invariance

To prove the invariance between $\text{Game}_{i-1}$ and $\text{Game}_i$, we define the security games $\text{Game}_{i,j}^N$ and $\text{Game}_{i,j}^T$ for $j \in [0, m_r]$ using NE, and TE, keys.

$\text{Game}_{i,j}^N$ This game is identical with $\text{Game}_{i,j}^N$ except the types of the $i$th key. In this game, the $i$th key is an NE key.

$\text{Game}_{i,j}^T$ This game is identical with $\text{Game}_{i,j}^T$ except the types of the $i$th key. In this game, the $i$th key is a TE key.

It should be noted that $\text{Game}_{i,0}^N$ and $\text{Game}_{i,m_r}^N$ are identical to $\text{Game}_{i-1}^N$ and $\text{Game}_i^N$ resp. by the definitions of keys. Also, due to the same reason, $\text{Game}_{i,0}^T$ is equal to $\text{Game}_i$ and $\text{Game}_{i,m_r}^T$ is equal to $\text{Game}_i^T$. To prove this invariance, the type of the $i$th key changes as table 4. The invariance between $\text{Game}_{i,j}^N$ and $\text{Game}_{i,j}^T$ are proved in lemma 4.2.1. Also, the invariance between $\text{Game}_{i,j}^N$ and $\text{Game}_{i,j}^T$ key is proved by lemmas 4.2.2 and 4.2.3 using computational $\alpha$ hiding. Finally, the invariance between $\text{Game}_{i,j}^T$ and $\text{Game}_{i,j-1}^T$ is showed in lemma 4.2.4.
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

<table>
<thead>
<tr>
<th>Table 4.3: Summary of Semi-functional Key Invariance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponents of semi-functional elements of the $i$th key for $x$</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>$\vec{k}(0, x, (1, h')); (0, ..., 0))$</td>
</tr>
<tr>
<td>$\vec{k}(0, x, (1, h')); (r_{1}', 0, 0, ..., 0))$</td>
</tr>
<tr>
<td>$\vec{k}(0, x, (1, h'); (r_{1}', r_{2}', 0, ..., 0))$</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$\vec{k}(0, x, (1, h'); (r_{1}', r_{2}', r_{3}', ... r_{m}, 0))$</td>
</tr>
<tr>
<td>$\vec{k}(\alpha', x, (1, h'); (r_{1}', ... , r_{m-2}', r_{m-1}', r_{m}'))$</td>
</tr>
<tr>
<td>$\vec{k}(\alpha', x, (1, h'); (r_{1}', ... , r_{m-2}', r_{m-1}', 0))$</td>
</tr>
<tr>
<td>$\vec{k}(\alpha', x, (1, h'); (r_{1}', ... , r_{m-2}', 0, 0))$</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>$\vec{k}(\alpha', x, (1, h'); (0, ... , 0, 0, 0))$</td>
</tr>
</tbody>
</table>

* : $\alpha'$ is projected by computational $\alpha$ hiding in lemmas 4.2.2 and 4.2.3.

Lemma 4.2. (Semi-functional Key Invariance) Suppose there exists a PPT adversary who distinguishes $\text{Game}_{k-1}$ and $\text{Game}_{k}$ with a non-negligible advantage $\epsilon$, then an algorithm which breaks Assumption 2 (LW2) or computational $\alpha$ hiding can be built with the advantage $\epsilon$ using the adversary.

Proof: This lemma is proved by lemmas 4.2.1. to 4.2.4. □

Lemma 4.2.1. Suppose there exists a PPT adversary $A$ who distinguishes $\text{Game}_{k,j-1}^{N}$ and $\text{Game}_{k,j}^{N}$ with a non-negligible advantage $\epsilon$, then an algorithm $B$ which breaks Assumption 2 (LW2) can be built with the advantage $\epsilon$ using $A$.

Proof: Using the given instance $\{f_1, f_1^d, f_1^{t_1}, f_1^{w}, f_1^{t_1w}, f_1^{d_1t}, f_2, f_2^c, f_2^d, f_2^w, T \in G_2\}$, $B$ will simulate either $\text{Game}_{k,j}^{N}$ or $\text{Game}_{k,j-1}^{N}$ using $A$ to break LW2 assumption.

Setup: $B$ randomly chooses $\alpha \in D, a, y' \in \mathbb{Z}_p, \vec{r}, \vec{h}, \vec{v} \in \mathcal{H}$. Then, it sets MSK as $g_2 = f_2^d, g_2^a = (f_2^d)^{\alpha}, g_2^h = (f_2^d)^{\alpha}, g_2^v = f_2^{d_2}, u_2 = f_2^{d_2}, f_2$. This sets $\tau = d - aw + y' + aw = d + y'$ and implicitly sets $b = 1/d$. It publishes public parameters as

$$g_1 = f_1^d, g_1^{\vec{h}} = (f_1^d)^{\vec{r}} f_1^{d}, g_1^{a \vec{h}} = f_1^{d}, \quad g_1^{\tau \vec{h}} = (f_1^{d_1})^{\vec{r} + y' \vec{r}} (f_1^{d_2})^{\vec{r}} e(g_1, g_2)^{\alpha} = e(f_1^{d_1}, f_2^d)^{\alpha}.$$

Phase I and II: The algorithm knows all MSK. Therefore, it can create the normal keys for ($> k$). For the first $k - 1$ key ($< k$), $B$ first generates a normal key. Then,
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

it randomly selects $\alpha'$ from $\mathcal{D}$ and creates an SF key. This is possible since $\mathcal{B}$ knows $a, \alpha', x$ and $f_2$.

For the $k^{th}$ key, it randomly selects $\vec{z}'$ from $\mathbb{Z}_p^{m_k}$ where $m_k = |\vec{k}|$ and sets $\vec{z} = \vec{z}' + \vec{k}(0, x, (1, \vec{h}'), c \cdot \vec{1}_j)$ where $\vec{1}_j$ is a vector of which only the $j^{th}$ coordinate is 1 and all other coordinates are 0. Then, it randomly chooses $\vec{r}'$ from $\mathcal{R}_x$ and sets $\vec{r} = \vec{r}' - c \cdot \vec{1}_j$. $\vec{z}$ and $\vec{r}$ are randomly distributed because of $\vec{z}'$ and $\vec{r}'$. It also generates $r_1', ..., r_{j-1}'$ from $\mathbb{Z}_p$ and sets $\vec{r}_{j-1}' = (r_1', ..., r_{j-1}', 0, 0, 0) \in \mathcal{R}_x$.

$$K_0 = (f_d(\vec{k}((0, x, (1, \vec{h}')); s')) f_2^2(0, x, 0, \vec{h}'; \vec{r}')) (f_d(\vec{k}(0, x, 0, \vec{h}'; \vec{1}_j)) (f_d(\vec{f}_2^{\vec{r}'}) - a f_d^{\vec{r}'})^{\vec{z}}$$

$$\cdot \cdot \cdot (f_d(\vec{f}_2^{\vec{r}'}) - a f_d^{\vec{r}'})^{\vec{r}'_{j-1}}$$

$$K_1 = (f_d(\vec{f}_2^{\vec{r}'})^{\vec{z}} \cdot (f_d(\vec{f}_2^{\vec{r}'}) - a f_d^{\vec{r}'})^{\vec{r}'_{j-1}}$$

If $T = f_2^{\vec{r}'}$, then this is a properly distributed NE key by linearity over common parameters and random values. Otherwise, if $T$ is a random and we let $f_2^{\vec{r}'+\gamma}$ denote $T$, this is the properly distributed NE key since this implicitly sets $\vec{r}_j' = \vec{r}_{j-1}' + \gamma \cdot \vec{1}_j$. It is worth noting that $\vec{r}_j'$ is uniformly random because $\gamma$ is randomly distributed.

Challenge: When the adversary requests the challenge ciphertext with two message $M_0$ and $M_1$, $\mathcal{B}$ randomly selects $\beta$ from $\{0, 1\}$. Then, it randomly selects $s'', \tilde{s} \in \mathbb{Z}_p$ and $\vec{s}'', \tilde{s}' \in \mathcal{R}_x$. Then, it implicitly sets $s = wt \tilde{s} + s''$, $s' = -d^2 t \tilde{s}$, $s'' = wt \tilde{s} + s''$ and $s' = -d^2 t \tilde{s}'$. Because of $s'', \tilde{s}$, $\tilde{s}'$ and $\vec{s}'$, they are randomly distributed. $\mathcal{B}$ sets $C = M_\beta \cdot e(f_1^{\vec{d}a}, f_2^{\vec{d}}) a \tilde{e}(f_1^{\vec{d}}, f_2^{\vec{d}})^a$ and the others as

$$\tilde{C}_0 = (f_1^{\vec{d}a})^a f_1^{\vec{a}} e(f_1^{\vec{d}a}, f_2^{\vec{d}})^a$$

$$\tilde{C}_1 = (C_0)^a f_1^{\vec{d}a} e(f_1^{\vec{d}a}, f_2^{\vec{d}})^a$$

$$\tilde{C}_2 = (f_1^{\vec{d}a})^a f_1^{\vec{a}} e(f_1^{\vec{d}a}, f_2^{\vec{d}})^a$$

This is the properly distributed challenge ciphertext (see the analysis following).

Analysis of equations:
If \( T = f_2^\gamma \), then \( \vec{K}_0, \vec{K}_1 \) and \( \vec{K}_2 \) are properly distributed NE\(_{j-1}\) since
\[
\begin{align*}
\vec{K}_0 &= (f_2^d) \vec{E}(\alpha, (1, \vec{R}^c); \vec{r}^c) \int_{2} \vec{k}(0, x, (0, \vec{R}^c); \vec{r}^c) \left( f_2^d \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) \right) \left( f_2^d \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) \right) \vec{r} \\
&= \int_{2} \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) f_2^d \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) \vec{r} \left( f_2^d \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) \right) \left( f_2^d \vec{k}(0, x, (0, \vec{R}^c); \vec{r}) \right) \vec{r}
\end{align*}
\]
(4.7)

This implicitly sets \( \vec{r} = \vec{r}' - c \cdot \vec{1}_j \) and \( \vec{z} = \vec{z}' + \vec{k}(0, x, (1, \vec{R}^c); c \cdot \vec{1}_j) \). The second equality (4.7) in above equation holds by the linearity of random values because
\[
\begin{align*}
\left( f_2^d \vec{k}(\alpha, x, (1, \vec{R}^c); \vec{r}) \right) &= \left( f_2^d \vec{k}(\alpha, x, (1, \vec{R}^c); \vec{r}) \right) f_2^d \vec{k}(0, x, (1, \vec{R}^c); \vec{r})
\end{align*}
\]
(4.8)

The third equality (4.8) holds because of the definition of \( \vec{r} = \vec{r}' - c \cdot \vec{1}_j \) and linearity over random values. The equalities (4.9) and (4.10) hold due to linearity over common parameters.
\[
\begin{align*}
\vec{K}_1 &= (f_2^w) \vec{E}(\alpha, (1, \vec{R}^c); \vec{r}) \int_{2} \vec{k}(0, x, (1, \vec{R}^c); \vec{r}) \vec{r} \left( f_2^d \vec{k}(0, x, (1, \vec{R}^c); \vec{r}) \right) \left( f_2^d \vec{k}(0, x, (1, \vec{R}^c); \vec{r}) \right) \vec{r}
\end{align*}
\]
(4.9)

If \( T \) is random in lemma 4.2.1. and we let \( f_2^{w+\gamma} \) denote it, this is properly distributed NE\(_{j}\) since \( (f_2^d) \vec{k}(0, x, (1, \vec{R}^c); \vec{r}) \) is multiplied to \( \vec{K}_1 \). By linearity over random values, this implicitly sets \( \vec{r}'_j = \vec{r}'_{j-1} + \gamma \cdot \vec{1}_j \). \( \vec{r}'_j \) is still randomly distributed since \( \gamma \) is a random value.
The challenge ciphertext is also properly distributed because

\[
\tilde{C}_0 = \left( f^{dwt}_1 \right)^2 \bar{c}_{y,t}(1,\tilde{H}';\tilde{s},\tilde{t})(f^{dwt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}';s',\tilde{s}') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';\tilde{s},\tilde{t}) (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]

\[
= \left( f^{dwt}_1 \right)^2 \bar{c}_{y,t}(y,d\tilde{H}';\tilde{s},\tilde{t}) (f^{dwt}_1)^2 \bar{c}_{y,t}(y,d\tilde{H}';s',\tilde{s}') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';\tilde{s},\tilde{t}) (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]  
(4.11)

\[
= \left( f^{dwt}_1 \right)^2 \bar{c}_{y,t}(y,d\tilde{H}';\tilde{s},\tilde{t}) \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}';s',\tilde{s}') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';\tilde{s},\tilde{t}) (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]  
(4.12)

\[
= f_1^{dwt} \bar{c}_{y,t}(y,d\tilde{H}';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f_1^{dwt})^2 \bar{c}_{y,t}(y,\tilde{H}';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]  
(4.13)

\[
= f_1^{dwt} \bar{c}_{y,t}(y,d\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]  
(4.14)

\[
= g_1 \bar{c}_{y,t}(y,\tilde{H}';s,\tilde{t})
\]

\[
\tilde{C}_1 = (C_0)^a \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}';\tilde{s},\tilde{t}) = g_1^a \bar{c}_{y,t}(y,\tilde{H}';s,\tilde{t}) (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]

\[
\tilde{C}_2 = \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}') (f_1^{dwt})^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]

\[
= \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f_1^{dwt})^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]  
(4.15)

\[
= f_1^{dwt} \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]  
(4.16)

\[
= g_1 \bar{c}_{y,t}(y,\tilde{H}'';s,\tilde{t}) (f^{wt}_1)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}')
\]

The equalities of (4.11) and (4.14) hold by linearity of common parameters. Also, those of (4.12) and (4.13) hold by linearity of random values. The equalities of (4.15) holds since

\[
\left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}') = \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f_1^{dwt})^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]

\[
\left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';s',\tilde{s}') = \left( f_1^{dwt} \right)^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''') (f_1^{dwt})^2 \bar{c}_{y,t}(y,\tilde{H}'';wt\tilde{s}+s'',wt\tilde{t}\tilde{s}+s''')
\]

It is worth noting that all equality above holds by linearity of random values. The last equalities in \( \tilde{C}_0 \) \( \tilde{C}_1 \) and \( \tilde{C}_2 \) holds because of \( s' = -d^2 t\tilde{s}, \tilde{s}' = -d^2 t\tilde{s} \) and the definitions of public parameters. \( \tilde{s} \) and \( \tilde{t} \) are randomly distributed to the adversary. They are also appear in \( s = wt\tilde{s} + s'', \tilde{s} = wt\tilde{s} + s'' \), but their values are not revealed because of \( s'' \) and \( \tilde{s}'' \).

**Lemma 4.2.2.** Suppose there exists a PPT adversary who distinguishes Game\(_N^k\) and Game\(_I^k\) with a non-negligible advantage \( \epsilon \) for \( k \leq q_1 \) where \( q_1 \) is the number of key queries in Phase I, then an algorithm which breaks co-selective \( \alpha \) hiding can be built with the advantage \( \epsilon \) using the adversary.
**Proof:** Since $k \leq q_1$, a type $k$ query is queried before the type $c$ query, since the $k^{th}$ key is requested in advance of the challenge ciphertext. Hence, $B$ breaks co-selective $\alpha$ hiding (i.e. $B$ distinguishes whether the oracle it works with is $O_{CAH}^1$ and $O_{CAH}^2$) using the adversary $A$ who distinguishes between $\text{Game}_k^N$ and $\text{Game}_k^T$.

**Setup:** $B$ makes an initial query to the oracle it works with. The oracle replies $\{f_1, f_2\}$. Then, $B$ randomly selects $\alpha \in D, y_g, a, b, y_u, y_v, h_1, ..., h_n \in Z_p$ and set $\tau = y_v + a \cdot y_u$, $\vec{h} = (h_1, ..., h_n)$, $b = y_g^{-1}$, $g_1 = f_1^{y_g}$ and $g_2 = f_2^{y_g}$. It publishes public parameters $\{g_1, g_1^{\vec{a}}, g_2, g_2^{\vec{b}}, g_1^{\vec{a}}, g_1^{\vec{b}}, e(g_1, g_2)^\alpha\}$. It sets MSK as $\{g_2, g_2^\alpha, g_2^\vec{a}, v_2 = f_2^{y_g}, u_2 = f_2^{y_u}, f_2\}$.

**Phase I and II:** For the first $k-1$ keys, $B$ generates a normal key $(\vec{K}_0', \vec{K}_1', \vec{K}_2')$ using the key generation algorithm and MSK. It randomly chooses $\alpha'$ from $D$ and sets $\vec{K}_0 = \vec{K}_0' \cdot f_2^{-\alpha'(\vec{y}, x, (0,0) \vec{y})}$, $\vec{K}_1 = \vec{K}_1' \cdot f_2(\vec{y}, x, (0,0) \vec{y})$, $\vec{K}_2 = \vec{K}_2'$. For the rest of keys except the $k^{th}$ key ($> k$), the algorithm responds to the key queries by sending a normal key. This is possible since $B$ knows all PK and MSK.

When the adversary requests the $k^{th}$ key, the algorithm creates a normal key $(\vec{K}_0', \vec{K}_1', \vec{K}_2')$ and requests a type $k$ response to the oracle it works with. We let $f_2(\vec{y}, (1, \vec{y}'), \vec{x}')$ denote the response from the oracle. $B$ does not know whether $\alpha''$ is 0 or a random value since it does not know the type of the oracle. It sets

\[
\vec{K}_0 = \vec{K}_0'(f_2^{\alpha''(\vec{y}, x, (1, \vec{y}'; \vec{x}')}) - \alpha', \quad \vec{K}_1 = \vec{K}_1' f_2^{\alpha''(\vec{y}, x, (1, \vec{y}'; \vec{x}')}, \quad \vec{K}_2 = \vec{K}_2'.
\]

Finally, it sends the key to the adversary. It is worth noting that $k^{th}$ key is queried only in phase I since $k \leq q_1$. Therefore, if $\alpha'' = 0$, the oracle which $B$ works with is $O_{CAH}^1$. If $\alpha''$ is a random, it works with $O_{CAH}^2$.

**Challenge:** When the adversary requests the challenge ciphertext with two message $M_0$ and $M_1$, the algorithm randomly selects $\beta$ from $\{0, 1\}$, the algorithm first creates a normal ciphertext $(C', \vec{C}_0', \vec{C}_1', \vec{C}_2')$. Then, it requests a type $c$ response to the oracle it works with. We let $f_1^{\vec{y}, (1, \vec{y}'; \vec{x}')}$ denote the response from the oracle. It sends to the adversary

\[
C = C', \quad \vec{C}_0 = \vec{C}_0', \quad \vec{C}_1 = \vec{C}_1 f_1^{\vec{y}, (1, \vec{y}'; \vec{x}')}, \quad \vec{C}_2 = \vec{C}_2 (f_1^{\vec{y}, (1, \vec{y}'; \vec{x}')})^y.
\]

$\square$

**Lemma 4.2.3.** Suppose there exists a PPT adversary who distinguishes $\text{Game}_k^N$ and $\text{Game}_k^T$ with a non-negligible advantage $\epsilon$ for $k > q_1$ where $q_1$ is the number of key queries in Phase I, then an algorithm which breaks selective $\alpha$ hiding can be built with the advantage $\epsilon$ using the adversary.

**Proof:** This proof is almost identical with lemma 4.2.2. except the type of the
or the algorithm works with and the $k^{th}$ key is simulated in phase II ($k > q_1$). In this proof, $B$ works with either $O^1_{SAH}$ or $O^2_{SAH}$ depending on the value of $\alpha''$ since it queries a type $c$ in advance of a type $k$. □

**Lemma 4.2.4.** Suppose there exists a PPT adversary who distinguishes $Game^T_k$ and $Game_k$ with a non-negligible advantage $\epsilon$, then an algorithm which breaks Assumption 2 (LW2) can be built with the advantage $\epsilon$ using the adversary.

**Proof:** This proof is identical with the proof of lemma 4.2.1 except the $k^{th}$ key.

For the $k^{th}$ key, it randomly selects $\alpha' \in D$ and $\bar{z}' \in \mathbb{Z}^{mk}_p$ and sets $\bar{z} = \bar{z}' + \bar{k}(\alpha', x, (1, \bar{h}'), c \cdot \bar{I}_j)$. Then, it chooses $\bar{r}''$ from $\mathcal{R}_r$ and sets $\bar{r} = \bar{r}'' - c \cdot \bar{I}_j$. $\bar{z}$ and $\bar{r}$ are randomly distributed because of $\bar{z}'$ and $\bar{r}''$. It also generates $r'_1, \ldots, r'_{j-1}$ from $\mathbb{Z}_p$ and sets $\bar{r}'_{j-1} = (r'_1, \ldots, r'_{j-1}, 0, 0, 0) \in \mathcal{R}_r$.

$$
\bar{K}_0 = (f^w_2)^{\bar{k}(\alpha + \alpha'.x,(1,\bar{h}''));\bar{r}}(f^w_2)\bar{k}(0,x,(0,\bar{h}''));\bar{r})\cdot \bar{K}_1 = (f^w_2)^{\bar{k}(\alpha',x,(1,\bar{h}''));\bar{r}'}(f^w_2)\bar{k}(\alpha',x,(1,\bar{h}''));\bar{r}'}
$$

If $T = f^{cw}_2$, this is the properly distributed $TE_{j-1}$ key. Therefore, $B$ has well simulated $Game^T_{k,j-1}$. If $T$ is random and we let $f^{cw+\gamma}_2$ denote $T$, this is the properly distributed $TE_{j}$ key since this implicitly sets $\bar{r}'_j = \bar{r}'_{j-1} + \gamma \cdot \bar{I}_j$. Due to $\gamma$, $r'_j$ is properly distributed. Hence, $B$ has simulated $Game^T_{k,j}$.

### 4.2.3 Semi-functional Security

We prove the semi-functional security by showing that $Game_q$ and $Game_{Final}$ are indistinguishable.

**Lemma 4.3.** Suppose there exists a PPT adversary who distinguishes $Game_q$ and $Game_{Final}$ with a non-negligible advantage $\epsilon$, then an algorithm which breaks Assumption 3 (LW3) can be built with the advantage $\epsilon$ using the adversary.

**Proof:** Using a given instance $\{f_1, f^o_1, f^c_1, f^d_1 \in G_1, f_2, f^a_2, f^b_2, f^d_2 \in G_2, T \in G_T\}$, $B$ will simulate either $Game_q$ or $Game_{Final}$ depending on the value of $T$.

**Setup:** $B$ randomly selects $y_g, y_u, y_e \in \mathbb{Z}_p, \bar{h} \in \mathbb{Z}_p^n$ and sets $\alpha = ac, a = a, b = 1/y_g$ and $\tau = y_u + ay_u$. $B$ publishes the public parameters

$$
g_1 = f^{y_g}_1, g_1^\alpha = f^{y_g}_{\bar{y}_h}, g_1^{\bar{h}} = f^{y_u}_{\bar{y}_u}, g_1^{\bar{h}^\alpha} = f^{a y_u}_{\bar{h}},
g_1^{\tau} = f^{y_g y_v}_1 + (f^a_1)^{y_g y_v}, g_1^{\tau} = f^{y_g y_v}_{\bar{h}} + (f^a_1)^{y_g y_v}, e(g_1, g_2)^{\alpha} = e(f^{a y_u}_1, f^{a y_u}_{\bar{x}}_2)
$$
CHAPTER 4. FUNCTIONAL ENCRYPTION IN PRIME ORDER GROUPS

It also sets \( g_2^\alpha = f_2^{\alpha y_{ac}}, g_2 = f_2^{y_{ac}}, \vec{h} = f_2^{y_{ac}}, \vec{v}_2 = f_2^{y_{ac}}, f_2 \). It should be noted that \( g_2^\alpha \) sets implicitly since \( f_2^{ac} \) is not given. The other elements can be calculated since \( f_2 \) is given.

**Phase I and II:** \( B \) randomly selects \( \alpha'' \in D \), \( \vec{z} \in \mathbb{Z}_p^{m_k} \) and \( \vec{r} \in \mathcal{R}_s \) and sets \( \alpha' = y_g c + \alpha'' \).

\[
\vec{K}_0 = f_2^{\vec{K}(\vec{a}'' \cdot x, (y_g, y_g \vec{h}); \vec{r})} v_2^{\vec{z}} \]
\[
\vec{K}_1 = f_2^{\vec{K}(y_g, x, (0, \vec{0}), \vec{0})} f_2^{\vec{K}(\alpha'', x, (0, \vec{0}), \vec{0})}, \quad \vec{K}_2 = f_2^{\vec{z}}
\]

This is a properly distributed key since
\[
\vec{K}_0 = f_2^{\vec{K}(\vec{a}'' \cdot x, (y_g, y_g \vec{h}); \vec{r})} v_2^{\vec{z}} \\
= f_2^{\vec{K}(y_g \vec{ac}, x, (y_g, y_g \vec{h}); \vec{r})} v_2^{\vec{z}} f_2^{\vec{K}(y_{ac} - y_g \alpha, x, (y_g, y_g \vec{h}); \vec{0})} \\
= g_2^{\vec{K}(\alpha x, (1, \vec{h}); \vec{r})} v_2^{\vec{z}} f_2^{a \vec{K}(y_g c + \alpha'', x, (0, 0); \vec{0})} \tag{4.17}
\]
\[
\vec{K}_1 = f_2^{\vec{K}(y_g x, (0, \vec{0}), \vec{0})} f_2^{\vec{K}(\alpha'', x, (0, \vec{0}), \vec{0})}, \quad \vec{K}_2 = f_2^{\vec{z}} \tag{4.18}
\]

The equality of (4.17) holds due to linearity of random values. Also, (4.18) holds by parameter vanishing.

**Challenge:** When the adversary requests the challenge ciphertexts with two messages \( M_0 \) and \( M_1 \), \( B \) randomly selects \( \beta \in \{0, 1\} \), \( s'' \in \mathbb{Z}_p \), \( \vec{s}, \vec{s}' \in \mathcal{R}_s \) and \( \vec{h}'' \in \mathbb{Z}_p \) and sets \( s = d \), \( s' = -y_g a d + a s'' \) and \( \vec{s}' = a \cdot \vec{s}'' \). \( d \) appears both in \( s \) and \( s' \). However, \( s' \) does not reveal the value of \( d \) because of \( s'' \). Therefore, setting \( s = d \) is hidden to the adversary. It calculates the challenge ciphertexts as follows:

\[
C = M \cdot T, \quad \vec{C}_0 = f_1^{y_g \vec{c}(y_g, (1, \vec{h}); s, \vec{s})}
\]
\[
\vec{C}_1 = (f_1^a)(\vec{c}(y_g, (1, \vec{h}); s'', y_g \vec{s} + \vec{s}'')) f_1^{\vec{c}(y_g, (0, \vec{h}''); s'', \vec{s}'')}(f_1^d)(\vec{c}(y_g, (0, \vec{h}''); -y_g, \vec{0}
\quad \vec{C}_2 = C_0^q C_1'^q \tag{4.19}
\]

This implicitly sets \( \vec{h}' = \vec{h} + a^{-1} \vec{h}'' \). \( \vec{C}_1 \) and \( \vec{C}_2 \) are properly distributed since

\[
\vec{C}_1 = (f_1^a)(\vec{c}(y_g, (1, \vec{h}); s'', y_g \vec{s} + \vec{s}'')) f_1^{\vec{c}(y_g, (0, \vec{h}''); s'', \vec{s}'')}(f_1^d)(\vec{c}(y_g, (0, \vec{h}''); -y_g, \vec{0}) \\
\quad = (f_1^a)(\vec{c}(y_g, (1, \vec{h}); y_g d - y_g d + s'', y_g \vec{s} + \vec{s}'')) f_1^{\vec{c}(y_g, (0, \vec{h}''); -y_g d + s' + \vec{s}'')} \tag{4.19}
\]
\[
\vec{C}_2 = C_0^q C_1'^q \tag{4.20}
\]

The equalities of (4.19) and (4.20) holds by linearity of random values. The equality of (4.21) holds because of linearity of common parameters. It should be noted that \( \vec{h}'' \) does not appear anywhere else, it only used in the challenge ciphertext. Therefore, due to \( \vec{h}'' \), \( \vec{h}' \) is randomly distributed. If \( T \) is \( e(f_1, f_2)^{acd} \), this has simulated \( \text{Game}_y \) properly. Otherwise, if \( T \) is a random, a randomness will be added to \( M \). Therefore,
this has simulated $\text{Game}_{\text{Final}}$. \hfill \Box
Chapter 5
Tag Based Encoding

In this chapter, we introduce a new encoding framework which is a tag based encoding. Tag based encoding provides functional encryption schemes which are adaptively secure under the standard decisional linear Assumption (DLIN). Compared with other encoding frameworks, it improves efficiency of resulting schemes when the sizes of predicates used in the schemes are large.

Tag based encoding. We introduce a tag based encoding. For a function $R$ which takes as inputs $X$ and $Y$ and outputs a binary ($R: X \times Y \to \{0, 1\}$), there are two algorithms $kE$ and $cE$ with common values $\vec{h} \in \mathbb{Z}_p^{\ell}$ where $\ell$ is a value allocated for each function $R$ and $p$ is a prime. We let $kE(x, \vec{h})$ and $cE(y, \vec{h})$ denote the outputs of $kE$ taking as inputs $x \in X$ and $\vec{h}$ and $cE$ taking as inputs $y \in Y$ and $\vec{h}$, resp. The tag based encoding must satisfy three essential properties, namely Reconstruction, Linearity and $\vec{h}$-hiding. Instances of our encoding are interpreted as FE schemes via our constructions. These constructions are often called compilers since they compile encodings to form FE schemes (Fig 5.1).

Efficiency improvement Prior to our work, the most efficient compiler in prime order groups was suggested by CGW [CGW15]. It was suggested for the predicate encoding [Wee14]. Multiple compilers under the generalized $k$-linear assumption [EHK+13] were included in CGW’s frameworks. The number of group elements that a compiler in CGW’s framework uses to represent a tuple of an encoding scheme depends on assumptions the compiler is from. In detail, each tuple of an encoding

<table>
<thead>
<tr>
<th>Predicates</th>
<th>Encodings</th>
<th>Construction</th>
<th>Schemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x, y) \in X \times Y$</td>
<td>$kE(x, \cdot)$</td>
<td>Setup</td>
<td>FE</td>
</tr>
<tr>
<td>$R: X \times Y \to {0, 1}$</td>
<td>$cE(y, \cdot)$</td>
<td>KeyGen</td>
<td>(IBE, HIBE, ...)</td>
</tr>
</tbody>
</table>

Figure 5.1: Encoding Frameworks for FE
Table 5.1: Efficiency Comparison between Our Compiler and CGW’s

<table>
<thead>
<tr>
<th></th>
<th>Assump.</th>
<th>PK (by bits)</th>
<th>SK (by bits)</th>
<th>CT (by bits)</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGW [CGW15]</td>
<td>SXDH</td>
<td>3584 + 512 $\ell$</td>
<td>1024 + 1024 $m_k$</td>
<td>3584 + 512 $m_c$</td>
<td>4P + 2 $\ell$ E</td>
</tr>
<tr>
<td></td>
<td>DLIN</td>
<td>7680 + 1536 $\ell$</td>
<td>1536 + 1536 $m_k$</td>
<td>3840 + 768 $m_c$</td>
<td>6P + 3 $\ell$ E</td>
</tr>
<tr>
<td>Ours</td>
<td>DLIN</td>
<td>5888 + 256 $\ell$</td>
<td>3584 + 768 $m_k$</td>
<td>5120 + 512$m_c$</td>
<td>8P + $\ell$ E</td>
</tr>
</tbody>
</table>

$\ell$: the size of predicates, $m_k$ and $m_c$: the size of encoding schemes used for keys and ciphertexts, Bits are calculated based on 128 bits security level [Gui13] ($|G_1| = |Z_p| = 256$ bits, $|G_2| = 512$ bits, $|G_T| = 3072$ bits)

scheme is represented by $k+1$ group elements in private keys and ciphertexts. Also, in public keys, $k(k + 1)$ elements are required for each predicate. The most efficient compiler is under the SXDH assumption (i.e. when $k$ equals to 1). Two group elements are used for a tuple of encoding schemes in this compiler. Other encodings [Att15, AC16] were also suggested, independently, but they are similar with CGW’s framework from an efficiency perspective. Therefore, we mainly compare our compiler with CGW’s compilers to highlight our contribution.

In our compiler, one group element represents each predicate in public keys. If the size of a predicate is large, our compiler reduces the size of public key to 50 percent compared with CGW’s compiler. Also, it reduces decryption time by 50 percent when the decryption process under the same condition. For the other parameters such as private keys and ciphertexts, our compiler needs a group element and an integer for one tuple of an encoding scheme. Because the size of an integer is as small as the size of a group element of $G_1$ but much less than that of $G_2$ due to embedding degree of asymmetric bilinear maps, our compiler reduces the size of either private keys or ciphertexts depending on where $G_2$ is used for. For example, in 128 bits security level, $G_2$ requires at least 512 bits. It is twice of the size of $Z_p$ [Gui13]. It means that only 768 bits are required to represent a tuple in our compiler. This outperforms CGW’s compiler which requires 1024 bits for a tuple. Therefore, our compiler saves the size of private keys or ciphertexts by 25 percent compared to their compiler under the SXDH assumption when the size of an encoding is large.

Moreover, CGW’s framework is also realized under the weaker assumption, DLIN, as our compiler. Also, 6 group elements are required for public keys. It implies that our compiler more outperforms their compiler. In comparison with

---

The DLIN assumption with asymmetric bilinear maps can be featured in various forms since it expanded from the DLIN assumption originally equipped with symmetric paring. The DLIN assumption of CGW’s compiler has a slightly different from our assumption. In particular, it has two less group elements in $G_2$. 
A compiler with symmetric bilinear maps. We additionally provide a new compiler with symmetric bilinear maps. Prior to our work, with symmetric bilinear maps, all encodings [Att14a, Wee14, CGW15] are secure only in composite order groups. It is because all prior encodings [CGW15, Att15, AC16] in prime order groups are based on dual system groups [CW13] which require asymmetric parings to feature different properties of left-hand groups and right-hand groups in pairings. To the best of our knowledge, our construction is the only compiler that provides adaptive security for encodings with symmetric pairings in prime order groups. This gives our framework an additional flexibility when the encryption scheme is implemented under a special requirement of the pairing type.

**Efficiency Improvement under Static Assumptions.** We additionally provide a compiler which improves the efficiency of our compiler with asymmetric bilinear maps.

### Table 5.2: Efficiency Comparison of IPE Schemes Based on Encodings

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Assumption</th>
<th>PK</th>
<th>SK</th>
<th>CT</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGW [CGW15]</td>
<td>SXDH</td>
<td>$(2^t + 4)</td>
<td>G_1</td>
<td>+</td>
<td>G_T</td>
</tr>
<tr>
<td>DLIN</td>
<td>$(\ell + 6)</td>
<td>G_1</td>
<td>+ 2</td>
<td>G_T</td>
<td>$</td>
</tr>
<tr>
<td>Ours</td>
<td>DLIN</td>
<td>$(11 + \ell)</td>
<td>G_1</td>
<td>+</td>
<td>G_T</td>
</tr>
</tbody>
</table>

$t$: the size of a predicate vector, $P$: Pairing computation, $E$: Exponentiation over a group element, $|G_N|$, $|G_N,T|$: the size of group elements of a composite order $N$, $|G_1|$, $|G_2|$ and $|G_T|$: the sizes of group elements of order $p$ of $e : G_1 \times G_2 \rightarrow G_T$

### Table 5.3: Efficiency Comparison of PA-IPE Schemes Based on Encodings

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Assumption</th>
<th>PK</th>
<th>SK</th>
<th>CT</th>
<th>Decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGW [CGW15]</td>
<td>SXDH</td>
<td>$(2^t + 4)</td>
<td>G_1</td>
<td>+</td>
<td>G_T</td>
</tr>
<tr>
<td>DLIN</td>
<td>$(\ell + 6)</td>
<td>G_1</td>
<td>+ 2</td>
<td>G_T</td>
<td>$</td>
</tr>
<tr>
<td>Ours, AL [AL12]</td>
<td>DLIN</td>
<td>$(11 + \ell)</td>
<td>G_1</td>
<td>+</td>
<td>G_T</td>
</tr>
</tbody>
</table>

$t$: the size of a predicate vector, $P$: Pairing computation, $E$: Exponentiation over a group element, $|G_1|$, $|G_2|$ and $G_T$: the sizes of group elements of order $p$ of $e : G_1 \times G_2 \rightarrow G_T$

their compiler under the same assumption with a 128 bits security level, our compiler saves 83 percent in a public key, 50 percent in private keys, 33 percent in ciphertexts and 66 percent in decryption time if the size of encoding is large. We provide table 1 for the details.

To compare the efficiency in practice, we compare our inner product encryption with short keys and public attribute inner product encryption to those of other encodings. The instance of Public Attribute Inner Product Encryption (PAIPE) is taken from [AL12] but it also explained as tag based encodings. It should be noted that encodings for our IPE schemes are slightly different from those of CGW [CGW15] and Wee [Wee14]. Our instances require one or two less elements.
maps. This compiler reduces the size of parameters by constant amounts. As a result, it reduces 7 elements in public parameters, 4 elements of secret keys and ciphertexts from the previous compiler. As a compensation of efficiency improvements, this compiler requires non-standard assumptions, but those assumptions are still simple and static.

New schemes. We introduce a number of new schemes as instances. Inner Product Encryption with short keys, Dual Spatial Encryption with short keys and Hierarchical Identity Based Encryption with short ciphertexts are newly introduced. Particularly, dual spatial encryption is a new primitive. It is a symmetric conversion of a spatial encryption [Ham11]. Therefore, an affine space and an affine vector are taken to generate ciphertexts and keys, respectively. Moreover, we describe as encodings a number of existing schemes such as IBE [Wat09], (Public Attribute) Inner Product Encryption [AL12], Spatial Encryption and Doubly Spatial Encryption [CZF12] to show the versatility of our framework.

5.1 Complexity Assumptions

We expand both the DLIN and the DBDH into asymmetric bilinear maps. We use subscripts to denote the type of groups. For example, \(g_1\) denotes a generator of \(G_1\), and \(g_2\) denotes a generator of \(G_2\).

(Asymmetric) Decisional Bilinear Diffie-Hellman (DBDH) assumption

Given a group generator \(G\), we define the following distribution:

\[
\mathbb{G} = (p, G_1, G_2, G_T, e) \overset{R}{\leftarrow} \mathbb{G}, \quad g_1 \overset{R}{\leftarrow} G_1, \quad g_2 \overset{R}{\leftarrow} G_2, \quad c_1, c_2, c_3 \overset{R}{\leftarrow} \mathbb{Z}_p
\]

\[
D = (g_1, g_1^{c_2}, g_1^{c_3} \in G_1, g_2, g_2^{c_1}, g_2^{c_2} \in G_2)
\]

\[
T_0 = e(g_1, g_2)^{c_1 c_2 c_3}, \quad T_1 \overset{R}{\leftarrow} G_T
\]

We define the advantage of an algorithm \(A\) in breaking (Asymmetric) DBDH to be:

\[
\text{Adv}_{\mathbb{G}, \mathcal{A}}^{\text{DBDH}}(\lambda) := |\Pr[A(D, T_0) = 1] - \Pr[A(D, T_1) = 1]|
\]

(Asymmetric) Decisional Linear (DLIN) assumption

Given a group generator \(G\), we define the following distribution:

\[
\mathbb{G} = (p, G_1, G_2, G_T, e) \overset{R}{\leftarrow} \mathbb{G}, \quad g_1 \overset{R}{\leftarrow} G_1, \quad g_2 \overset{R}{\leftarrow} G_2, \quad y_f, y_v, c_1, c_2 \overset{R}{\leftarrow} \mathbb{Z}_p
\]

\[
D = (g_1, f_1, \nu_1, g_1^{c_1}, f_1^{c_2} \in G_1, g_2, f_2, \nu_2 \in G_2)
\]
where \( f_1 = g_1^{y_f}, \nu_1 = g_1^{w_f}, f_2 = g_2^{y_f} \) and \( \nu_2 = g_2^{w_f} \).

\[
T_0 = \nu_1^{c_1+c_2}, \quad T_1 \leftarrow_R G_1
\]

We define the advantage of an algorithm \( A \) in breaking (Asymmetric) DLIN to be:

\[
\text{Adv}^{\text{ADLIN}}_{A}(\lambda) := |\Pr[A(D, T_0) = 1] - \Pr[A(D, T_1) = 1]|.
\]

It should be noted that (Asymmetric) DBDH assumption also reduced to (Asymmetric) DLIN assumption.

**Proposition 1.** Suppose that there exists an algorithm \( A \) which breaking (Asymmetric) DBDH with non-negligible advantage \( \epsilon \). Then, we can build an algorithm \( B \) which breaks (Asymmetric) DLIN assumption with advantage \( \epsilon \).

**Proof:** \( B \) takes \( \{g_1, f_1, \nu_1, g_1^{\nu_1}, f_1^{\nu_1}, T, g_2, f_2, \nu_2\} \) as an instance from (Asymmetric) DLIN assumption. \( B \) will simulate (Asymmetric) DBDH from the instance using \( A \) who breaks (Asymmetric) DLIN assumption with non-negligible advantage.

If the adversary requests a instance of (Asymmetric) DBDH

\[
\{g_1, g_2, g_1^{\nu_2}, g_2^{\nu_1}, \tilde{T}\}
\]

to break (Asymmetric) DLIN, the algorithm sets

\[
\tilde{g}_1 = g_1, \tilde{g}_2^{\nu_2} = f_1, \tilde{g}_1^{\nu_1} = g_1^{\nu_1}, \tilde{g}_2 = g_2, \tilde{g}_1^{\nu_2} = \nu_2, \tilde{g}_2^{\nu_1} = f_2, \tilde{T} = e(T, f_2)/e(f_1^{\nu_2}, \nu_2).
\]

This implicitly sets \( \tilde{c}_1 = y_\nu, \tilde{c}_2 = y_f \) and \( \tilde{c}_3 = c_1 \) where \( y_\nu \) and \( y_f \) are the discrete logarithms of \( \nu_1 \) and \( f_1 \) to the base \( g_1 \) modulo \( p \), respectively. If \( T \) is \( \nu_1^{c_1+c_2} \), then \( \tilde{T} = e(T, f_2)/e(f_1^{\nu_2}, \nu_2) = e(\nu_1, f_2)^{c_3} = e(\tilde{g}_1, \tilde{g}_2)^{\tilde{c}_3} = e(\tilde{g}_1, \tilde{g}_2)^{\tilde{c}_1 \tilde{c}_2 \tilde{c}_3} \). Otherwise, if \( T \) is a random element from \( G_1, \tilde{T} \) is randomized by \( T \).

\( \square \)

**Simple LW2** Given a group generator \( G \), we define the following distribution:

\[
\mathbb{G} = (p, G_1, G_2, G_T, e) \leftarrow_R G, \quad f_1 \leftarrow_R G_1, \quad f_2 \leftarrow R G_2, \quad a, c, d, z \leftarrow R \mathbb{Z}_p
\]

\[
D = (f_1, f_1^d, f_1^{dw}, f_1^{dt} \in G_1, f_2, f_2^c, f_2^d, f_2^w \in G_2)
\]

\[
T_0 = f_2^w, \quad T_1 \leftarrow_R G_2
\]

We define the advantage of an algorithm \( A \) in breaking Simple LW2 to be:

\[
\text{Adv}^{\text{SimpleLW2}}_{A}(\lambda) := |\Pr[A(D, T_0) = 1] - \Pr[A(D, T_1) = 1]|.
\]
CHAPTER 5. TAG BASED ENCODING

The generic security of Simple LW2 holds trivially since we only remove tuples in instances of the original LW2 assumption [LW10]. We emphasize that Simple LW2 assumption is still simple and static although it is not a standard assumption.

5.2 Tag based encoding

For a predicate \( R : \mathcal{X} \times \mathcal{Y} \to \{0,1\} \) and \( \ell \) which is an integer allocated for a predicate \( R \) (e.g. the size of a universe of attributes in ABE, the dimension of an affine space in spatial encryption), tag based encoding \( \text{TE}(R) \) consists of two deterministic algorithms \((\vec{k}_E, \vec{c}_E)\) where \( \vec{k}_E(x, \vec{h}) \) takes as inputs \( x \in \mathcal{X} \) and \( \vec{h} \in \mathbb{Z}_p^\ell \), and \( \vec{c}_E(y, \vec{h}') \) also takes as inputs \( y \in \mathcal{Y} \) and \( \vec{h}' \in \mathbb{Z}_p^\ell \). We let \( \ell_k \) and \( \ell_c \) denote \(|\vec{k}_E(x, \vec{h})|\) and \(|\vec{c}_E(y, \vec{h}')|\), resp.

Property 1. (Reconstruction) For all \((x, y)\) such that \( R(x, y) = 1 \), there exists an efficient algorithm to compute non-zero vectors \( \vec{m}_x \in \mathbb{Z}_p^{\ell_k} \) and \( \vec{m}_y \in \mathbb{Z}_p^{\ell_c} \) such that

\[
\vec{m}_x \vec{k}_E(x, \vec{h}) = \vec{m}_y \vec{c}_E(y, \vec{h}) \quad \forall \vec{h} \in \mathbb{Z}_p^\ell.
\]

Property 2. (Linearity) For all \((x, y, \vec{h}', \vec{h}'') \in \mathcal{X} \times \mathcal{Y} \times \mathbb{Z}_p^\ell \times \mathbb{Z}_p^\ell,

\[
\vec{k}_E(x, \vec{h}') + \vec{k}_E(x, \vec{h}'') = \vec{k}_E(x, \vec{h}' + \vec{h}'') \quad \text{and} \quad \vec{c}_E(y, \vec{h}') + \vec{c}_E(y, \vec{h}'') = \vec{c}_E(y, \vec{h}' + \vec{h}'').
\]

Property 3. (\( \vec{h} \)-hiding) For all \((x, y)\) \in \mathcal{X} \times \mathcal{Y} \) such that \( R(x, y) = 0 \) and for all \( \vec{h}, \vec{h}' \in \mathbb{Z}_p^\ell \), following two joint distributions are statistically indistinguishable.

\[
(x, y, \vec{k}_E(x, \vec{h}), \vec{c}_E(y, \vec{h})) \quad \text{and} \quad (x, y, \vec{k}_E(x, \vec{h}), \vec{c}_E(y, \vec{h}'))
\]

Remark 5.1. Reconstruction is necessary for the correctness of our construction. In our construction, \( \vec{k}_E(x, \vec{h}) \) and \( \vec{c}_E(y, \vec{h}) \) cancel each other out. Hence, the property implies that there exists an efficient algorithm to make both tuples identical.

Remark 5.2. Linearity of \( \vec{k}_E(x, \vec{h}) \) and \( \vec{c}_E(y, \vec{h}) \) implies that \( \vec{k}_E \) and \( \vec{c}_E \) are linear functions over \( \vec{h} \) when \( x \) and \( y \) are given. Therefore, \( g^{\vec{k}_E(x, \vec{h})} \) and \( g^{\vec{c}_E(y, \vec{h})} \) can be efficiently computed from \( g^\vec{h} \) if \( x \) and \( y \) are given.

An example of tag based encodings. We provide a simple IBE scheme as an instance of our encoding. This encoding results in an adaptively secure IBE scheme via our compiler introduced in the next section.

Let \( \mathcal{X} = \mathcal{Y} := \mathbb{Z}_p \). For all \( ID \in \mathcal{X} \) and \( ID' \in \mathcal{Y} \), \( R(ID, ID') = 1 \) iff \( ID = ID' \).

\[
\bullet \quad \vec{k}_E(ID, (y_u, y_h)) := (y_u ID + y_h) \in \mathbb{Z}_p \quad \text{where} \quad \vec{h} = (y_u, y_h) \in \mathbb{Z}_p^2
\]
For a tag based encoding

5.3.1 The Construction

Waters’ IBE are created randomly. In particular, tags in our construction have structures although tags of

Our compiler is quite similar to those of Waters’ IBE [Wat09]. To be more precise,

Our compiler is quite similar to those of Waters’ IBE [Wat09]. To be more precise,

In particular, tags in our construction have structures although tags of Waters’ IBE are created randomly.

5.3 Our Compiler

Our compiler is quite similar to those of Waters’ IBE [Wat09]. To be more precise,

the main differences between Waters’ IBE and ours are the way of generating Tags in KeyGen and Encrypt; and the type of bilinear maps with which the scheme is equipped. In particular, tags in our construction have structures although tags of Waters’ IBE are created randomly.

5.3.1 The Construction

For a tag based encoding TE for a predicate $R$ where $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, with $\ell$ which is an integer to associated with $R$, $\text{FE}_A(TE)$ is constructed by four algorithms following:

Setup($\lambda$): The algorithm randomly generates three groups $G_1$, $G_2$ and $G_T$ from $\mathcal{G}(\lambda, p)$. Next, it generates $g_1 \in G_1$ and $g_2 \in G_2$ and exponents $\alpha, y_u, y_v, y', y_w, a_1, a_2, b, h_1, \ldots, h_{\ell} \in \mathbb{Z}_p$. Let $\tau_1 = g_1^{y_u+a_1y_v}, \tau'_1 = g_1^{y_u+a_2y_v}$ and $\vec{h} = (h_1, \ldots, h_{\ell})$.

It publishes the public parameters $PK$ as follows

\[
(g_1, g'_1, g^{a_1}_1, g^{a_2}_1, g^{b_1}_1, g^{b_2}_1, \tau_1, \tau'_1, \tau^{b}_1, w_1 = g_1^{y_u}, g'_1, e(g_1, g_2)^{a_1-b})
\]

The MSK consists of $(g_2, g'_2, g^{a_1}_2, g^{b_1}_2, u_1 = g_2^{y_u}, u'_2 = g'_2, w_2 = g_2^{y_v}, \vec{h}).$

Keygen(MSK, PK, $x$): The algorithm chooses randomly $r_1, r_2, z_1, z_2, h'_1, \ldots, h'_{\ell} \in \mathbb{Z}_p$ and sets $r = r_1 + r_2$ and $Tag_k = k\bar{E}(x, \vec{h}')$ where $\vec{h}'$ equals to $(h'_1, \ldots, h'_{\ell})$. 

\[
\bar{E}(ID', (y_u, y_h)) := (y_u ID' + y_h) \in \mathbb{Z}_p
\]

Reconstruction: This is an exact cancellation. Therefore, $m_x = m_y = 1$.

Linearity: For all $\vec{h}' = (y'_u, y'_h)$,

\[
k\bar{E}(ID, (y'_u, y'_h)) + k\bar{E}(ID, (\tilde{y}_u, \tilde{y}_h)) = y'_u ID + \tilde{y}_u ID + y'_h \tilde{y}_h
\]

\[
= k\bar{E}(ID, (y'_u + \tilde{y}_u, y'_h + \tilde{y}_h)).
\]

The linearity of $k\bar{E}(ID', (y'_u, y'_h))$ is identical showed with $k\bar{E}(ID, (y'_u, y'_h))$.

h-hiding: $y_u ID + y_h$ and $y_u ID' + y_h$ are pair-wise independent since $R(ID, ID') = 0$ (i.e. $ID \neq ID'$). Therefore, it is not distinguishable from $y'_u ID' + y'_h$ where $(y'_u, y'_h)$ is randomly selected from $\mathbb{Z}_p^2$. 

Then, it creates $SK$ as follows

$$D_1 = g_2^{a_1}u_2', \quad D_2 = g_2^{-a}v_2'g_2^1, \quad D_3 = (g_2^b)^{-z_1}, \quad D_4 = v_2'g_2^2, \quad D_5 = (g_2^b)^{-z_2},$$

$$D_6 = (g_2^b)^{r_2}, \quad D_7 = g_2^{r_1}, \quad \vec{K} = (g_2^{k\vec{E}(x, \vec{h})})w_2^{\vec{Tag}_k})^{r_1}.$$

It sets $SK = (D_1, ..., D_7, \vec{K}, \vec{Tag}_k)$.

- **Encrypt**(PK, M, y): The algorithm selects $s_1$, $s_2$, $t$, $h'_1$, $..., h'_f \in \mathbb{Z}_p$ and set $s = s_1 + s_2$, and $\vec{Tag}_c = e\vec{E}(y, \vec{h}')$ where $\vec{h}'$ equals to $(h'_1, ..., h'_f)$. It creates the ciphertext, CT as follows

$$C = M\cdot(e(g_1, g_2)^{a_{1-b}^s}e, \quad C_1 = (g_2^b)^s, \quad C_2 = (g_1^{a_1})^s, \quad C_3 = (g_1^{b-a_2})^s, \quad C_4 = (g_1^{b-a_2})^s, \quad C_5 = (g_1^{a_2}), \quad C_6 = r_1^{s_1}t_1^{s_2}, \quad C_7 = (r_1^{s_1}t_1^{s_2})^{s_1}w_1^{-t}, \quad C_8 = g_1^{t_1}, \quad \vec{E} = (g_1^{\vec{E}(y, \vec{h})}w_1^{\vec{Tag}_c})^t.$$

It sets $CT = (C, C_1, ..., C_8, \vec{E}, \vec{Tag}_c)$.

- **Decrypt**(SK, CT, PK): First, the algorithm calculates

$$A_1 = e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5), \quad A_2 = e(C_6, D_6)e(C_7, D_7).$$

Since $R(x, y) = 1$, there exist reconstruction vectors $\vec{m}_x$ and $\vec{m}_y$ such that $\vec{m}_x\vec{k}\vec{E}(x, \vec{h}) = \vec{m}_y\vec{c}\vec{E}(y, \vec{h})$. If $\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c$ is 0, it aborts. Otherwise,

$$A_3 = e(C_8, \vec{K}^{\vec{m}})/e(\vec{E}^{\vec{m}}_y, D_7) = e(g_1, g_2)^{y\cdot r_1(t(\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c)).$$

Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c))}.$

**Correctness.** Calculating $A_1/A_2$ is trivial and can be found in [Wat09]. We only point out that $A_1/A_2 = e(g_1, g_2)^{a_{1-b}^s}e(g_1, w_2)\cdot r_1$. For $\vec{m}_x$ and $\vec{m}_y$ such that $\vec{m}_x\vec{k}\vec{E}(x, \vec{h}) = \vec{m}_y\vec{c}\vec{E}(y, \vec{h})$, the correctness of $A_3$ is calculated as follows

$$A_3 = e(C_8, \vec{K}^{\vec{m}})/e(\vec{E}^{\vec{m}}_y, D_7)$$

$$= e(g_1, (g_2^{k\vec{E}(x, \vec{h})})w_2^{\vec{Tag}_k})^{r_1\cdot\vec{m}})/e((g_1^{\vec{E}(y, \vec{h})})w_1^{\vec{Tag}_c})^{t\cdot\vec{m}_y}, g_2^{r_1})$$

$$= (e(g_1, g_2)^{r_1\cdot t\cdot\vec{m}_x\vec{E}(x, \vec{h})}e(g_1, w_2)^{r_1\cdot t\cdot\vec{m}_y\vec{E}(y, \vec{h})})^{e(g_1, g_2)^{r_1\cdot t\cdot\vec{m}_y\vec{E}(y, \vec{h})}}e(g_1, w_2)^{r_1\cdot t\cdot\vec{m}_y\vec{Tag}_c})$$

Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c))}.$

**Remark 5.3.** Alternatively, to reduce the number of pairing computations, we set $\vec{m}'_x = \vec{m}_x/(\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c)$ and $\vec{m}'_y = \vec{m}_y/(\vec{m}_x\vec{Tag}_k - \vec{m}_y\vec{Tag}_c)$.
Then, the decryption can be done by calculating
\[ A'_1 := e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5)/e(C_6, D_6), \]
\[ A'_2 := e(C_8, \tilde{K}^\alpha)/e(\tilde{E}, D_7). \]
where \( \tilde{E} := (C_7, E^\alpha). \) Finally, \( M \) is retrieved since \( M = C/(A'_1 \cdot A'_2) \).

**Theorem 5.1.** (Informal) Suppose there exists a tag based encoding \( TE \), then our \( FE_A(TE) \) is adaptively secure under the (Asymmetric) DLIN assumption.

**Proof:** This is proved by lemmas 5.1 to 5.3. \( \square \)

### 5.3.2 Security Analysis

**Semi-functional Ciphertext.** By running Encrypt algorithm for a message \( M \) and an input \( y \), the algorithm generates a normal ciphertext
\[ CT = (C', C'_1, ..., C'_8, \tilde{E}', \tilde{Tag}'_c). \]

Then, it randomly selects \( \kappa \in \mathbb{Z}_p \) and sets
\[ C = C', \quad C_1 = C'_1, \quad C_2 = C'_2, \quad C_3 = C'_3, \quad C_4 = C'_4g^\beta_{a^2\kappa}, \]
\[ C_5 = C'_5g^\gamma, \quad C_6 = C'_6v^\alpha_{a^2\kappa}, \quad C_7 = C'_7v^\beta_{a^2\kappa}, \quad C_8 = C'_8, \quad \tilde{E} = \tilde{E}', \quad \tilde{Tag}_c = \tilde{Tag}'_c. \]

**Semi-functional Key.** By running Keygen algorithm for an input \( x \), the algorithm generates a normal key
\[ SK = (D'_1, ..., D'_7, \tilde{K}', \tilde{Tag}'_k). \]

Then, it sets
\[ D_1 = D'_1g_2^{-\alpha_1\alpha_2\gamma}, \quad D_2 = D'_2g_2^{\alpha_2\gamma}, \quad D_3 = D'_3, \quad D_4 = D'_4g^{\alpha_1\gamma} \]
\[ D_5 = D'_5, \quad D_6 = D'_6, \quad D_7 = D'_7, \quad \tilde{K} = \tilde{K}', \quad \tilde{Tag}_k = \tilde{Tag}'_k. \]
It should be noted that \( e(g_1, g_2)^{\alpha_1\alpha_2\beta\gamma} \) will be added to the message to be encrypted if the semi-functional key is used to decrypt the semi-functional ciphertext.

**Security Games**

**Game\textsubscript{real}:** This game is a real game. It is identical with the adaptive security model.
**Game:** This game is identical with \( \text{Game}_{\text{real}} \) except the challenge ciphertext and the first \( i \) keys. In this game, the challenge ciphertext and the first \( i \) keys are semi-functional.

**Game_{final}:** This game is identical with \( \text{Game}_q \) except the challenge ciphertext where \( q_i \) is the total number of key queries. In this game, the challenge ciphertext is still semi-functional, but it is an encryption on a random message.

First, we prove that \( \text{Game}_{\text{real}} \) and \( \text{Game}_0 \) are indistinguishable (semi-functional ciphertext invariance) in lemma 5.1. Then, we show that \( \text{Game}_{k-1} \) is also indistinguishable from \( \text{Game}_k \) (semi-functional key invariance) in lemma 5.2. Finally, in lemma 5.3, we prove the invariance between \( \text{Game}_q \) and \( \text{Game}_{final} \) (semi-functional security).

**Lemma 5.1. (Semi-functional Ciphertext Invariance)** Suppose that there exists an algorithm \( \mathcal{A} \) which distinguishes \( \text{Game}_{\text{real}} \) and \( \text{Game}_0 \) with non-negligible advantage \( \epsilon \). Then, we can build an algorithm \( \mathcal{B} \) which breaks (Asymmetric) DLIN assumption with advantage \( \epsilon \).

**Proof:** Firstly, \( \mathcal{B} \) takes an \( \{g_1, f_1, \nu_1, g_1^{e_1}, f_1^{e_2}, T, g_2, f_2, \nu_2\} \) as an instance from (Asymmetric) DLIN assumption. Depending on the value of \( T \), \( \mathcal{B} \) will simulate \( \text{Game}_{\text{real}} \) or \( \text{Game}_0 \) to take an advantage from \( \mathcal{A} \) which can distinguish both games with non-negligible advantage \( \epsilon \).

**Setup** The algorithm, \( \mathcal{B} \), chooses exponents \( \alpha, b, y_u, y_v, y_w, h_1, \ldots, h_\ell \) randomly from \( \mathbb{Z}_p \). It sets \( u_1 = g_1^{y_u}, v_1 = g_1^{y_v} \) and \( v'_1 = g_1^{y_v'} \). Then, it publishes public parameters as:

\[
\text{PK} := \{g_1, g_1^b, g_1^{a_1} = f_1, g_1^{a_2} = \nu_1, g_1^{b-a_1} = f_1^b, g_1^{b-a_2} = \nu_1^b, w_1 = g_1^{y_w}, g_1^{h_1}, \ldots, g_1^{h_\ell},
\]

\[
\tau_1 = g_1^{y_u} f_1^{w_1}, \tau_1' = g_1^{y_v} \nu_1^{y_v}, \tau_1^b, \tau_1^{b_1}, \tau_1^b g_1^{y_v}, e(g_1, g_2)^{a_1} = e(g_1, f_2)^{a_1} \}.
\]

This implicitly sets \( a_1 = y_f \) and \( a_2 = y_v \) where \( y_f \) and \( y_v \) are the discrete logarithms of \( \nu_1 \) and \( f_1 \) to the base \( g_1 \) modulo \( p \), respectively. It sets \( \text{MSK} := \{g_2, g_2^a, g_2^{a_1} = f_2^a, g_2^{a_2} = g_2^{a_1}, \nu_2 = g_2^{y_v}, h_1, \ldots, g_2^{h_\ell} \} \).

**Phase I and II** Because \( \mathcal{B} \) knows the public parameters and the master secret key, it can generate normal keys using **Keygen**.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for \( y^* \) with messages \( M_0, M_1 \), \( \mathcal{B} \) randomly selects \( \beta \) from \( \{0, 1\} \). \( \mathcal{B} \) runs the encryption algorithm to generate a normal ciphertext \( CT' = (C', C'_1, \ldots, C'_8, \vec{E'}, \vec{T} \vec{a} \vec{g}_1') \) for \( y^* \) and \( M_\beta \). We denote the random exponents of \( CT' \) as \( s'_1, s'_2, t' \). Then it sets the challenge
ciphertext as:

\[ C = C' \cdot (e(g_1^{c_1}, f_2) \cdot e(g_1^{c_2}, f_2^{c_2}))^{b_\alpha}, \quad C_1 = C'_1 \cdot (g_1^{c_1})^b, \quad C_2 = C'_2 \cdot (f_1^{c_2})^{-b}, \quad C_3 = C'_3 f_1^{c_2}, \]

\[ C_4 = C'_4(T)^b, \quad C_5 = C'_5 \cdot T, \quad C_6 = C'_6 \cdot (g_1^{c_1})^{y_\nu} \cdot (f_1^{c_2})^{-y_\nu} \cdot T^{y_\nu'}, \]

\[ C_7 = C'_7 \cdot ((g_1^{c_1})^{y_\nu} \cdot (f_1^{c_2})^{-y_\nu} \cdot T^{y_\nu'})^b, \quad C_8 = C'_8, \quad \vec{E} = \vec{E}', \quad \vec{Tag}_c = \vec{Tag}_c'. \]

This implicitly sets \( s_1 = -c_2 + s_1', s_2 = s_2' + c_1 + c_2 \) and \( s = s_1 + s_2 = c_1 + s_1' + s_2' \).

If \( T = \nu_1^{c_1+c_2} \), the challenge ciphertext is a normal ciphertext, and \( Game_{\text{real}} \) has been simulated properly. Otherwise, if \( T \) is a random, and is denoted as \( \nu_1^{c_1+c_2} \nu_1^{c_1} \), this is a properly distributed semi-functional key, and \( Game_0 \) has been simulated properly.

We present the semi-functional key invariance of our construction similar to that of Waters’ IBE, but using encodings. In Waters’ IBE, tags are used to hide the type of the challenge key (i.e. the \( k^{th} \) key in lemma 5.2). If the simulator try to distinguish the type of this key by generating a valid semi-functional ciphertext to be decrypted only if the key is normal, tags hinder the simulator’s trial. To do this, tags in a ciphertext and a private key are correlated but this correlation is information theoretically hidden to the adversary. It means that tags remain randomly distributed to the adversary. Our framework basically follows this strategy. However, in our framework, tags have structures. Tags are constructed by the encodings \( \vec{kE} \) and \( \vec{cE} \) and take random values instead of public parameters. To prove semi-functional key invariance, the random values as inputs of tags in both the challenge key and the challenge ciphertext must be identical because those tags must restrict the simulator’s ability. At the same time, sharing random values also must be hidden to the adversary to hide the correlation caused by sharing random values.

In lemma 5.2, we will show that these requirements are satisfied by \( \vec{h}\)-hiding and linearity properties.

**Lemma 5.2. (Semi-functional Key Invariance)** Suppose that there exists an algorithm \( \mathcal{A} \) which distinguishes \( Game_{k-1} \) and \( Game_k \) with a non-negligible advantage \( \epsilon \). Then, we can build an algorithm \( \mathcal{B} \) which breaks (Asymmetric) DLIN assumption with \( \epsilon \).

**Proof:** \( G_1 \) and \( G_2 \) of (Asymmetric) DLIN are reversed. Therefore, \( \mathcal{B} \) takes an \( \{g_1, f_1, \nu_1, g_2, f_2, \nu_2, g_2^{c_1}, f_2^{c_2}, T\} \) as an instance from (Asymmetric) DLIN assumption. Depending on the value of \( T \), \( \mathcal{B} \) will simulate \( Game_{k-1} \) or \( Game_k \) to take an advantage from \( \mathcal{A} \) which can distinguish. It should be noted that \( T \) is in \( G_2 \) in the reversed assumption.
Setup Algorithm $B$ chooses exponents $\alpha, a_1, a_2, y_v, y'_v, y_w, h'_1, \ldots h'_\ell, \bar{h}_1, \ldots \bar{h}_\ell$, randomly from $\mathbb{Z}_p$. Then, it sets

$$g_1 = g_1, \quad g_1^b = f_1, \quad g_1^{b-a_1} = f_1^{a_1}, \quad g_1^{b-a_2} = f_1^{a_2}, \quad u_1 = \nu_1^{-a_1a_2},$$

$$v_1 = v_1^{a_2} \cdot g_1^{y_v}, \quad v'_1 = v_1^{a_1} \cdot g_1^{y'_v}, \quad \tau_1 = uv^{a_1} = g_1^{y_v}, \quad \tau'_1 = u_1 v_1^{a_2} = g_1^{y'_v},$$

$$\tau_1^b = (u_1 v_1)^b = f_1^{b-a_1}, \quad \tau'_1^b = (u_1 v_1^{a_2})^b = f_1^{a_2}, \quad e(g_1, g_2)^{a_1b} = e(f_1, g_2)^{a_1}.$$

Finally, it sets $w_1 = f_1 g_1^{y_w}, \{g_1^h = f_1^{-h'_1} \bar{g}_1^h, \forall i \in [1, \ell]\}$. It publishes the public parameters following

$$(g_1, g_1^b, g_1^{a_2}, g_1^{b-a_1}, g_1^{b-a_2}, \tau_1, \tau'_1, \tau_1^b, \tau'_1^b, w_1, g_1^h, e(g_1, g_2)^{a_1b})$$

where $\bar{h} = (h_1, \ldots, h_\ell)$. $B$ sets $\text{MSK} = \{g_2, g_2^b, g_2^{a_2}, g_2^{b-a_1}, g_2^{b-a_2} = f_2, u_2 = \nu_2^{-a_1a_2}, v_2 = v_2^{a_2} g_2^{y_v}, v_2' = v_2^{a_1} g_2^{y'_v}, w_2 = f_2 g_2^{y_w}, \{g_2^{h_i} = f_2^{-h'_1} \bar{g}_2^{h_i}, \forall i \in [1, \ell]\}\}.$

In the setting, $\bar{h}$ is implicitly set by $\bar{h}_i = (h'_1, \ldots, h'_\ell)$ and $\tilde{h}_i = (\hat{h}_1, \ldots, \hat{h}_\ell)$ if we write $f_1 = g_1^{y_v}$. $B$ calculates $g_1^\tilde{h}$ because it knows $g_1, f_1, \bar{h}_i, \tilde{h}_i$. It should be noted that the values of $\{h'_i; \forall i \in [1, \ell]\}$ are not revealed. It means that they are initially information theoretically hidden because, for all $i \in [1, \ell]$, $\hat{h}$ is uniquely added where $h'_i$ appears.

Phase I and II For the first $k-1$ semi-functional keys, $B$ generates a normal key and selects $\gamma$ randomly from $\mathbb{Z}_p$. It then adds semi-functional parts to the normal key. This is possible because $B$ knows $a_1, a_2$ and $\text{MSK}$. Similarly, for the rest keys except $k$th key ($i > k$), $B$ can generate normal keys using the key generation algorithm, KeyGen, for the same reason.

For the $k$th key, $B$ sets $T\bar{a}g_k = k\bar{E}(x, \tilde{h})$. Then, with $T\bar{a}g_k$, it generates a normal key $SK' = (D'_1, \ldots, D'_\ell, \tilde{K'}, T\bar{a}g_k)$ using the key generation algorithm. Then, it sets the $k$th key as follows

$$D_1 = D'_1 \nu_1^{-a_2}, \quad D_2 = D'_2 \nu_2^{-a_1a_2}, \quad D_3 = D'_3 (f_2^{c_2})^{y_v}, \quad D_4 = D'_4 (g_2^{c_1})^{y'_v},$$

$$D_5 = D'_5 (f_2)^{y'_v}, \quad D_6 = D'_6 f_2^{c_2}, \quad D_7 = D'_7 (g_2^{c_1}), \quad \tilde{K} = \tilde{K}' (g_2^{c_1})^{k\bar{E}(x, \tilde{h} + y_v, \tilde{h})}, T\bar{a}g_k.$$

We let $r'_1, r'_2, z_1', z_2'$ denote the random exponents of $SK'$. Then, it implicitly sets $z_1 = z'_1 - y_v c_2$ and $z_2 = z'_2 - y'_v c_2$. Also, by linearity property,

$$g_2^k \bar{E}(x, \tilde{h}) = g_2^{k\bar{E}(x, -y_v \tilde{h})} = g_2^{k\bar{E}(x, -y_f \bar{h})}, \quad g_2^k \bar{E}(x, \hat{h}) = f_2^{-k\bar{E}(x, \tilde{h})} g_2^{k\bar{E}(x, \hat{h})}$$

Therefore, the value of $\tilde{K}'$ can be represented as follows:

$$\tilde{K}' = (f_2^{-k\bar{E}(x, \tilde{h})} g_2^{k\bar{E}(x, \hat{h})})^r_1 = (g_2^{k\bar{E}(x, \hat{h} + y_v, \tilde{h})})^r_1.$$
This implies that $\bar{K} = \bar{K'} (g_2^{c_1})^{\text{a}E(x,\hat{h}+yw,\hat{h'})} = (g_2^{\text{a}E(x,\hat{h}+yw,\hat{h'})})^{r_1+c_1}$.

If $T$ equals to $\nu_2^{c_1+c_2}$, then the $k^{th}$ key is a normal key with $r_1 = r_1' + c_1$ and $r_2 = r_2' + c_2$. Otherwise, if $T$ is $\nu_2^{c_1+c_2} g_2^3$, which means a random group element, then the $k^{th}$ key is a properly distributed semi-functional key.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for $y^*$ with messages $M_0, M_1$, $B$ randomly selects $\beta$ from $\{0,1\}$. With $\bar{T}ag_c = c\text{a}E(y^*,\hat{h'})$, $B$ runs the encryption algorithm to generate a normal ciphertext $CT' = (C', \bar{C}_1', ... , \bar{C}_\ell', \vec{E}, \bar{T}ag_c)$ for $y^*$ and $M_\beta$. We let $s_1', s_2', t'$ denote the random exponents of $CT'$. To make the semi-functional challenge ciphertext, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets $C = C', \bar{C}_1 = \bar{C}_1', \bar{C}_2 = \bar{C}_2', \bar{C}_3 = \bar{C}_3'$. Additionally, it sets

$$C_4 = C_4' f_1^{a_2, \kappa}, \quad C_5 = C_5' \cdot g_1^{a_2, \kappa}, \quad C_6 = C_6' \cdot v_1^{a_2, \kappa}, \quad C_7 = C_7' \cdot f_1^{\kappa \cdot a_2 \cdot \nu_1 a_1, \kappa \cdot yw - a_2}$$

$$C_8 = g_1' \cdot v_1^{a_2, \kappa}, \quad \vec{E} = \vec{E}' \cdot (\nu_1^{\text{a}E(y^*,\hat{h}+yw,\hat{h'}) \cdot a_1, \kappa \cdot yw - a_2})^{t'}$$

This implicitly sets $g_1' = g_1' \cdot v_1^{a_2, \kappa}$. It should be noted that $\nu_1^{a_2 \cdot b \cdot c}$ of $v_1^{a_2 \cdot b \cdot c}$ is cancelled out by $w_1^{-t}$ in $C_7$. The fact that $\bar{T}ag_c$ and $\bar{T}ag_k$ share the same vector $\hat{h}'$ is hidden to the adversary by $h$-hiding property since $R(x, y^*) = 0$. Also, $\vec{E}$ is valid since

$$\vec{E}' = (f_1^{\text{a}E(y^*,\hat{h'})} g_1^{\text{a}E(y^*,\hat{h'})} (f_1 g_1 yw) e^{\text{E}(y^*,\hat{h'})})^{t'} = (g_1^{\text{a}E(y^*,\hat{h}+yw,\hat{h'})})^{t'}.$$

The second equality of the above equation holds by linearity property.

$B$ cannot test whether the $k^{th}$ key is normal or semi-functional by creating a ciphertext which can be decrypted only by a normal key because $\bar{T}ag_k$ and $\bar{T}ag_c$ share $\hat{h}'$. It means that $\vec{m}_e \bar{T}ag_k - \vec{m}_w \bar{T}ag_c$ equals to 0 if the simulator creates a semi-functional ciphertext such that $R(x, y) = 1$. Hence, the decryption algorithm will abort. \hfill \Box

**Lemma 5.3. (Semi-functional Security)** Suppose that there exists an algorithm $A$ which distinguishes $\text{Game}_b$ and $\text{Game}_{final}$ with non-negligible advantage $\epsilon$. Then, we can build an algorithm $B$ which breaks (Asymmetric) DBDH assumption with advantage $\epsilon$.

**Proof:** $B$ takes an $\{g_1, g_1^c, g_1^C, T, g_2, g_2^c, g_2^C\}$ as an instance from (Asymmetric) DBDH assumption. Depending on the value of $T$, $B$ will simulate $\text{Game}_b$ or $\text{Game}_{final}$ to take an advantage from $A$ which can distinguish both games.

**Setup** $B$ chooses exponents $b, a_1, y_w, y_v, y'_v, y_w, h_1, ... h_\ell$, randomly from $\mathbb{Z}_p$, and sets

$$g_1, g_1^b, g_1^a, g_1^{a_2} = g_2, g_1^b a_1, g_1^{b a_2} = (g_1^c)^b, w_1 = g_1^{y_w}, \{g_1^{h_i}; \forall i \in [1, \ell]\}$$

$$\tau_1 = g_1^{y_w+y_w a_1}, \tau_1' = g_1^b (g_1^c)^{y_v}, \tau_1 = g_1^b, \tau_1 = e(g_1, g_2)^{\alpha a_1 b} = e(g_1^c, g_2^c)^{b a_1}.$$
It implicitly sets $u_1 = g^{\alpha}, v_1 = g^{\beta}, v'_1 = g^{\gamma'}, \alpha = c_1 \cdot c_2, a_2 = c_2$. It should be noted that MSK cannot be explicitly calculated because $B$ does not know $g_1^\alpha$.

**Phase I and II** For semi-functional keys, if the adversary requests a private key for $x$, it randomly generates $r_1, r_2, z_1, z_2, \gamma', h'_1, ..., h'_\ell$ from $\mathbb{Z}_p$ and sets $Tag_k = k \tilde{E}(x, \vec{h}')$ where $\vec{h}' = (h'_1, ..., h'_\ell)$. It then, creates semi-functional key as follows:

$$D_1 = (g_2^{c_2})^{-\gamma' \cdot a_1} \cdot u_2, \quad D_2 = (g_2^{c_2})^{\gamma' \cdot v_2 \cdot z_1}, \quad D_3 = (g_2^{c_2})^{-z_1}, \quad D_4 = (g_2^{c_2})^{a_1 \cdot \gamma' \cdot v' \cdot z_2},$$

$$D_5 = (g_2^b)^{-z_2}, \quad D_6 = g_2^{r_2 \cdot b}, \quad D_7 = g_2^{r_1}, \quad \vec{K} = (g_2^{k \tilde{E}(x, \vec{h})} \cdot w_2^{Tag_k})^r_1, \quad Tag_k.$$

where $\vec{h} = (h_1, ..., h_\ell)$. This implicitly sets $\gamma = c_1 + \gamma'$ and $r = r_1 + r_2$.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for $y^*$ with messages $M_0, M_1, B$ randomly selects $\beta \in \{0, 1\}$. Then it generates random values $\kappa', s_1, t, h''_1, ..., h''_\ell$ from $\mathbb{Z}_p$ and sets $Tag_c = c \tilde{E}(y^*, \vec{h}'')$ where $\vec{h}'' = (h''_1, ..., h''_\ell)$. It then, creates the challenge ciphertext as follows:

$$C = M_0 T^{a_1 \cdot b}, \quad C_1 = g_1^{s_1 \cdot b} \cdot (g_1^{c_3})^b, \quad C_2 = g_1^{b \cdot a_1 \cdot s_1}, \quad C_3 = g_1^{a_1 \cdot s_1}, \quad C_4 = (g_1^{c_2})^{\kappa' \cdot b},$$

$$C_5 = (g_1^{c_2})^{\kappa'}, \quad C_6 = \tau_1^{s_1} (g_1^{c_3})^{y_1} (g_1^{c_2})^{y_2 \cdot \kappa'}, \quad C_7 = (\tau_1^{s_1} (g_1^{c_3})^{y_1} (g_1^{c_2})^{y_2 \cdot \kappa'})^w_1, \quad \tau_2 = g_1^{\tau_1} \cdot \vec{E} = (g_1^{c \tilde{E}(y^*, \vec{h}'')} \cdot w_1^{Tag_c}) \cdot Tag_c.$$

This implicitly sets $s_2 = c_3$ and $\kappa = \kappa' - c_3$.

If $T = c(g_1, g_2)^{c_1 \cdot c_2 \cdot c_3}$, this has properly simulated $Game_{q_1}$. Otherwise, if $T$ is random, a random value is added into $M_\beta$. Hence, it has properly simulated $Game_{final}$.

\[\square\]

### 5.4 Our Compiler with Symmetric Bilinear Maps

#### 5.4.1 The Construction

For a tag based encoding TE for a predicate $R$ where $R : \mathbb{X} \times \mathbb{Y} \rightarrow \{0, 1\}$, with $\ell$ which is an integer to associated with $R$, $FE(TE)$ is constructed by four algorithms following:

- **Setup($\lambda$):** The algorithm randomly generates two groups $G$ and $G_T$ from $\mathcal{G}(\lambda, p)$.
  
  Next, it generates $g, v, v_1, v_2, w \in G$ and exponents $\alpha, a_1, a_2, b, h_1, ..., h_\ell \in \mathbb{Z}_p$.
  
  Let $\tau_1 = vv_1^{a_1}, \tau_2 = vv_2^{a_2}$ and $\vec{h} = (h_1, ..., h_\ell)$. It publishes the public parameters $PK$ as follows:

$$\langle g, g^{b}, g^{a_1}, g^{a_2}, g^{b \cdot a_1}, g^{b \cdot a_2}, r_1, r_2, \tau_1, \tau_2, w, g^{\vec{h}}, c(g, g)^{\alpha \cdot a_1 \cdot b} \rangle.$$
The MSK consists of $(g^a, g^{a_1}, v, v_1, v_2)$.

- **Keygen** (MSK, PK, x): The algorithm chooses randomly $r_1, r_2, z_1, z_2, h'_1, ..., h'_t \in \mathbb{Z}_p$ and sets $r = r_1 + r_2$ and $T \hat{a}g_k = k \hat{E}(x, \tilde{h}')$ where $\tilde{h}'$ equals to $(h'_1, ..., h'_t)$. Then, it creates $SK$ as follows

$$D_1 = g^{a_1}v^r, D_2 = g^{-a}v_1^rz_1, D_3 = (g^b)^{z_1}, D_4 = v_2^r, D_5 = (g^b)^{z_2},$$

$$D_6 = g^{r_2b}, D_7 = g^{r_1}, K = (g^{k \hat{E}(x, \tilde{h})}wT \hat{a}g_k)^{r_1}.$$ 

It sets $SK = (D_1, ..., D_7, K, T \hat{a}g_k)$.

- **Encrypt** (PK, M, y): The algorithm selects $s_1, s_2, t, h''_1, ..., h''_t \in \mathbb{Z}_p$ and set $s = s_1 + s_2$, and $T \hat{a}g_c = c \hat{E}(y, \tilde{h}'')$ where $\tilde{h}''$ equals to $(h''_1, ..., h''_t)$. It creates the ciphertext, CT as follows

$$C = M \cdot (e(g, g)^{a_1-b})^{s_2}, C_1 = (g^b)^s, C_2 = (g^{b-a_1})^{s_1}, C_3 = (g^{a_1})^{s_1}, C_4 = (g^{b-a_2})^{s_2},$$

$$C_5 = (g^{a_2})^{s_2}, C_6 = \tau_1^{s_1} \tau_2^{s_2}, C_7 = (\tau_1^{s_1})^{s_1} (\tau_2^{s_2})^{s_2} w^{-t}, C_8 = g^t, \hat{E} = (g^{c \hat{E}(x, \tilde{h})}wT \hat{a}g_c)^t.$$ 

It sets $CT = (C, C_1, ..., C_8, \hat{E}, T \hat{a}g_c)$.

- **Decrypt** (SK, CT, PK): First, the algorithm calculates

$$A_1 = e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5), A_2 = e(C_6, D_6)e(C_7, D_7).$$

Since $R(x, y) = 1$, there exist reconstruction vectors $\tilde{m}_x$ and $\tilde{m}_y$ such that $\tilde{m}_xk \hat{E}(x, \tilde{h}) = \tilde{m}_y \hat{E}(y, \tilde{h})$. If $\tilde{m}_xT \hat{a}g_k - \tilde{m}_yT \hat{a}g_c$ is 0, it aborts. Otherwise,

$$A_3 = e(C_8, K^{\tilde{m}_x})/e(\hat{E}^{\tilde{m}_y}, D_7) = e(g, w)^{\tau_1 t (\tilde{m}_xT \hat{a}g_k - \tilde{m}_yT \hat{a}g_c)}.$$ 

Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\tilde{m}_xT \hat{a}g_k - \tilde{m}_yT \hat{a}g_c)}).

The correctness of our construction is almost identical with that of the construction with asymmetric pairing in the previous section. It should be noted that the number of pairing computations of this construction also can be reduced as the construction with asymmetric pairing does.

**Theorem 5.2.** (Informal) Suppose there exists Tag based Encoding TE, then our FE(TE) is adaptively secure under DLIN assumption.

**Proof:** This is proved by lemmas 5.4, 5.5 and 5.6. □
5.4.2 Security Analysis

Semi-functional Ciphertext. By running Encrypt algorithm for a message $M$ and an input $y$, the algorithm generates a normal ciphertext $CT = (C', C_1', ..., C_8', \vec{E}', \vec{Tag}_c')$.

Then, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets

$$
\begin{align*}
C &= C', \\
C_1 &= C_1', \\
C_2 &= C_2', \\
C_3 &= C_3', \\
C_4 &= C_4'g^{\alpha_2\gamma}, \\
C_5 &= C_5'g^{\alpha_2\gamma}, \\
C_6 &= C_6'g^{\alpha_2\gamma}, \\
C_7 &= C_7'g^{\alpha_2\gamma}, \\
C_8 &= C_8', \\
\vec{E} &= \vec{E}', \\
\vec{Tag}_c &= \vec{Tag}_c'.
\end{align*}
$$

Semi-functional Key. By running Keygen algorithm for an input $x$, the algorithm generates a normal key $SK = (D_1', ..., D_7', \vec{K}', \vec{Tag}_k')$.

Then, it sets

$$
\begin{align*}
D_1 &= D_1'g^{-\alpha_1\alpha_2\gamma}, \\
D_2 &= D_2'g^{\alpha_2\gamma}, \\
D_3 &= D_3', \\
D_4 &= D_4'g^{\alpha_1\gamma}, \\
D_5 &= D_5', \\
D_6 &= D_6', \\
D_7 &= D_7', \\
\vec{K} &= \vec{K}', \\
\vec{Tag}_k &= \vec{Tag}_k'.
\end{align*}
$$

It should be noted that $e(g, g)^{\alpha_1\alpha_2\beta_2\gamma}$ will be added to the message to be encrypted if the semi-functional key is used for decrypting the semi-functional ciphertext.

Security Games

The definitions of security games and the strategy to prove adaptive security are identical with those of the compiler in symmetric paring.

Lemma 5.4. (Semi-functional Ciphertext Invariance) Suppose that there exists an algorithm $A$ which distinguishes $\text{Game}_{\text{real}}$ and $\text{Game}_0$ with non-negligible advantage $\epsilon$. Then, we can build an algorithm $B$ which breaks DLIN assumption with advantage $\epsilon$.

Proof: Firstly, $B$ takes an $\{g, f, \nu, g^{\alpha_1}, f^{\alpha_2}, T\}$ as an instance from DLIN assumption. Depending on the value of $T$, $B$ will simulate $\text{Game}_{\text{real}}$ or $\text{Game}_0$ to take an advantage from $A$ which can distinguish both games with non-negligible advantage $\epsilon$.

Setup Algorithm $B$ chooses exponents $\alpha, b, y_v, y_{v_1}, y_{v_2}, y_w, h_1, ..., h_\ell$ randomly from $\mathbb{Z}_p$ and group elements $w, g$. Then, it publishes public parameters as:

$$
\begin{align*}
\text{PK} := \{g, b, g^{\alpha_1} = f, g^{\alpha_2} = \nu, g^{b\alpha_1} = f^b, g^{b\alpha_2} = \nu^b, w = g^{y_w}, g^{h_1}, ..., g^{h_\ell}, \\
\tau_1 = g^{y_e}f^{y_{v_1}}, \tau_2 = g^{y_w}\nu^{y_{v_2}}, \tau_1^b, \tau_2^b, e(g, g)^{\alpha_1\alpha_2} = e(g, f)^{\alpha b}\}
\end{align*}
$$
it implies that \( v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}} \). It sets \( \text{MSK} := \{ g^\alpha, g^{\alpha-a_1} = f^\alpha \} \).

**Phase I and II** Because \( B \) knows the public parameters and the master secret key, it can generate normal keys using Keygen.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for \( y^* \) with messages \( M_0, M_1, B \) randomly selects \( \beta \) from \( \{0, 1\} \). \( B \) runs the encryption algorithm to generate a normal ciphertext \( CT' = (C', C'_1, ..., C'_8, \bar{E}', \bar{T}_c, \bar{T}_s) \) for \( y^* \) and \( M_3 \). We denote the random exponents of \( CT' \) as \( s_1', s_2', t' \). Then it sets the challenge CT as:

\[
C = C' \cdot (e(g^{c_1}, f) \cdot e(g, f^{c_2}))^{b^\alpha},
\]

\[
C_1 = C'_1 \cdot (g^{c_1})^{b},
\]

\[
C_2 = C'_2 \cdot (f^{c_2})^{-b},
\]

\[
C_3 = C'_3 f^{c_2},
\]

\[
C_4 = C'_4 \cdot (T)^{b},
\]

\[
C_5 = C'_5 \cdot T,
\]

\[
C_6 = C'_6 \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}},
\]

\[
C_7 = C'_7 \cdot ((g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}})^b,
\]

\[
C_8 = C'_8 \cdot \bar{E} = \bar{E}',
\]

\[
\bar{T}_c = \bar{T}_s.
\]

This implicitly sets \( s_1 = -c_2 + s_1', s_2 = s_2' + c_1 + c_2 \) and \( s = s_1 + s_2 = c_1 + s_1' + s_2' \). If \( T = \nu^{c_1+c_2} \), the challenge ciphertext is a normal ciphertext, and Game\(_{real} \) has been simulated properly. Otherwise, if \( T \) is a random, and is denoted as \( \nu^{c_1+c_2} \nu^\epsilon \), this is a properly distributed semi-functional key, and Game\(_{0} \) has been simulated properly.

\[\square\]

**Lemma 5.5. (Semi-functional Key Invariance)** Suppose that there exists an algorithm \( A \) which distinguishes Game\(_{k-1} \) and Game\(_{k} \) with a non-negligible advantage \( \epsilon \). Then, we can build an algorithm \( B \) which breaks DLIN assumption with \( \epsilon \).

**Proof:** Firstly, \( B \) takes an \( \{ g, f, \nu, g^{c_1}, f^{c_2}, T \} \) as an instance from DLIN assumption. Depending on the value of \( T \), \( B \) will simulate Game\(_{k-1} \) or Game\(_{k} \) to take an advantage from \( A \) which can distinguish between both games with non-negligible advantage.

**Setup** Algorithm \( B \) chooses exponents \( \alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_{w}, h_1', ..., h_\ell', \bar{h}_1, ..., \bar{h}_\ell, \) randomly from \( \mathbb{Z}_p \). Then, it sets

\[
g = g, g^b = f, g^{b-a_1} = f^{a_1}, g^{b-a_2} = f^{a_2}, \nu = \nu^{-a_1a_2}, v_1 = \nu^{a_2} \cdot g^{y_{v_1}}, v_2 = \nu^{a_1} \cdot g^{y_{v_2}},
\]

\[
\tau_1 = v_{v_1}^{a_1} = g^{y_{v_1}a_1}, \tau_2 = v_{v_1}^{a_2} = g^{y_{v_2}a_2}, \tau_1^b = f^{y_{v_1}a_1}, \tau_2^b = f^{y_{v_2}a_2},
\]

\[
w = fg^{y_{w}}, g^{h_i} = f^{-h_i}g^{h_{i}}, \forall i \in [1, \ell], e(g, g)^{\alpha-a_1b} = e(f, g)^{\alpha-a_1}.
\]

It publishes the public parameters following

\[
(g, g^b, g^{a_1}, g^{a_2}, g^{b-a_1}, g^{b-a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, g^{\bar{h}}, e(g, g)^{\alpha-a_1b})
\]
where \( \vec{h} = (h_1, ..., h_\ell) \). In the setting, \( \vec{h} \) is implicitly set by \( \tilde{\vec{h}} - y_f \vec{h}' \) where \( \vec{h}' = (h'_1, ..., h'_\ell) \) and \( \vec{h} = (\tilde{h}_1, ..., \tilde{h}_\ell) \) if we write \( f = g^{y_f} \). Because \( B \) does not know \( y_f \), it cannot calculate the values of \( \vec{h} \), but it calculates \( g^\vec{h} \) because it knows \( g, f, \vec{h}, \vec{h}' \).

It should be noted that the values of \( \{h'_i; \forall i \in [1, \ell]\} \) are not revealed initially. It means that they are initially information theoretically hidden because, for all \( i, \tilde{h}_i \) is uniquely added where \( h_i' \) appears. \( B \) knows all MSK = \( (g^{a_i}, g^{\gamma a_1}, v, v_1, v_2) \) since it knows \( \alpha \) and \( a_1 \).

**Phase I and II** For the first \( k - 1 \) semi-functional keys, \( B \) generates a normal key and selects \( \gamma \) randomly from \( \mathbb{Z}_p \). It then adds semi-functional parts to the normal key. This is possible because \( B \) knows all public parameters and MSK. Similarly, for the rest keys except \( k^{\text{th}} \) key \((i > k)\), \( B \) can generate normal keys using the key generation algorithm, \textbf{KeyGen}, as the same reasons.

For the \( k^{\text{th}} \) key, \( B \) sets \( T\vec{a}g_{k} = \vec{k}E(x, \vec{h}') \). Then, with \( T\vec{a}g_{k} \), it generates a normal key \( SK' = (D'_1, ..., D'_\ell, \vec{K}', T\vec{a}g_{k}) \) using the key generation algorithm. Then, it sets the \( k^{\text{th}} \) key as follows:

\[
D_1 = D'_1 T^{-a_1 c_2}, D_2 = D'_2 T^{a_2} (g^{c_1})^{y_1}, D_3 = D'_3 (f^{c_2})^{y_1}, D_4 = D'_4 T^{a_1} (g^{c_1})^{y_2},
\]

\[
D_5 = D'_5 (f^{c_2})^{y_2}, D_6 = D'_6 f^{c_2}, D_7 = D'_7 (g^{c_1}), \vec{K} = \vec{K}' (g^{c_1})^{\vec{k}E(x, \vec{h}+y_w \vec{r})}, T\vec{a}g_{k}.
\]

We let \( r'_1, r'_2, z'_1, z'_2 \) denote the random exponents of \( SK' \). Then, it implicitly sets \( z_1 = z'_1 - y_{v_1} c_2 \) and \( z_2 = z'_2 - y_{v_2} c_2 \). Also, by linearity property,

\[
g^{\vec{k}E(x, \vec{h})} = g^{\vec{k}E(x, -y_f \vec{h}' + \vec{h})} = g^{\vec{k}E(x, -y_f \vec{h}'')} g^{\vec{k}E(x, \vec{h})} = f^{-\vec{k}E(x, \vec{h}'')} g^{\vec{k}E(x, \vec{h})}
\]

Therefore, the value of \( \vec{K}' \) can be represented as follows:

\[
\vec{K}' = (f^{-\vec{k}E(x, \vec{h}'')})^{g^{\vec{k}E(x, \vec{h}')}} (f g^{y_w})^{\vec{k}E(x, \vec{h}'')} g^{\vec{k}E(x, \vec{h})} = (g^{\vec{k}E(x, \vec{h}+y_w \vec{r}')})^{r'_1 + c_1}.
\]

This implies that \( \vec{K} = \vec{K}' (g^{c_1})^{\vec{k}E(x, \vec{h}+y_w \vec{r}')} = (g^{\vec{k}E(x, \vec{h}+y_w \vec{r}')} r'_1 + c_1) \).

If \( T \) equals to \( \nu^{c_1 + c_2} \), then the \( k^{\text{th}} \) key is a normal key with \( r_1 = r'_1 + c_1 \) and \( r_2 = r'_2 + c_2 \). Otherwise, if \( T \) is \( \nu^{c_1 + c_2} g^{1} \), which means a random group element, then, the \( k^{\text{th}} \) key is a properly distributed semi-functional key.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for \( y^* \) with messages \( M_0, M_1 \), \( B \) randomly selects \( \beta \) from \( \{0, 1\} \). With \( T\vec{a}g_{\ell} = e\vec{E}(y^*, \vec{h}') \), \( B \) runs the encryption algorithm to generate a normal ciphertext \( CT' = (C', C'_1, ..., C'_8, \vec{E}', T\vec{a}g_{\ell}) \) for \( y^* \) and \( M_\beta \). We let \( s'_1, s'_2, t' \) denote the random exponents of \( CT' \). To make the semi-functional challenge ciphertext, it randomly selects.
\[ \kappa \in \mathbb{Z}_p \] and sets \( C = C', C_1 = C'_1, C_2 = C'_2, C_3 = C'_3. \) Additionally, it sets
\[ C_4 = C'_4 f^a_{\kappa}, \quad C_5 = C'_5 \cdot g^a_{\kappa}, \quad C_6 = C'_6 \cdot \gamma^2_{\kappa}, \quad C_7 = C'_7 \cdot f^{v_{2-\kappa}} a_{2} v^{-a_{1} - \gamma_{w}} a_{2} \]
\[ C_8 = g^\ell \cdot \nu^{a_{2} \kappa}, \quad \bar{E} = \bar{E}^\nu \cdot (\nu^C y_{r} + y_{\kappa}) a_{1} a_{2} \kappa, \quad T_{\kappa} \bar{a} g_{c} \]

This implicitly sets \( g^1 = g^\nu \cdot \nu^{a_{1} \kappa}. \) It should be noted that \( \nu^{a_{1} \kappa} \) of \( v_{2} \) is cancelled out by \( w^{-t} \) in \( C_7. \) The fact that \( T_{\kappa} \bar{a} g_{c} \) in the challenge ciphertext and \( T_{\kappa} \bar{a} g_{c} \) in the \( k^{th} \) key share the same vector \( \bar{h} \) is hidden to the adversary by \( h\)-hiding property since \( R(x, y^s) = 0. \) Also, \( \bar{E} \) is a valid ciphertext elements since
\[ \bar{E}^\nu = (f^{C y_{r}} h_{r}) g^{E y_{r}} (f^{g_{2} y_{r}} c E y_{r}) \bar{a}^\nu = (g^{E y_{r}} h_{r}) \bar{a}^\nu. \]

The second equality of the above equation holds by linearity property.

In the simulation, \( B \) cannot test whether the \( k^{th} \) key is normal or semi-functional by generating a ciphertext which can be decrypted only by a normal key because \( T_{\kappa} \bar{a} g_{c} \) and \( T_{\kappa} \bar{a} g_{c} \) must share \( \bar{h} \). In our simulation, \( \bar{m}_s T_{\kappa} \bar{a} g_{k} - \bar{m}_w T_{\kappa} \bar{a} g_{c} = 0 \) if the simulator generates a valid semi-functional ciphertext such that \( R(x, y) = 1. \) Hence, the decryption algorithm will abort.

\[ \square \]

**Lemma 5.6. (Semi-functional Security)** Suppose that there exists an algorithm \( \mathcal{A} \) which distinguishes \( \text{Game}_{n} \) and \( \text{Game}_{\text{final}} \) with non-negligible advantage \( \epsilon. \) Then, we can build an algorithm \( \mathcal{B} \) which breaks DBDH assumption with advantage \( \epsilon. \)

**Proof:** First, \( \mathcal{B} \) takes an \( \{g, g^{a_1}, g^{a_2}, g^{a_3}, T\} \) as an instance from DLIN assumption. Depending on the value of \( T, \) \( \mathcal{B} \) will simulate \( \text{Game}_{n} \) or \( \text{Game}_{\text{final}} \) to take an advantage from \( \mathcal{A} \) which can distinguish both games with non-negligible advantage \( \epsilon. \)

**Setup** \( \mathcal{B} \) chooses exponents \( b, a_1, y_1, y_{v_1}, y_{v_2}, y_{w}, h_1, \ldots h_{\ell}, \) randomly from \( \mathbb{Z}_p, \) and sets
\[ g = g, g^b, g^{a_1}, g^{a_2} = g^{c_2}, g^{b a_1}, g^{b a_2} = (g^{c_2})^b, w = g^{y_w}, \{g^{b i}; \forall i \in [1, \ell]\} \]
\[ \tau_1 = g^{y_{v_1} a_1}, \tau_2 = g^{y_{v_2}} (g^{c_2} y_{v_2}), \tau_1^{b}, \tau_2^{b}, e(g, g)^{a_{1} b} = e(g^{c_1}, g^{c_2})^{b a_1}. \]

It implicitly sets \( v = g^{y_w}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}, \alpha = c_1 \cdot c_2, a_2 = c_2. \) It should be noted that \( MSK \) cannot be explicitly calculated because \( \mathcal{B} \) does not know \( g^a. \)

**Phase I and II** For semi-functional keys, if the adversary requests a private key for \( x, \) it randomly generates \( r_1, r_2, z_1, z_2, \gamma', h'_1, \ldots, h'_{\ell}, \) from \( \mathbb{Z}_p \) and sets \( T_{\kappa} \bar{a} g_{k} = k \bar{E}_k(x, \bar{h}') \) where \( \bar{h}' = (h'_1, \ldots, h'_{\ell}). \) It, then, creates semi-functional key as follows:
\[ D_1 = (g^{c_2})^{-\gamma} a_1 v^r, D_2 = (g^{c_2})^{\gamma} v_1^r g^{z_1}, D_3 = (g^{b})^{-z_1}, D_4 = (g^{c_1})^{a_1} g^{a_2} v_2^r g^{z_2}, \]
\( D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2 - b}, \quad D_7 = g^{r_1}, \quad \vec{K} = (g^{kE(x,h)}v_{i}^{T\vec{a}g_k})^{r_1}, \quad T\vec{a}g_k. \)

where \( \vec{h} = (h_1, ..., h_\ell). \) This implicitly sets \( \gamma = c_1 + \gamma' \) and \( r = r_1 + r_2. \)

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for \( y^* \) with messages \( M_0, M_1, B \) randomly selects \( \beta \) from \( \{0, 1\} \). Then it generates random values \( \kappa', s_1, t, h_\ell'', ..., h_\ell'' \) from \( \mathbb{Z}_p \) and sets \( T\vec{a}g_c = c\vec{E}(y^*, \vec{h}''') \) where \( \vec{h}''' = (h_1'', ..., h_\ell'''). \) It, then, creates the challenge ciphertext as follows:

\[
\begin{align*}
C &= M_3 T^{a_1b}, \\
C_1 &= g^{s_1b} + (g^{c_1})^b, \\
C_2 &= g^{b-a_1:s_1}, \\
C_3 &= g^{a_1:s_1}, \\
C_4 &= (g^{c_2})^{\kappa'-b}, \\
C_5 &= (g^{c_2})^{\kappa'}, \\
C_6 &= (g^{c_2})^{y_1(g^{c_2})^{y_2 \cdot \kappa'}}, \\
C_7 &= (g^{c_2})^{y_2 \cdot (g^{c_2})^{y_2 \cdot \kappa' - b} \cdot w_1}, \\
C_8 &= g^t, \\
\vec{E} &= (g^{E(y^*, \vec{h}''')})T\vec{a}g_c, \\
\vec{Tag}_c.
\end{align*}
\]

This implicitly sets \( s_2 = c_3 \) and \( \kappa = \kappa' - c_3 \).

In this simulation, if \( T = e(g, g)^{c_1 c_2 c_3} \), then this has properly simulated \( \text{Game}_{eq}. \) Otherwise, if \( T \) is random, a random value is added into \( M_3 \). Hence, it has properly simulated \( \text{Game}_{final}. \)

\[\square\]

5.5 Our Efficient Compiler under Static Assumptions

In this section, we introduce our construction in asymmetric bilinear maps under \( \text{SXDH}, \text{Simple LW2} \) and \( \text{Simple LW3} \) assumptions. While some assumptions we uses are shared with IBE scheme of Lewko and Waters [LW10], the underlying technique is fairly different. Our construction with asymmetric bilinear maps is simpler although it is created without any change of the properties of our encodings.

5.5.1 Our Construction

For a tag based encoding \( \text{TE} \) of a predicate \( R \) where \( R : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \), with \( \ell \) which is an integer to associated with \( R \), \( \text{FE2(TE)} \) is constructed by four algorithms as follows:

- **Setup(\( \lambda \)):** The algorithm sets \( G_1, G_2 \) and \( G_T \leftarrow \mathcal{G}(\lambda, p) \), and generates \( g_1 \in G_1 \) and \( g_2 \in G_2 \), randomly. Also, it randomly selects \( \alpha, b, y_v, y_a, y_w, h_1, ..., h_\ell \) from \( \mathbb{Z}_p \) and sets \( v_1 = g_1^{y_v}, u_1 = g_1^{y_a}, v_1 = g_1^{y_v}, u_1 = g_1^{y_a}, v_2 = g_2^{y_v}, u_2 = g_2^{y_a}, \) and \( w_2 = g_2^{y_w} \). It publishes public parameters as

\[
g_1, g_1^b, (v_1 u_1^b), v_1, g_1^{h_1}, g_1^{h_2}, ..., g_1^{h_\ell}, e(g_1, g_2)^\alpha.
\]
It sets MSK as $g_2, g_2^a, v_2, u_2, w_2, g_2^{h_1}, ..., g_2^{h_\ell}$.

- **Keygen**(MSK, PK, $x$): The algorithm chooses randomly $r, h_1', ..., h_\ell' \in \mathbb{Z}_p$, and sets $\overrightarrow{Tag}_k = k\overrightarrow{E}(x, \vec{h})$ where $\vec{h}' = (h_1', ..., h_\ell')$. Then, it creates $SK$ such that

$$D_1 = g_2^a v_2^r, \quad D_2 = u_2^r, \quad D_3 = g_2^{-r}, \quad \vec{K} = (g_2^{k\overrightarrow{E}(x, \vec{h})_2^r} \overrightarrow{Tag}_k)_r.$$

It outputs $SK = (D_1, ..., D_3, \vec{K}, \overrightarrow{Tag}_k)$.

- **Encrypt**(PK, M, $y$): The algorithm chooses $a, s, h_1'', ..., h_\ell''$ from $\mathbb{Z}_p$, and sets $\overrightarrow{Tag}_c = c\overrightarrow{E}(y, \vec{h}'')$ where $\vec{h}'' = (h_1'', ..., h_\ell'')$. It creates $CT$ as follows

$$C = M \cdot (e(g_1, g_2)^a), \quad C_1 = (g_1)^a, \quad C_2 = (g_2^b)^a, \quad C_3 = (v_1 u_1)^a w_1^{-a}, \quad C_4 = g_1^a, \quad \vec{E} = (g_1^{c\overrightarrow{E}(y, \vec{h})} \overrightarrow{Tag}_c)^a.$$

It outputs $CT = (C_1, C_2, C_3, C_4, \vec{E}, \overrightarrow{Tag}_c)$.

- **Decrypt**(SK, CT, PK): First, the algorithm calculates

$$A_1 = e(C_1, D_1) e(C_2, D_2) e(C_3, D_3).$$

Because $R(x, y) = 1$, there exist reconstruction vectors $\vec{m}_x$ and $\vec{m}_y$ such that $\vec{m}_x \overrightarrow{E}(x, \vec{h}) = \vec{m}_y c \overrightarrow{E}(y, \vec{h})$. If $\vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c$ is 0, it aborts. Otherwise,

$$A_2 = e(C_4, \vec{K}^{\vec{m}_x}) e(\vec{E}^{\vec{m}_y}, D_3) = e(w_1, g_2)^{ra(\vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c)}.$$

Therefore, $M = C \cdot A_2^{1/(\vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c)}/A_1$.

**Correctness** Calculating $A_1$ is trivial. $A_1 = e(g_1, g_2)^{ra} e(w_1, g_2)^{ar}$. For $\vec{m}_x$ and $\vec{m}_y$ such that $\vec{m}_x \overrightarrow{E}(x, \vec{h}) = \vec{m}_y c \overrightarrow{E}(y, \vec{h})$, the correctness of $A_2$ is calculated as follows

$$A_2 = e(C_4, \vec{K}^{\vec{m}_x}) e(\vec{E}^{\vec{m}_y}, D_3)$$

$$= e(g_1^a, (g_2^{k\overrightarrow{E}(x, \vec{h})} \overrightarrow{Tag}_k)^r \vec{m}_x) / e((g_1^{c\overrightarrow{E}(y, \vec{h})} \overrightarrow{Tag}_c)^a \vec{m}_y, g_2^{-r})$$

$$= e(g_1, g_2)^{ra \cdot a \cdot \vec{m}_x k\overrightarrow{E}(x, \vec{h})} e(g_1, w_2)^{ra \cdot a \cdot \vec{m}_y \overrightarrow{Tag}_k} e(g_1, g_2)^{-ra \cdot a \cdot \vec{m}_y c\overrightarrow{E}(y, \vec{h})} e(w_1, g_2)^{-ra \cdot a \cdot \vec{m}_y \overrightarrow{Tag}_c}$$

$$= e(g_1, w_2)^{ra \cdot a \cdot \vec{m}_x \overrightarrow{Tag}_k} e(w_2, g_1)^{-ra \cdot a \cdot \vec{m}_y \overrightarrow{Tag}_c}$$

$$= e(w_1, g_2)^{ra \cdot a \cdot \vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c}.$$

The last equality holds since the discrete log of $w_1$ base $g_1$ equals the discrete log of $w_2$ base $g_2$. Therefore, $M = C \cdot A_2^{1/(\vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c)}/A_1$.

**Remark 5.4.** Alternatively, to reduce the number of pairing computations, the decryption algorithm first calculates $a_1 := \vec{m}_x \overrightarrow{Tag}_k - \vec{m}_y \overrightarrow{Tag}_c$ and sets $\vec{E} :=$
Then, it calculates $M = C \cdot A'_2/A'_1$ where $A'_1 = e(C_1, D_1)e(C_2, D_2)$ and $A'_2 = e(C_4, R\tilde{m}_x/a_1)e(\tilde{E}, D_3)$. It is worth noting that we can calculate $A'_2$ only by two pairing operations

$$A'_2 = e(g_{a_1}^\gamma, \prod_{i \in [1,|\tilde{K}|]} K_{m_{x,i}}^{r_{i}}) = e(C_{3}^{-1}, \prod_{i \in [1,|\tilde{E}|]} E_{i}^{m_{y,i}/a_1}, g_{-r_1}).$$

where $K_i, E_i, m_{x,i}$ and $m_{y,i}$ are the $i$th coordinate of vectors $\tilde{K}, \tilde{E}, \tilde{m}_x$ and $\tilde{m}_y$, resp.

**Theorem 5.1. (Informal)** Suppose there exists Tag based Encoding $TE$, then our $FE2(TE)$ is adaptively secure under SXDH, Simple LW2 and (Asymmetric) DBDH assumptions.

**Proof:** This is proved by lemmas 5.7, 5.8 and 5.9.

### 5.5.2 Security Analysis

**Semi-functional Ciphertext.** By running Encrypt algorithm for a message $M$, the algorithm generates a normal ciphertext $CT = (C', C'_1, ..., C'_4, \tilde{E}, \tilde{K}, \tilde{Tag}_e)$. Then, it randomly selects $\gamma \in \mathbb{Z}_p$ and sets

$$C = C', \ C_1 = C'_1, \ C_2 = C'_2g_1^\gamma, \ C_3 = C'_3u_1^\gamma, \ C_4 = C'_4, \ \tilde{E} = \tilde{E}', \ \tilde{Tag}_e = \tilde{Tag}'_e.$$

**Semi-functional Key.** The algorithm generates a normal key $SK = (D'_1, D'_2, D'_3, \tilde{K}', \tilde{Tag}'_k)$ by running KeyGen algorithm. Then, it randomly selects $\omega$ from $\mathbb{Z}_p$ and sets

$$D_1 = D'_1g_2^{-\omega}, \ D_2 = D'_2g_2^\omega, \ D_3 = D'_3, \ \tilde{K} = \tilde{K}', \ \tilde{Tag}_k = \tilde{Tag}'_k.$$

It should be noted that if a semi-functional key is being used for decrypting a semi-functional ciphertext, $e(g_1, g_2)^\omega$ will be added to the message encrypted.

We will prove an adaptive security of our symmetric construction using exactly the same steps with our compiler with asymmetric pairings.

**Lemma 5.7. (Semi-functional Ciphertext Invariance)** Suppose that there exists an algorithm $A$ which distinguishes $\text{Game}_{\text{real}}$ and $\text{Game}_0$ with a non-negligible advantage $\epsilon$. Then, we can build an algorithm $B$ which breaks SXDH assumption with $\epsilon$.

**Proof:** First, from SXDH assumption, $B$ takes as instances $\{g_1, g_1^d, g_1^u, T \in G_1, g_2 \in G_2\}$. Depending on the value of $T$, $B$ will simulate $\text{Game}_{\text{real}}$ or $\text{Game}_0$ to take an
advantage from $A$ which can distinguish both games with non-negligible advantage $\epsilon$.

**Setup** The algorithm chooses random exponents $\alpha, y_v, y_w, h_1, \ldots h_{\ell}$ from $\mathbb{Z}_p$. Then, it implicitly sets $b = d$. Also, it sets $v_1 = g_1^{y_v}, u_1 = g_1^{y_w}, w_1 = g_1^{y_v}$ and $\bar{h} = (h_1, \ldots, h_{\ell})$. It publishes the public parameters following:

$$g_1, g_1^b = g_1^d, (v_1 u_1)^b = g_1^{y_v} (g_1^d)^{y_w}, w_1 = g_1^{y_v}, g_1^{-1} \in \mathbb{G}, e(g_1^c, g_2)^{a^2}.$$ 

It also sets MSK $\{g_2, g_2^a, v_2 = g_2^{y_v}, u_2 = g_2^{y_w}, w_2 = g_2^{y_v}, g_2^\bar{h}\}$.

**Phase I and II** Since $B$ knows all MSK. It can generate normal keys using Keygen.

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for $y^\gamma \in \mathcal{Y}$ with messages $M_0, M_1$, $B$ randomly selects $a, h_1', \ldots, h_{\ell}' \in \mathbb{Z}_p$ and sets $s = z$ and $T_{\bar{a}g_c} = cE(y^\gamma, \bar{h}')$ where $\bar{h}' = (h_1', \ldots, h_{\ell}')$. The algorithm selects $\beta$ from $\{0, 1\}$ and sets the challenge ciphertexts as

$$C = M \cdot e(g_1^d, g_2^a), C_1 = g_1^z, C_2 = T, C_3 = (g_1^z)^{y_v} T^{y_v} g_1^{-y_w}, a, C_4 = g_1^a, e = (g_1^{cE(y^\gamma, \bar{h})} g_1^{y_v T_{\bar{a}g_c}})^a, T_{\bar{a}g_c}$$

Therefore, if $T = g_1^{dz}$, this is a properly distributed normal ciphertext and $B$ has simulated Game\textsubscript{real}. Otherwise, if $T$ is a random value, we write $T = g_1^{dz} g_1^{\gamma'}$. It implies that $\gamma$ equals to $a \gamma'$ and the challenge ciphertext is an well distributed semi-functional ciphertext. Therefore, $B$ has simulated properly Game\textsubscript{0}.

\[\square\]

**Lemma 5.8. (Semi-functional key Invariance)** Suppose that there exists an algorithm $A$ which distinguishes Game\textsubscript{k-1} and Game\textsubscript{k} with a non-negligible advantage $\epsilon$. Then, we can build an algorithm $B$ which breaks Simple LW2 assumption with $\epsilon$.

**Proof:** First, from LW assumption, $B$ takes as instances $\{f_1, f_1^d, f_1^{dtw}, f_1^{dt} \in G_1, f_2, f_2^d, f_2^{dtw}, f_2^{dt} \in G_2\}$. Depending on the value of $T$, $B$ will simulate Game\textsubscript{k-1} or Game\textsubscript{k} to take an advantage from $A$ which can distinguish both games with non-negligible advantage $\epsilon$.

**Setup** The algorithm chooses random exponents $\alpha, b, y_v', y_w', h_1', \ldots h_{\ell}', h_1''', h_{\ell}'''$ from $\mathbb{Z}_p$. It sets $g_1 = f_1$ and $g_2 = f_2$. Then, it implicitly sets $y_a = w, y_v = bw + y_v'$ and $y_v = -d + y_v'$. It publishes the public parameters as follows:

$$g_1, g_1^b, (v_1 u_1) = (g_1)^{y_v'}, w_1 = (g_1^d)^{-1} g_1^{y_v'}, \{g_1^{h_i} = (g_1^d)^{h_i}, g_1^{h_i}, \forall i \in [1, \ell]\}, e(g_1^c, g_2)^a.$$
also, it sets MSK as follows

\[ g_2, g_2^w, v_2 = (g_2^w)^{-b} g_2^h, u_2 = g_2^w, w_2 = (g_2^d)^{-1} g_2^h, \{g_2^h| \forall i \in [1, \ell]\} \]

**Phase I and II** For normal keys (> k), the algorithm uses the key generation algorithm (Keygen). Since it knows all public parameters and MSK, it can properly generate normal keys. To response the first k – 1 semi-functional key queries (< k), it generates a normal key \( D_1', D_2', D_3', \tilde{K}', T\tilde{a}g_k' \) using Keygen. Then, it randomly selects \( \omega \in \mathbb{Z}_n \) and adds semi-functional parts as

\[ D_1 = D_1'g_2^{bw}, D_2 = D_2'g_2^c, D_3 = D_3', \tilde{K} = \tilde{K}', T\tilde{a}g_k = T\tilde{a}g_k' \]

To generate the \( k^{th} \) key for a predicate \( x \in \mathcal{X} \), first, \( B \) sets \( r = c \) and \( T\tilde{a}g_k = k\tilde{E}(x, \tilde{h}') \) and generates the key as follows:

\[ D_1 = g_2^o T^{-b}(g_2^c)^{y'}, D_2 = T, D_3 = (g_2^c)^{-1}, \tilde{K} = g_2^{k\tilde{E}(x, \tilde{h}'' + y''r')}, T\tilde{a}g_k \]

where \( \tilde{h}' = (h_1', ..., h_i') \) and \( \tilde{h}'' = (h_1'', ..., h_i'') \).

It should be noted that \( \tilde{K} \) can be calculated by linearity since

\[ \tilde{K}^{1/c} = g_2^{k\tilde{E}(x, \tilde{h})} T\tilde{a}g_k = (g_2^d)^{k\tilde{E}(x, \tilde{h}'')} g_2^{k\tilde{E}(x, \tilde{h}')} (g_2^d - k\tilde{E}(x, \tilde{h}')) g_2^{y''k\tilde{E}(x, \tilde{h}')} = g_2^{k\tilde{E}(x, \tilde{h}'' + y''r')} \]

Therefore, if \( T = g_2^{cw} \), then this is a properly distributed normal key. Otherwise, if \( T \) is a random, this is a proper semi-functional key if we denote \( T = g_2^{cw}g_2^c \).

**Challenge Ciphertext** When the adversary requests the challenge ciphertext for \( y^* \in \mathcal{Y} \) with messages \( M_0, M_1, B \) randomly selects \( t', s \in \mathbb{Z}_n \) and implicitly sets \( a = tdw + t' \) and \( \gamma = -d^2t \). \( \gamma \) is randomly distributed to the adversary since the value of \( t \) does not appear anywhere else except \( a \), but the value is not revealed in \( a \) due to uniquely allocated random value \( t' \). The algorithm selects \( \beta \) from \( \{0, 1\} \) and sets \( T\tilde{a}g_c = c\tilde{E}(y^*, \tilde{h}') \). It creates the challenge ciphertexts as

\[ C = M \cdot e(g_1, g_2)^as, C_1 = g_1^s, C_2 = g_1^{bs} g_2^{-dt'}, C_3 = g_1^{ys} (g_1^d)^{t'} g_1^{-y'r'} (g_1^{adw} - y'w) \]

\[ C_4 = g_1^{dtw} g_1^{t'}, B = (g_1^{dtw} g_1^{yE(y^*, \tilde{h}'' + y''r')})^t c\tilde{E}(y^*, \tilde{h}'' + y''r'), T\tilde{a}g_c. \]

In \( C_3, u_1 = g_1^{dtw} \) which is not given in the instance is cancelled out by \( w^{-a} = g_1^{dtw} (g_1^d)^{t'} g_1^{-y'r'} (g_1^{adw} - y'w) \). Since \( R(x, y^*) = 0 \), sharing \( \tilde{h}' \) between \( T\tilde{a}g_k \) and \( T\tilde{a}g_c \) are statistically hidden to the adversary by \( h \)-hiding. It should be noted that the values of \( \tilde{h}' \) are initially information theoretically hidden due to the corresponding
values of $\vec{h}^\nu$. Therefore, if $T = g_1^{\nu^w}$, $B$ has simulated $\text{Game}_{k-1}$. Otherwise, if $T$ is a
random value, $B$ has simulated properly $\text{Game}_k$. \hfill $\square$

**Lemma 5.9. (Semi-functional Security)** Suppose that there exists an algorithm $A$ which distinguishes $\text{Game}_q$ and $\text{Game}_{\text{final}}$ with non-negligible advantage $\epsilon$. Then, we can build an algorithm $B$ which breaks (Asymmetric) DBDH assumption with advantage $\epsilon$.

**Proof:** Firstly, $B$ takes an $\{g_1, g_1^{c_2}, g_1^{c_3} \in G_1, g_2, g_2^{c_1}, g_2^{c_2} \in G_2, T \in G_T\}$ as an instance from $\text{ADBDH}$ assumption. Depending on the value of $T$, $B$ will simulate $\text{Game}_q$ or $\text{Game}_{\text{final}}$ to take an advantage from $A$ which can distinguish both games with non-negligible advantage $\epsilon$.

**Setup** Algorithm $B$ chooses exponents $y_e, y_u, y_w, h_1, ..., h_\ell$, randomly from $\mathbb{Z}_p$ and set implicitly $\alpha = c_1c_2$ and $b = c_2$. Then, it publishes public parameters as follows:

$$g_1, g_1^{b} = g_1^{c_2}, (v_1u_1^b) = (g_1)^{y_e}(g_1^{c_2})^{y_u}, w_1 = g_1^{y_w}, g_1^{h_1}, ..., h_1^{h_\ell},$$

$$e(g_1, g_2)^\alpha = e(g_1^{c_2}, g_2^{c_1}).$$

For $MSK$, it sets $\{g_2, v_2 = g_2^{y_e}, u_2 = g_2^{y_u}, w_2 = g_2^{y_w}, g_2^{h_1}, ..., g_2^{h_\ell}\}$. However, it does not know $g_2^\alpha$ of $MSK$ since $g_2^{c_1c_2}$ is not given.

**Phase I and II** For semi-functional keys, if the adversary requests a private key for $x$, it randomly generates $r, w', h_1', ..., h_\ell'$ from $\mathbb{Z}_p$ and implicitly sets $\omega = c_1 + w'$. It sets $T\vec{a}g_k = k\vec{E}(x, \vec{h}')$ where $\vec{h}' = (h_1', ..., h_\ell')$ and creates semi-functional key as follows:

$$D_1 = g_2^{y_w}(g_2^{c_2})^{-w'}, D_2 = g_2^{y_u}(g_2^{c_1})^{y_u}, D_3 = g_2^{-r}, k\vec{K} = (g_2^{k\vec{E}(x, \vec{h}')})^{y_w}T\vec{a}g_k, T\vec{a}g_k.$$

**Challenge Ciphertext** When the adversary requested the challenge ciphertext for $y^*$ with messages $M_0, M_1$, $B$ randomly selects $\beta$ from $\{0, 1\}$. Then it generates random values $\gamma', a, h_1'', ..., h_\ell''$ from $\mathbb{Z}_p$. $B$, then, sets $s = c_3$ and $\gamma = -c_2c_3 + \gamma'$, implicitly. Sharing $c_3$ between $s$ and $\gamma$ is acceptable since the value of $c_3$ does not reveal from $\gamma$ due to $\gamma'$ which only appears on $\gamma$. It sets $T\vec{a}g_c = c\vec{E}(y^*, \vec{h}'')$ where $\vec{h}'' = (h_1'', ..., h_\ell'')$. It creates the challenge ciphertext as follows:

$$C = M_\beta T, C_1 = g_1^{c_3}, C_2 = g_1^{\gamma'}, C_3 = (g_1^{c_3})^{y_w}g_1^{-y_wa}g_1^{y_\nu^\gamma'}, C_4 = g_1^{a},$$

$$\vec{E} = (g_1^{\vec{E}(y^*, \vec{h}'')})^{y_w}T\vec{a}g_c)^a, T\vec{a}g_c.$$  

In this simulation, if $T = e(g_1, g_1)^{c_1c_2c_3}$, then this has properly simulated $\text{Game}_q$. Otherwise, if $T$ is random, $A$ random value is added into $M_\beta$. Hence, it has properly simulated $\text{Game}_{\text{final}}$. \hfill $\square$
5.6 New Schemes

In this section, we provide instances for our encoding to achieve new functional encryption schemes. The instances of Inner Product Encryption (IPE) with short keys, Dual Spatial Encryption (Dual SE) with short keys and HIBE with short ciphertexts will be presented. Inner Product Encryption (IPE) with short keys and Dual Spatial Encryption (Dual SE) are new instances. HIBE with short ciphertexts is also found in [Wee14, CW14b], but applying this instance to our compilers results in new schemes in both asymmetric and symmetric bilinear maps. It should be noted that security analysis of each scheme is replaced by showing that the corresponding instance satisfies the properties that tag based encoding requires.

Inner Product Encryption with short keys

Let define $X = Y := \mathbb{Z}_p^\ell$. For all, $\vec{x} \in X$ and $\vec{y} \in Y$, $R(x,y) = 1$ iff $\langle \vec{x}, \vec{y} \rangle = 0$.

- $kE(\vec{x}, \vec{h}) := u_0 + \vec{x}^\top \vec{u} \in \mathbb{Z}_p$ where $\vec{h} = (u_0, \vec{u}) \in \mathbb{Z}_p^{\ell}$.
- $cE(\vec{y}, \vec{h}) := (-h_1(y_2/y_1) + h_2, ..., -h_1(y_\ell/y_1) + h_\ell) \in \mathbb{Z}_p^{\ell-1}$
- **Reconstruction:** $\vec{m}_x = 1$ and $\vec{m}_y = (x_2, ..., x_\ell)$.
- **Linearity:** Firstly, the linearity of $kE$ holds trivially since $\langle \vec{h}, \vec{x} \rangle + \langle \vec{h}', \vec{x} \rangle = \langle \vec{h} + \vec{h}', \vec{x} \rangle$. Also, $cE(\vec{y}, \vec{h}) + cE(\vec{y}, \vec{h}') = cE(\vec{y}, \vec{h} + \vec{h}')$ since, for all $i \in [1, \ell - 1]$, $-h_1(y_{i+1}/y_1) + h_{i+1} - h'_1(y_{i+1}/y_1) + h'_{i+1} = -(h_1 + h'_1)(y_{i+1}/y_1) + h_{i+1} + h'_{i+1}$.
- **$h$-hiding:** In the following equation, the first $\ell - 1$ coordinates of the right hand vector in the above equation are independent from the last coordinate by $\ell$-wise independence [AL12]. Hence, sharing $\vec{h}$ between $kE, cE$ is hidden to the adversary.

$$
\begin{pmatrix}
-y_2/y_1 & 1 \\
\vdots & \ddots \\
-y_\ell/y_1 & 1 \\
x_1 & x_2 & x_3 & \cdots & x_\ell
\end{pmatrix}
\begin{pmatrix}
h_1 \\
h_2 \\
h_{\ell-1} \\
h_\ell
\end{pmatrix}
= 
\begin{pmatrix}
-h_1(y_2/y_1) + h_2 \\
\vdots \\
-h_1(y_\ell/y_1) + h_\ell \\
\langle \vec{h}, \vec{x} \rangle
\end{pmatrix}
$$

Dual Spatial Encryption with short keys

For a matrix $M \in \mathbb{Z}_p^{(\ell-1)\times d}$ and a vector $\vec{c} \in \mathbb{Z}_p^{\ell-1}$, it defines the affine space $\text{Aff}(M, \vec{c}) = \{ M\vec{w} + \vec{c} | \vec{w} \in \mathbb{Z}_p^d \}$. Then, $R(\vec{x}, \text{Aff}(M, \vec{c})) = 1$ iff there exists $\vec{w} \in \mathbb{Z}_p^d$ such that $M\vec{w} + \vec{c} = \vec{x}$.

- $kE(\vec{x}, \vec{h}) := u_0 + \vec{x}^\top \vec{u} \in \mathbb{Z}_p$ where $\vec{h} = (u_0, \vec{u}) \in \mathbb{Z}_p^{\ell}$.
- $cE(\text{Aff}(M, \vec{c}), \vec{h}) := (u_0 + \vec{c}^\top \vec{u}, M^\top \vec{u}) \in \mathbb{Z}_p^{d+1}$
CHAPTER 5. TAG BASED ENCODING

• **Reconstruction:** $\vec{m}_x = 1$ and $\vec{m}_y = (1, \tilde{\vec{w}}) \text{ s.t. } M\tilde{\vec{w}} + \vec{c} = \vec{x}$.

• **Linearity:** Since all coordinates of $k\vec{E}(\vec{x}, \vec{h})$ and $c\vec{E}(\text{Aff}(M, \vec{c}), \vec{h})$ are linear over $\vec{h}$, linearity is proved, trivially.

• **$h$-hiding:** In the following equation, for $\vec{x} \in \mathcal{X}$, there is no $\tilde{\vec{w}}$ such that $M\tilde{\vec{w}} + \vec{c} = \vec{y}$ since $R(\vec{x}, \text{Aff}(M, \vec{c})) = 0$. Hence, the last row of the matrix on the left is linearly independent from the other rows. Hence, it is hidden that they share $u_0$ and $\vec{u}$.

\[
\begin{pmatrix}
1 & \vec{c} \\
\vec{0} & M^\top \\
1 & \vec{x}^\top
\end{pmatrix}
\begin{pmatrix}
u_0 \\
u_0 + \vec{c}^\top \vec{u} \\
u_0 + \vec{x}^\top \vec{u}
\end{pmatrix}
= \begin{pmatrix}
h_0 \\
h_\ell-1 \\
h_\ell
\end{pmatrix}
\]

**HIBE with short ciphertexts [Wee14, CW14b]**

For a vector $I\vec{D}_d := (id_1, ..., id_d) \in \mathbb{Z}_p^d$ and a vector $I\vec{D}_d' := (id_1', ..., id_d') \in \mathbb{Z}_p^{d'}$, $R(I\vec{D}_d, I\vec{D}_d') = 1$ if $d \leq d'$ and $id_i = id'_i \forall i \in [1, d]$.

• $k\vec{E}(I\vec{D}_d, \vec{h}) := (h_0 + h_1(id_1) + ... + h_d(id_d), h_{d+1}, ..., h_\ell) \in \mathbb{Z}_p^{\ell-d}$ where $\vec{h} \in \mathbb{Z}_p^{\ell+1}$

• $c\vec{E}(I\vec{D}_d', \vec{h}) := h_0 + h_1(id_1') + ... + h_{d'}(id_{d'}) \in \mathbb{Z}_p$

• **Reconstruction:** $\vec{m}_x = (1, id_{d+1}', ..., id_{d'}, 0, ..., 0) \in \mathbb{Z}_p^{\ell-d}$ and $\vec{m}_y = 1$.

• **Linearity:** Linearity is proved trivially since all coordinates of $k\vec{E}(I\vec{D}_d, \vec{h})$ and $c\vec{E}(I\vec{D}_d', \vec{h})$ are linear over $\vec{h}$.

• **$h$-hiding:** In the following equation, the first $\ell - d + 1$ rows are linearly independent with the last row of matrix on the left since $id_d \neq id_d'$ and $h_0, ..., h_\ell$ appear at most twice. Therefore, the sharing $\vec{h}$ between the first $\ell + 1$ coordinates of the vector of the right hand of the equation with the last coordinate of the vector is hidden.

\[
\begin{pmatrix}
1 & id_1 & \cdots & id_d \\
1 & id_1' & \cdots & id_{d'}
\end{pmatrix}
\begin{pmatrix}
h_0 + h_1(id_1) + ... + h_d(id_d) \\
h_{d+1} \\
h_\ell
\end{pmatrix}
= \begin{pmatrix}
h_0 \\
h_\ell-1 \\
h_\ell
\end{pmatrix}
\]
5.7 Instances from Existing Schemes

In this section, we extract instances from the literature. New instances that result in Identity based encryption (IBE) [Wat09], Public Attribute Inner Product Encryption (PAIPE) with short ciphertexts [AL12], Spatial Encryption (SE) [CZF12] and Doubly Spatial Encryption (DSE) [Wee14] are derived from previous works. It should be noted that the schemes created by applying those instances to our compiler with symmetric bilinear maps are identical with the original constructions of their literature.

Waters’ IBE is a good example to describe our framework due to its simplicity.

**Identity Based Encryption [Wat09]**

Let $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p$. For all $ID \in \mathcal{X}$ and $ID' \in \mathcal{Y}$, $R(ID, ID') = 1$ iff $ID = ID'$.

- $kE(ID, (y_u, y_h)) := (y_u ID + y_h) \in \mathbb{Z}_p$ where $\overrightarrow{h} = (y_u, y_h) \in \mathbb{Z}_p^2$
- $cE(ID', (y_u, y_h)) := (y_u ID' + y_h) \in \mathbb{Z}_p$

**Reconstruction:** This is an exact cancellation. Therefore, $\overrightarrow{m}_x = \overrightarrow{m}_y = 1$.

**Linearity:** For all $\overrightarrow{h}' = (y'_u, y'_h)$,

$$kE(ID, (y'_u, y'_h)) + kE(ID, (\tilde{y}_u, \tilde{y}_h)) = y'_u ID + y'_h + \tilde{y}_u ID + \tilde{y}_h$$

$$= kE(ID, (y'_u + \tilde{y}_u, y'_h + \tilde{y}_h))$$

The linearity of $cE(ID', (y'_u, y'_h))$ is identical showed with $kE(ID, (y'_u, y'_h))$.

**h-hiding:** $y_u ID + y_h$ and $y_u ID' + y_h$ are pair-wise independent since $R(ID, ID') = 0$ (i.e. $ID \neq ID'$). Hence, for given $y_u ID + y_h$, the value of $y_u ID' + y_h$ is uniformly distributed from $\mathbb{Z}_p$. Therefore, it is not distinguishable from $y'_u ID' + y'_h$ where $(y'_u, y'_h)$ is randomly selected from $\mathbb{Z}_p^2$.

Public Attribute Inner Product Encryption with short ciphertexts and Spatial Encryption with short ciphertexts are the symmetric conversion of IPE with short keys and Dual SE with short keys which were introduced in the previous section. If we let $kE'$ and $cE'$ denote the encodings of the original schemes (e.g. IPE with short keys), then we define $kE = cE'$ and $cE = kE'$. All properties are identically proved with the schemes in previous sections.

**Public Attribute Inner Product Encryption with short ciphertexts [AL12]**

Let $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p^\ell$. For all, $\overrightarrow{x} \in \mathcal{X}$ and $\overrightarrow{y} \in \mathcal{Y}$, $R(x, y) = 1$ iff $\langle \overrightarrow{x}, \overrightarrow{y} \rangle = 0$

- $kE(\overrightarrow{x}, \overrightarrow{h}) := \{-h_1(x_i/x_1) + h_i\}_{i=2, \ldots, \ell} \in \mathbb{Z}_p^{\ell-1}$ where $\overrightarrow{h} \in \mathbb{Z}_p^\ell$
• $cE(\vec{y}, \vec{h}) := \langle \vec{h}, \vec{y} \rangle \in \mathbb{Z}_p$

• All properties are proved similarly with our IPE with short keys.

**Spatial Encryption with short ciphertexts [CZF12]**

For a matrix $M \in \mathbb{Z}_p^{(\ell-1) \times d}$ and a vector $\vec{c} \in \mathbb{Z}_p^{\ell-1}$, it defines the affine space $\text{Aff}(M, \vec{c}) = \{ M \vec{w} + \vec{c} | \vec{w} \in \mathbb{Z}_p^d \}$. Then, $R(\text{Aff}(M, \vec{c}), \vec{y}) = 1$ iff there exists $\vec{w} \in \mathbb{Z}_p^d$ such that $M \vec{w} + \vec{c} = \vec{y}$.

• $kE(\text{Aff}(M, \vec{c}), \vec{h}) := (u_0 + \vec{c}^\top \vec{u}, M^\top \vec{u}) \in \mathbb{Z}_p^{d+1}$ where $\vec{h} = (u_0, \vec{u}) \in \mathbb{Z}_p^\ell$.

• $cE(\vec{y}, \vec{h}) := (u_0 + \vec{y}^\top \vec{u}) \in \mathbb{Z}_p$

• All properties are proved similarly with our Dual SE with short keys.

**Doubly Spatial Encryption [CZF12]**

For affine matrices $X \in \text{AffM}(\mathbb{Z}_p^{\ell \times d})$ and $Y \in \text{AffM}(\mathbb{Z}_p^{\ell \times f})$, $R(X, Y) = 1$ iff there exist $\vec{w} \in \text{AffM}(\mathbb{Z}_p^d)$ and $\vec{z} \in \text{AffM}(\mathbb{Z}_p^f)$ such that $\vec{w}X^\top = \vec{z}Y^\top$

• $kE(X, \vec{h}) := (X^\top \vec{h}^\top) \in \mathbb{Z}_p^d$ where $\vec{h} \in \mathbb{Z}_p^\ell$.

• $cE(Y, \vec{h}) := (Y^\top \vec{h}^\top) \in \mathbb{Z}_p^f$.

**Reconstruction:**

Since $\vec{w}X^\top = \vec{z}Y^\top$, $\vec{w} \cdot kE(X, \vec{h}) = \vec{z} \cdot cE(Y, \vec{h})$ (i.e. $\vec{m}_x = \vec{w}$, and $\vec{m}_y = \vec{z}$).

• **Linearity:** For all $\vec{h}$ and $\vec{h}'$ from $\mathbb{Z}_p^\ell$,

  $$kE(X, \vec{h}) + kE(X, \vec{h}') = X^\top \vec{h}^\top + X^\top \vec{h}'^\top = X^\top (\vec{h}^\top + \vec{h}'^\top) = kE(X, \vec{h}^\top + \vec{h}'^\top)$$

  $$cE(Y, \vec{h}) + cE(Y, \vec{h}') = Y^\top \vec{h}^\top + Y^\top \vec{h}'^\top = Y^\top (\vec{h}^\top + \vec{h}'^\top) = cE(Y, \vec{h}^\top + \vec{h}'^\top).$$

• **$h$-hiding:** In the following equation, there exist no $\vec{w}$ and $\vec{z}$ such that $\vec{w}X^\top = \vec{z}Y^\top$ since $R(X, Y) = 0$. Hence, the last $f$ rows of matrix on the left is linearly independent from the first $d$ rows. Therefore, sharing $\vec{h}$ is hidden.

$$
\begin{pmatrix}
X^\top \\
Y^\top
\end{pmatrix}
\begin{pmatrix}
\vec{h}^\top
\end{pmatrix} =
\begin{pmatrix}
X^\top \vec{h}_x^\top \\
Y^\top \vec{h}_y^\top
\end{pmatrix}$$
Chapter 6

KP-ABE with short ciphertexts

In this work, we introduce two KP-ABE schemes. One is in a small universe of attributes but the other supports a large universe of attributes. The techniques used to achieve these schemes are similar to those of tag based encryption, but the schemes in this chapter are semi-adaptively secure. Semi-adaptive security model is identical with adaptive security model except that the adversary must declare its target after receiving public keys but before querying any private keys. Therefore, semi-adaptive security model is weaker than adaptive security model but it still stronger than selective security model. It should be noted that tag based encoding achieves adaptively secure ABE scheme using IPE or Spatial encryption [LOS+10, Ham11], but we introduce these specific schemes since they achieve ABE scheme more efficiently using linear secret sharing scheme [Bei96] and also show how the limitation of the encoding framework can be eased using weaker security notion.

Achieving ABE employing linear secret sharing scheme using our tag based encoding in the previous chapter is daunting since the challenge key and the challenge ciphertext can have multiple attributes and some of them are shared between them. But, with linear secret sharing scheme, tags of both are correlated and this correlation cannot be hidden information theoretically. Therefore, the independence requirement of tag based encoding cannot be easily satisfied.

KP-ABE in a small universe First, KP-ABE in a small universe shows how we overcome the problem we mentioned previously. To solve the problem, we keep key elements of attributes which appear in both without any change of type from the normal challenge key. It means that semi-functional transformation applies only to attributes which appear only in the challenge key, but not in the challenge ciphertext. Therefore, in a semi-functional space of our scheme, no attribute is shared between the challenge ciphertext and the challenge key. However, since the challenge key and the challenge ciphertext in KP-ABE still have more than one attribute, reusing parameters to represent attributes (e.g. reusing $y_a$ and $y_h$ for all
attributes in Waters’ IBE) is still problem. To solve this problem, we also convert the domain of attributes from a large universe to a small universe, and allocate an unique random value for tag of each attribute. Although the disadvantage of the change is obvious, it enables us to solve the problem easily.

**KP-ABE with short ciphertexts in a large attribute universe** Achieving KP-ABE with short ciphertexts in a large attribute universe is more difficult because we cannot allocate unique random values for attributes since the total number of attributes in the system is not bounded or exponentially large. We must reuse some values for each tag. We solve this problem by designing a *nested dual system encryption*. We let semi-functional row denote a row of which corresponding key elements have the same distribution with those of a semi-functional key. Due to the similar reason of our KP-ABE with small universe, we convert rows in the challenge key from normal rows to semi-functional rows such that the corresponding attributes of rows $\rho(x)$ are not in the set of attributes for the challenge ciphertext $S^*$ where $\rho$ is a mapping from a row of an access matrix to an attribute in the access policy of the challenge key. For rows in the challenge key such that $\rho(x) \notin S^*$, instead of changing them from normal type to semi-functional type at once (i.e. changing the challenge key from normal type to semi-functional type), we change a row of an access matrix of the challenge key from normal type to semi-functional type one-by-one. In other words, we show that the invariance between a normal key and a semi-functional key by showing the invariance of a normal row and a semi-functional row, alternatively.

Since we isolate single row from the other rows, we do not care about tags for the other rows because they can be generated by a normal key generated algorithm. In other words, they can be created randomly. However, we still do not use pair-wise independence for our construction, since the challenge ciphertext still have multiple attributes. Therefore, we use $n$-wise independence [AL10] for our construction. In detail, we let $A_x$ denote the $x^{th}$ row of $A$. Tags for the $A_x$, $kTag_{j,x} \forall j \in [1,n]$ and a tag for the challenge ciphertext, $cTag$, are generated as

\[
\begin{pmatrix}
-\rho(x) & 1 \\
-(\rho(x))^2 & 1 \\
\vdots & \ddots \\
-(\rho(x))^n & 1 \\
c_0 & c_1 & c_2 & \cdots & c_n
\end{pmatrix}
\begin{pmatrix}
h'_0 \\
h'_1 \\
h'_2 \\
\vdots \\
h'_n
\end{pmatrix}
=
\begin{pmatrix}
kTag_{1,x} \\
kTag_{2,x} \\
kTag_{3,x} \\
\vdots \\
kTag_{n,x}
\end{pmatrix}
\begin{pmatrix}
c_0 \\
c_1 \\
c_2 \\
\vdots \\
c_n
\end{pmatrix}
\]

where $c_j$ is coefficients of $y^j$ of $\prod_{\rho(x) \in S^*}(y-\rho(x))$ and $S^*$ is the target set of attributes for the challenge ciphertext, and $h'_0, \ldots h'_n$ are values allocated to public parameters, but information theoretically hidden to the adversary. It means that the values of $h'_0, \ldots h'_n$ are not revealed anywhere else.
To claim \( n \)-wise independence in the matrix equation above, we must show that \( \rho(x) \) is not in \( S^* \) since the correlation can be detected to the adversary if \( \rho(x) \) is in \( S^* \) by checking \( \sum_{i \in [1,n]} c_i kTAg_{i,x} = cTag \). Therefore, similarity with our KP-ABE with short ciphertexts, we leave a row \( A_x \) as a normal row if \( \rho(x) \in S^* \).

In a short, we overcome the barriers to construct KP-ABE using Waters’ IBE by our technique. In KP-ABE in small attribute universe, we separate rows of an access structure by whether their corresponding attribute is included in \( S^* \), then changed the elements for all \( A_x \) such that \( \rho(x) \notin S^* \) in a challenge key to semi-functional type at once. Also, in KP-ABE with short ciphertexts, we additionally isolate each \( A_x \) such that \( \rho(x) \notin S^* \) via nested dual system encryption, then change all rows such that \( \rho(x) \notin S^* \) to semi-functional type one-by-one to permit a large attribute universe and short ciphertexts.

Comparisons with the results of other schemes having short ciphertexts are shown in Table 6.1. The first KP-ABE with short ciphertexts was introduced by Attrapadung, Libert and Panafieu [ALdP11]. It is expressive but is only selectively secure under \( q \)-type assumption and supports a small universe of attributes. Later, Attrapadung [Att14a] introduced an adaptively secure KP-ABE scheme with a constant sized ciphertext. However, it depends on several assumptions including two \( q \)-type assumptions in composite order groups. Subsequently, Chen and Wee [CW14c] achieved KP-ABE without \( q \)-types assumptions, but it is secure only in a small universe of attributes in composite order groups. Currently the best scheme is from Takashima [Tak14]. Using DPVS, it is semi-adaptively secure in prime order groups under a standard assumption. Unfortunately, the efficiency of our scheme cannot be directly compared to [Tak14] because Takashima’s scheme supports non-monotone access structures, while ours supports only monotone access structures. However, in a scenario requiring only monotone access policies, our scheme has smaller sized private keys and ciphertexts (Table 6.1). It should be noted that the sizes of keys and ciphertexts of [Tak14] for monotone access structures are identical to those for non-monotone access structures.

### 6.1 KP-ABE in a small universe

We introduce KP-ABE in a small universe \( U \) with \( |U| = n_t \). Our construction consists of four algorithms **Setup**, **KeyGen**, **Encrypt** and **Decrypt**. We omit \( n \) in the construction since it equals to \( n_t \).
Table 6.1: Comparison of KP-ABE Schemes Having Constant Sized Ciphertexts

<table>
<thead>
<tr>
<th>Assump.</th>
<th>ALP11</th>
<th>A14</th>
<th>CW14</th>
<th>T14</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universe</td>
<td>small</td>
<td>M</td>
<td>small</td>
<td>large</td>
<td>large</td>
</tr>
<tr>
<td>Order of $G$</td>
<td>Prime</td>
<td>Composite</td>
<td>Composite</td>
<td>Prime</td>
<td>Prime</td>
</tr>
<tr>
<td>$n$ (\text{CT})</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>$\text{SK}$</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
<td>$O(mn)$</td>
<td>$(5 + 6mn)$</td>
<td>$(7 + n)m[G]$ + $nm[Z_p]$</td>
</tr>
<tr>
<td>$cTag$</td>
<td>$g$, $v$, $v$, $v$</td>
<td>$g$, $v$, $v$, $v$</td>
<td>$g$, $v$, $v$, $v$</td>
<td>$g$, $v$, $v$, $v$</td>
<td>$g$, $v$, $v$, $v$</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$\ell$</td>
<td>$\ell$</td>
<td>$\ell$</td>
<td>$\ell$</td>
<td>$\ell$</td>
</tr>
</tbody>
</table>

$n$: the maximum number of attributes per ciphertext, $m$: the number of rows of $A$  
$NM$: Non-monotone Access Structure, $M$: Monotone Access Structure

6.1.1 Construction

- **Setup**($\lambda$, $U$) First, the algorithm generates $G$ and $G_T \leftarrow G(\lambda, p)$. Then, it generates $g,v,v_1,v_2,w \in G$ and exponents $a_1,a_2,b,\alpha, h_1,...,h_n \in Z_p$ where $h_1,...,h_n$ are uniquely allocated to each attribute $i \in [1,n]$. Let $\tau_1 = vv_1^{a_1}, \tau_2 = vv_2^{a_2}$. It publishes the public parameters $PK$ as follows

\[(g, g^b, g^{a_1}, g^{a_2}, g^{b-a_1}, g^{b-a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, g^{h_1}, ..., g^{h_n}, e(g, g)^{\alpha a_1-b})\]

It sets $MSK$ as $(g^a, g^{\alpha-a_1}, v, v_1, v_2)$.

- **Encrypt**($PK$, $M$, $S = \{Atti_1,...,Atti_u\}$) The algorithm chooses $s_1, s_2,t,cTag_{Atti_1},... ,cTag_{Atti_u}$ from $Z_p$, and sets $s = s_1 + s_2$. It outputs the ciphertext, $CT$ as follows

\[C = M \cdot (e(g, g)^{\alpha-a_1-b})^{s_2}, C_1 = (g^b)^{s_1}, C_2 = (g^{b-a_1})^{s_1}, C_3 = (g^{a_1})^{s_1},\]

\[C_4 = (g^{b-a_2})^{s_2}, C_5 = (g^{a_2})^{s_2}, C_6 = \tau_1^{s_1} \tau_2^{s_2}, C_7 = (\tau_1^b)^{s_1} (\tau_2^b)^{s_2} w^{-t},\]

\[E_0 = g^l, E_{Atti_i} = (g^{h_{Atti_i}} w^{cTag_{Atti_i}} t)^l, cTag_{Atti_i}, \forall i \in [1,u].\]

- **KeyGen**($MSK$, $PK$, $A = (A, \rho)$) The algorithm chooses random values $r_1, r_2, z_1, z_2$ from $Z_p$, and sets $r = r_1 + r_2$. We let $A_x$ denote the $x^{th}$ row of $A$ and $\rho(x)$ write the attribute associated $A_x$ by the mapping, $\rho$. It randomly chooses $kTag_{\rho(x)}$ for all $x \in [1,m]$ from $Z_p$ where $A$ is an $m \times \ell$ matrix. It randomly selects an $\ell$ sized vector $\bar{\mu}$ of which the first coordinate equals to 1 from $Z_p$, and sets
\[ \lambda_x = r_1 A_x \cdot \bar{\mu} \] as the share of \( A_x \). Then, it creates \( SK \) as follows

\[ D_1 = g^{\alpha_1} v^r, D_2 = g^{-a} v^r g^{z_1}, D_3 = (g^b)^{-z_1}, D_4 = v_2^r g^{z_2}, D_5 = (g^b)^{-z_2}, D_6 = g^{r_2 b}, \]

\[ L_x = g^{\lambda_x}, K_x = (g^{h_\rho(x)} w^{k Tag_\rho(x)})^{\lambda_x}, k Tag_\rho(x) \forall x \in [1, m]. \]

Remark 1. \( k Tag \) is uniquely allocated for each attribute in \( \{ \rho(x); \forall x \in [1, m] \} \), not for each row of \( A \). Therefore, if \( \rho(x_1) \) equals to \( \rho(x_2) \) but \( x_1 \neq x_2 \), then \( k Tag_\rho(x_1) \) equals to \( k Tag_\rho(x_2) \).

- Decrypt \((SK, CT, PK, \Lambda, S)\) First, the algorithm calculates constants \( w_j \) such that \( \sum_{\rho(j) \in S} w_j A_j = (1, 0, \ldots, 0) \). Using \( w_j \), it calculates

\[ W_1 = e(C_1, D_1) e(C_2, D_2) e(C_3, D_3) e(C_4, D_4) e(C_5, D_5), \]

\[ W_2 = e(C_6, D_6) e(C_7, \prod_{\rho(x) \in S} (L_x)^{w_x}). \]

Let \( \Delta_x \) be \( e(K_x, E_0)/e(L_x, E_\rho(x)) \). Then, it calculates

\[ W_3 = \prod_{\rho(x) \in S} (\Delta_x)^{\frac{w_x}{(k Tag_\rho(x) - c Tag_\rho(x))}}. \]

Finally, \( M = C \cdot (W_2/(W_1 \cdot W_3)) \).

Correctness

Since \( \sum_{\rho(j) \in S} w_j A_j = (1, 0, \ldots, 0) \) and the first coordinate of \( \bar{\mu} \) equals to 1,

\[ \prod_{\rho(x) \in S} (L_x)^{w_x} = \prod_{\rho(x) \in S} (g)^{w_x \cdot r_1 A_x \bar{\mu}} = g^{r_1}. \]

Then, the computations of \( W_1 \) and \( W_2 \) are trivial, and appear also in [Wat09]. We only point out that \( W_2/W_1 = e(g, g)^{-\alpha_1 - b s_2} e(g, w)^{r_1 t} \). We provide the rest parts of the correctness to show that our scheme can be correctly decrypted.

\[ e(K_x, E_0) = e((g^{h_\rho(x)} w^{k Tag_\rho(x)} )^{\lambda_x}, g^t) = e(g, g)^{h_\rho(x) \lambda_x t} e(g, w)^{k Tag_\rho(x) \lambda_x t} \]

\[ e(L_x, E_\rho(x)) = e(g^{\lambda_x}, (g^{h_\rho(x)} w^{c Tag_\rho(x)})^t) = e(g, g)^{h_\rho(x) \lambda_x t} e(g, w)^{c Tag_\rho(x) \lambda_x t} \]

Therefore,

\[ (\Delta_x)^{1/(k Tag_\rho(x) - c Tag_\rho(x))} = e(g, w)^{\lambda_x t} \]
Because $\sum_{\rho(x) \in S} w_x \lambda_x = r_1$,

$$\prod_{\rho(x) \in S} (\Delta_x)^{w_x/(kTag_{\rho(x)} - cTag_{\rho(x)})} = \prod_{\rho(x) \in S} e(g, w)^{w_x \lambda_x t} = e(g, w)^{r_1 t}.$$ 

### 6.1.2 Security Analysis

We define two semi-functional algorithms $\text{SFKeyGen}$ and $\text{SFEncrypt}$ which output a semi-functional key and a semi-functional ciphertext, respectively. A semi-functional key can decrypt a normal ciphertext, but cannot decrypt a semi-functional ciphertext although a normal key can decrypt both.

$\text{SFKeyGen}(\text{MSK}, \text{PK}, k = (A, \rho))$ The algorithm takes as input an access structure $(A, \rho)$. Using $\text{KeyGen}$, it generates a normal key

$$(D_1', ..., D_6', \{L'_x, K'_x, kTag'_{\rho(x)}; \forall x \in [1, m]})$$

where $A$ is an $m \times \ell$ access matrix. Then, it sets the elements of a semi-functional key identically with those of the normal key except $D_1, D_2, D_4$. Then, it randomly selects $\gamma$ from $\mathbb{Z}_p$ and defines $D_1, D_2, D_4$ as

$$D_1 = D_1' \cdot g^{-a_1 a_2 \gamma}, D_2 = D_2' \cdot g^{a_2 \gamma}, D_4 = D_4' \cdot g^{a_1 \gamma}.$$ 

$\text{SFEncrypt}(\text{PK}, M, S)$ For a set of attributes $S$, the algorithm generates a normal ciphertext

$$C', C_1', ..., C_7', E_0', \{E'_{\text{Att}_i}, cTag'_{\text{Att}_i}; \forall \text{Att}_i \in S\}$$

using $\text{Encrypt}$. Then, it sets a semi-functional ciphertext identically with those of the normal ciphertext except $C_4, ..., C_7$. To generate $C_4, ..., C_7$, it randomly selects $\kappa$ from $\mathbb{Z}_p$ and sets $C_4, ..., C_7$ as

$$C_4 = C_4' \cdot g^{b_2 a_2 \kappa}, C_5 = C_5' \cdot g^{a_2 \kappa}, C_6 = C_6' v_2^{a_2 \kappa}, C_7 = C_7' v_2^{b_2 a_2 \kappa}.$$ 

$\text{Game}_{\text{Real}}$. This game is identical with the semi-adaptive security model. It should be noted that all keys and the challenge ciphertext are normal in this game.

$\text{Game}_\delta$ This game is identical with $\text{Game}_{\text{Real}}$ except the first $\delta$ keys and the challenge ciphertext. In this game, the first $\delta$ keys and the challenge ciphertext are semi-functional. In other words, $\text{SFKeyGen}$ and $\text{SFEncrypt}$ replace $\text{KeyGen}$ and $\text{Encrypt}$ to create the first $\delta$ keys and the challenge ciphertext. It should be noted that, in $\text{Game}_0$, all keys are normal, but the challenge cipher is semi-functional.
**Game**$_{Final}$ This game is identical with **Game**$_q$ except the message encrypted in the challenge ciphertext where $q$ is the total number of keys queried by the adversary. In this game, the random message replaces the original message of the challenge ciphertext.

**Theorem 6.1.** (Informal) Our KP-ABE scheme in a small universe is semi-adaptively secure under DLIN.

**Proof:** This is proved by Lemmas 6.1, 6.2 and 6.3. □

**Lemma 6.1.** (Semi-functional ciphertext invariance) Suppose there exists a PPT algorithm $A$ to distinguish between **Game**$_{Real}$ and **Game**$_0$ with a non-negligible advantage $\epsilon$. Then we can build an algorithm $B$ breaking DLIN with the advantage, $\epsilon$, using $A$.

**Proof:** This proof is quite similar with that of Waters’ IBE [Wat09] except the structure of the challenge ciphertext, and some elements in public keys. $B$ takes $(g, f, \nu, g^{c_1}, f^{c_2}, T)$ as an instance from DLIN assumption. It will simulate either **Game**$_{Real}$ or **Game**$_0$ based on the value of $T$.

**Setup:** The algorithm selects $a_1, b, y_v, y_v^1, y_v^2, y_w, h_1, ..., h_{nt}$ from $\mathbb{Z}_p$, and sets $g^{a_1} = f, g^{a_2} = \nu$. Then, it publishes the public parameters as follows

\[ g, g^b, g^{b\cdot a_1} = f^b, g^{b\cdot a_2} = \nu^b, w = g^{y_v}, g^{h_1}, ..., g^{h_{nt}}, \]

\[ \tau_1 = g^{y_v} f^{y_v^1}, \tau_2 = g^{y_v} \nu^{y_v^2}, \tau_1^b, \tau_2^b, e(g, g)^{a\cdot a_1 b} = e(g, f)^{a\cdot b}. \]

It also sets MSK = \{ $g^a, g^{a\cdot a_1} = f^a, \nu, v_1, v_2$ \}.

**Init:** Before it generates any private key, $B$ requests to the adversary a target set of attributes $S^*$ which will be used to generate the challenge ciphertext.

**Phase I and II:** To generate normal keys, $B$ uses the key generation algorithm, **KeyGen**. It is possible because $B$ knows all public parameters and MSK.

**Challenge:** When $A$ requests the challenge ciphertext for $S^* = \{ Att_1, ..., Att_u \}$ with two message $M_0$ and $M_1$, it randomly selects $\beta$ from $\{0, 1\}$. Using **Encrypt**, $B$ generates a normal ciphertext,

\[ C', C'_1, ..., C'_7, E_0', \{ E_{Att_i}' , cTag_{Att_i}' ; \forall i \in [1, u] \}. \]

Then, it sets the challenge ciphertext as follows:

\[ C = C' \cdot (e(g^{c_1}, f) \cdot e(g, f^{c_2}))^{b\cdot a} C_1 = C'_1, (g^{c_1})^b, C_2 = C'_2 \cdot (f^{c_2})^{-b}, C_3 = C'_3 \cdot (f^{c_2})^{-1}, \]

\[ C_4 = C'_4(T)^b, C_5 = C'_5 \cdot T, C_6 = C'_6 \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_v^1} \cdot T^{y_v^2}, \]
\[ C_7 = C'_7 \cdot \left( (g^{\tau_1})^{y_v} \cdot (f^{\tau_2})^{-y_v} \cdot T^{y_{w_1}} \right)^b, \]

\[ E_0 = E'_0, \{ E_{\text{Att}_i} = E'_{\text{Att}_i}, cTag_{\text{Att}_i} = cTag'_{\text{Att}_i}; \forall i \in [1, u] \} \]

We let \( s_1, s_2, t \) denote the randomization parameters of the normal challenge ciphertext. This implicitly sets \( s_1 = -s_2 + s_1' \) and \( s_2 = s_2' + s_1 + s_2 \). Therefore, if \( T \) equals to \( n^{-1} + c_2 \), \( B \) has properly simulated \( \text{Game}_{\text{Real}} \). Otherwise, if \( T \) is a random value, it has simulated \( \text{Game}_0 \), also properly by letting \( n^{-1} + c_2 g^c \) denote \( T \). \( \square \)

For semi-functional key invariance of KP-ABE in a small universe, we use the fact if there an access structure \( A = (A, \rho) \) and a set of attributes, \( S^* \) such that \( S^* \) cannot satisfy \( A \), there exists \( \bar{\mu} \) having 1 as the first coordinate and being orthogonal to all rows of \( A \) such that corresponding attributes of the rows of \( A \) by \( \rho \) are also included in \( S^* \). We use \( \bar{\mu} \) as a main tool which unlocks the values of \( \text{tag} \) in the challenge key which is the \( k^{th} \) key in Lemma 6.2. Using this property of \( \bar{\mu} \), if an attribute appears in both the challenge key and the challenge ciphertext, we can set \( \text{tag} \) for this attribute in the challenge key as a random value by nullifying exponents of an incalculable element (\( f^{c_1} \) in Lemma 6.2) of the attribute although we keep \( \text{tag} \) for the attribute in the challenge ciphertext as a value uniquely allocated to each attribute. This makes \( \text{tags} \) in the challenge key and the challenge ciphertext be independent (i.e. they do not correlate to each other), since they represent attributes when it appears in either the private key or the challenge ciphertext, not in both.

**Lemma 6.2. (Semi-functional key invariance)** Suppose there exists a PPT algorithm \( A \) to distinguish between \( \text{Game}_{k-1} \) and \( \text{Game}_k \) with a non-negligible advantage \( \epsilon \). Then we can build an algorithm \( B \) to break DLIN with advantage \( \epsilon \), using \( A \).

**Proof:** First, \( B \) takes in \((g, f, \nu, g^{c_1}, f^{c_2}, T)\) as an instance from DLIN. It will simulate either \( \text{Game}_{k-1} \) or \( \text{Game}_k \) based on the value of \( T \).

**Setup:** The algorithm selects \( \alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, h'_1, \ldots, h'_n, \bar{h}_1, \ldots, \bar{h}_n \) from \( \mathbb{Z}_p \), and sets

\[ g = g, g^b = f, v = \nu^{-a_1} a_2, v_1 = \nu^{a_2} \cdot g^{y_{v_1}}, v_2 = \nu^{a_1} \cdot g^{y_{v_2}}, \]

\[ w = fg^{y_w}, \{ g^{h_i} = f^{-h'_i} g^{\bar{h}_i}; \forall i \in [1, n_i] \} \]

In this setting, we do not know \( h_i \), directly, but we can calculate \( g^{h_i} \) using \( g, f, h'_i \), and \( \bar{h}_i \). Then, it publishes the public parameters following

\[ g, g^b, g^{b-a_1} = f^{a_1}, g^{b-a_2} = f^{a_2}, g^{h_1}, \ldots, g^{h_{n_1}}, v, v_1, v_2, w, \]

\[ \tau_1 = g^{y_{v_1} a_1}, \tau_2 = g^{y_{v_2} a_2}, \tau_1^b = f^{y_{v_1} a_1}, \tau_2^b = f^{y_{v_2} a_2}, e(g, g)^{\alpha - a_1} = e(f, g)^{\alpha_1}. \]
B generates MSK = \{g^a, g^{\alpha a_1}, v, v_1, v_2\}. This is possible because it knows \(a_1\) and \(\alpha\).

We stress that \(B\) does not require any information of the target set of the challenge ciphertext when it sets all parameters in Setup.

**Init:** Before \(B\) generates any private key, it requests to the adversary a target set of attributes \(S^*\) which will be used to generate the challenge ciphertext.

Phase I and II: For the first \(k - 1\) keys (< \(k\), first it generates a normal key \(D_1', ..., D_6', \{L_x', K_x', kTag_{p(x)}\}; \forall x \in [1, m]\)), then it selects \(\gamma\) from \(\mathbb{Z}_p\) and sets

\[
D_1 = D_1' \cdot g^{-a_2 \gamma}, \quad D_2 = D_2' \cdot g^{a_2 \gamma}, \quad D_3 = D_3', \quad D_4 = D_4' \cdot g^{\alpha \gamma}, \quad D_5 = D_5', \quad D_6 = D_6',
\]

\[
L_x = L_x', \quad K_x = K_x', \quad kTag_{p(x)} = kTag_{p(x)}' \quad \forall x \in [1, m]
\]

For the rest keys except the \(k\)-th key (> \(k\)), \(B\) runs the key generation algorithm to generate normal keys. It is computable since \(B\) knows all public parameters and MSK.

To generate the \(k\)-th key for \(A = (A, \rho)\), it first generates tags for the key as follow:

\[
kTag_{p(x)} = h_{\rho(x)}' \quad \forall x \text{ s.t. } \rho(x) \notin S^*
\]

and

\[
kTag_{p(x)} \leftarrow R \quad \forall x \text{ s.t. } \rho(x) \in S^*.
\]

We point out that \(\rho\) is not necessary to be injective, and \(kTag_{p(x)}\) is allocated uniquely to each attribute \(\rho(x)\). Then, using \(kTag_{p(x)}\), it generates a normal key \(D_1', ..., D_6', \{L_x', K_x', kTag_{p(x)}; \forall x \in [1, m]\}\).

We let \(z_1', z_2', r_1', r_2', \vec{\mu}''\) denote randomized parameters of the normal key. Then, \(B\) generates an additional random vector \(\vec{\mu}''\) of which the first coordinate equals to 1, and \(A_x \cdot \vec{\mu}'' = 0\) for all \(x\) such that \(\rho(x) \in S^*\) where \(A_x\) is the \(x\)-th row of \(A\). \(\vec{\mu}''\) exists because \(S^*\) cannot satisfy the access structure. It calculates \(\lambda_x' = c_1 A_x \cdot \vec{\mu}''\), and sets the \(k\)-th key as follows

\[
D_1 = D_1' T^{-a_2}, \quad D_2 = D_2' T^{a_2 (g^{r_1})^{y_{v_1}}}, \quad D_3 = D_3', \quad D_4 = D_4' (f^{c_2})^{y_{v_1}}, \\
D_5 = D_5' (f^{c_2})^{y_{v_2}}, \quad D_6 = D_6' f^{c_2},
\]

\[
\{L_x = L_x' g^{\lambda_x'}, K_x = K_x' (g^{h_{\rho(x)}} w^{kTag_{p(x)}})^{\lambda_x''}, kTag_{p(x)}; \forall x \in [1, m]\}.
\]

This implicitly sets \(z_1 = z_1' - y_{v_1} c_2\) and \(z_2 = z_2' - y_{v_2} c_2\). Also, it sets \(r_2 = r_2' + c_2\) and \(r_1 = r_1' + c_1\). Additionally, this implicitly sets \(\lambda_x = r_1 A_x \cdot \vec{\mu} + c_1 A_x \cdot \vec{\mu}''\). \(\lambda_x\) is the proper share of \(A_x\) since it actually implies that \(\lambda_x = (r_1 + c_1) A_x \vec{\mu}\) where

\[
\vec{\mu} = \left(\frac{r_1 \mu_2' + c_1 \mu_2''}{r_1 + c_1}, \ldots, \frac{r_1 \mu_2' + c_1 \mu_2''}{r_1 + c_1}\right)
\]
CHAPTER 6. KP-ABE WITH SHORT CIPHERTEXTS

if we let \( \mu' \) and \( \mu'' \) denote the \( i \)th coordinate of \( \vec{\mu}' \) and \( \vec{\mu}'' \), respectively. \( \vec{\mu} \) is randomly distributed because of the random vector, \( \vec{\nu}' \).

Therefore, if \( T \) equals to \( \nu^{c_1 + c_2} \), this is a properly distributed normal key. Otherwise, \( T \) is a random value and we denote it as \( \nu^{c_1 + c_2} g^\gamma \), this is the properly distributed semi-functional key.

Challenge: When \( \mathcal{A} \) requests the challenge ciphertext for \( S^* = \{ \text{Att}_1, ..., \text{Att}_n \} \) with two messages, \( M_0 \) and \( M_1 \), it randomly selects \( \beta \) from \( \{0, 1\} \). Then, it sets \( c\text{Tag}_{\text{Att}_i} = y_{h_{\text{Att}_i}}' \forall \text{Att}_i \in S^* \). With \( M_\beta \) and \( c\text{Tag}_{\text{Att}_i} \), it generates a normal challenge ciphertext,

\[
(C', C_1', ..., C_t', E_0, \{ E_{\text{Att}_i}, c\text{Tag}_{\text{Att}_i}; \forall \text{Att}_i \in S^* \}).
\]

Then, it randomly generates \( \kappa \) from \( \mathbb{Z}_p \) and sets

\[
C = C', C_1 = C_1', C_2 = C_2', C_3 = C_3', C_4 = C_4' \cdot f^{a_2 \kappa}, C_5 = C_5' \cdot g^{a_3 \kappa}, C_6 = C_6' \cdot v_2^{a_2 \kappa},
\]

\[
C_7 = C_7' \cdot f^{y_{v_2 \kappa} - a_2 \nu^{a_1 \kappa} y - a_2}, E_0 = E_0' \cdot \nu^{a_1 a_2 \kappa}, E_{\text{Att}_i} = E_{\text{Att}_i}' \cdot \left( \nu^{y_{h_{\text{Att}_i}} + y_{c\text{Tag}_{\text{Att}_i}}} \right)^{a_1 a_2 \kappa}.
\]

This implicitly sets \( g^t = g'^t \cdot \nu^{a_2 \kappa} \) where \( t \) is a randomization parameter of the normal challenge ciphertext. It should be noted that tags in the \( k \)th key and the challenge ciphertext are not correlated because there is no shared value in tags between the \( k \)th key and the challenge ciphertext. Also, all tags are randomly distributed to the adversary since the values of \( y_{h_{\text{ti}}} \) has not been revealed, and generated randomly. Therefore, if \( T \) equals to \( \nu^{c_1 + c_2} \), \( \mathcal{B} \) has properly simulated \( Game_{k-1} \). Otherwise, it has simulated \( Game_k \), also properly. \( \square \)

Lemma 6.3. (Semi-functional Security) Suppose there exists a PPT algorithm \( \mathcal{A} \) to distinguish \( Game_q \) and \( Game_{\text{Final}} \) with a non-negligible advantage \( \epsilon \). Then we can build an algorithm \( \mathcal{B} \) breaking DBDH with the advantage, \( \epsilon \), using \( \mathcal{A} \).

Proof: This proof is similar with that of Waters’ IBE [Wat09]. We modified their proof to be fitted with our construction.

\( \mathcal{B} \) takes \((g, g^{c_1}, g^{c_2}, g^{c_3}, T)\) as an instance from DBDH assumption. It will simulate either \( Game_{\text{Real}} \) or \( Game_{\text{Final}} \) based on the value of \( T \).

Setup: The algorithm selects \( \alpha, b, y_v, y_{v_1}, y_{v_2}, y_w, h_1, ..., h_n \) from \( \mathbb{Z}_p \), and sets

\[
g^{a_2} = g^{c_2}, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}, w = g^{y_w}
\]

Then, it publishes the public parameters as follows

\[
g, g^b, g^{b \alpha_1}, g^{b \alpha_2} = (g^{c_2})^b, w, g^{h_1}, ..., g^{h_n}, \tau_1 = v_1^{\alpha_1}, \tau_2 = (g^{c_2})^{y_{v_2}},
\]

\[
\frac{\tau_1^b}{\tau_2^b}, e(g, g)^{\alpha - a_1 b} = e(g^{c_1}, g^{c_2})^{\alpha_1 b}.
\]
This implicitly sets $\alpha = c_1 \cdot c_2$ and $\sigma_2 = c_2$. In this setting, the simulator does not know $\alpha$. Hence, it cannot calculate $MSK$.

**Init:** Before it generates any private key, $B$ requests to the adversary a target set of attributes $S^*$ which will be used to generate the challenge ciphertext.

**Phase I and II:** To generate a semi-functional key for $A = (A, \rho)$, it randomly generates $z_1, z_2, r_1, r_2; \gamma', \{kTag_{\rho(x)}; \forall x \in [1, m]\}$ from $\mathbb{Z}_p$, and sets $r = r_1 + r_2$ where $A$ is an $m \times \ell$ matrix. $B$ selects randomly $\mu$ from $\mathbb{Z}_p^\ell$ of which the first coordinate equals to 1, and other elements are random values, and sets $\lambda_x = r_1 A_x \mu$. It sets semi-functional key as follows

$$D_1 = (g^{c_2})^{-\gamma' \cdot a_1} v^r, D_2 = (g^{c_2})^{-\gamma'} v_1^r g^{z_1}, D_3 = (g^{b})^{-z_1},$$

$$D_4 = v_2^r g^{z_2} (g^{c_1})^{a_1} g^{a_1 \cdot \gamma'}, D_5 = (g^{b})^{-z_2}, D_6 = g^{z_2 b},$$

$$L_x = g^{\lambda_x}, \{K_x = (g^{h_{\rho(x)} w^{kTag_{\rho(x)}}})^{\lambda_x}, kTag_{\rho(x)}; x \in [1, m]\}$$

This implicitly sets $\gamma = c_1 + \gamma'$.

**Challenge:** When $A$ requests the challenge ciphertext for $S^* = \{\text{Att}_1, ..., \text{Att}_u\}$ with two message $M_0$ and $M_1$, first, $B$ randomly selects $\beta$ from $\{0, 1\}$. It, then, randomly generates $s_1, \kappa, cTag_{\text{Att}_1}, ..., cTag_{\text{Att}_u}$ and $t$ from $\mathbb{Z}_p$ and sets $s_2 = c_3$ sets $\kappa = -c_3 + \kappa'$. It sets the challenge ciphertext as follows

$$C = M_\beta \cdot T^{a_1 \cdot b}, C_1 = g^{b \cdot s_1} (g^{c_3})^{b}, C_2 = g^{b \cdot a_1 \cdot s_1}, C_3 = g^{a_1 s_1}, C_4 = (g^{c_2})^{b \cdot \kappa'},$$

$$C_5 = (g^{c_2})^{\kappa'}, C_6 = \tau_1^{s_1} (g^{c_3})^{\kappa} (g^{c_2})^{\kappa' \cdot \kappa'}, C_7 = \tau_1^{s_1 \cdot b} (g^{c_3})^{\kappa \cdot \kappa'} (g^{c_2})^{\kappa' \cdot \kappa'} w^t, E_0 = g^t, \{E_{\text{Att}_i} = (g^{h_{\text{Att}_i} w^{cTag_{\text{Att}_i}}})^t, cTag_{\text{Att}_i}; i \in [1, u]\}$$

Therefore, if $T$ equals to $g^{a_1 c_2 c_3}$, $B$ has properly simulated $\text{Game}_q$. Otherwise, a random value will be added in $M_\beta$, and it has simulated $\text{Game}_{\text{Final}}$, also properly.

\[\Box\]

### 6.2 KP-ABE with short ciphertexts in a large universe

We introduce KP-ABE with short ciphertexts in a large universe $U$. In this system, the maximum number of attributes per ciphertext is bounded by $n$, but the total number of attributes in the system is not bounded. In this scheme, we achieve short ciphertexts using a formula provided in [ALdP11]. Using the same formula,
interestingly, we prove that our construction is semi-adaptively secure with a large
universe of attributes under a standard assumption via dual system encryption.

6.2.1 Construction

- **Setup**($\lambda$, $U$, $n$) First, $G$ and $G_T \xleftarrow{\$} G(\lambda, p)$. Then, the algorithm generates
  $g, v, v_1, v_2, w, u \in G$ and exponents $a_1, a_2, b, \alpha, h_0, \ldots, h_n \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the public parameters $PK$ as follows:

  $$(g, g^b, g^{a_1}, g^{a_2}, g^{b-a_1}, g^{b-a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, g^{h_0}, \ldots, g^{h_n}, e(g,g)^{a_1-b})$$

  The MSK consists of the following MSK = $(g^{a_1}, g^{a_2}, v, v_1, v_2)$.

- **Encrypt**($PK$, $M$, $S = \{\text{Att}_1, \ldots, \text{Att}_n\}$) the algorithm chooses $s_1, s_2, t$ and $cTag$
  from $\mathbb{Z}_p$ and sets $s = s_1 + s_2$. It creates a ciphertext, $CT$ as follows:

  $$C = M \cdot (e(g,g)^{a_1-b})^{s_1}, C_1 = (g^b)^{s_1}, C_2 = (g^{b-a_1})^{s_1}, C_3 = (g^{a_1})^{s_1}, C_4 = (g^{b-a_2})^{s_2},$$
  $$C_5 = (g^{a_2})^{s_2}, C_6 = \tau_1^{s_1} \tau_2^{s_2}, C_7 = (\tau_1^b) (\tau_2^b)^{s_2} w^{-t},$$
  $$E_0 = g^t, E_1 = ((g^{h_0})^{(b)}(g^{h_1})^{c_1} \ldots (g^{h_n})^{c_n}) e^{w(Tag)} t, cTag$$
  where $c_0, \ldots, c_n$ are coefficients of $y^0, \ldots, y^n$ for $\prod_{\text{Att}_i \in S} (y - \text{Att}_i)$, respectively.

- **KeyGen**($MSK$, $PK$, $A = (A, \rho)$) We let $A_x$ denote the $x^{th}$ row of $A$ and $\rho(x)$
  write an attribute associated $A_x$ by the mapping, $\rho$. The algorithm randomly chooses $r_{1,x}, r_{2,x}, z_{1,x}, z_{2,x}$ from $\mathbb{Z}_p$
  for each $x \in [1, m]$, and sets $r_x = r_{1,x} + r_{2,x}$ where $A$ is an $m \times \ell$
  matrix. It randomly chooses $kTag_{j,x}$ for each $x \in [1, m]$ and $j \in [1, n]$ from $\mathbb{Z}_p$. Then, it randomly selects an $\ell$
  sized vector $\tilde{\mu}$ of which the first coordinate equals to $\alpha$ from $\mathbb{Z}_p^\ell$, and sets $\lambda_x = A_x \tilde{\mu}$ as the share of
  $A_x$. It creates $SK$ as follows:

  $D_{1,x} = g^{h_{1,x}^\rho} v^{r_x}, D_{2,x} = g^{-h_{1,x}^\rho} v^{r_x} g^{z_{1,x}}, D_{3,x} = (g^b)^{z_{1,x}}, D_{4,x} = v^{r_x} g^{z_{2,x}},$
  $D_{5,x} = (g^b)^{-z_{2,x}}, D_{6,x} = g^{r_{2,x} \rho}, D_{7,x} = g^{r_{1,x}},$
  $\{K_{j,x} = (h_{j,x}^{-\rho(x)} w^{kTag_{j,x}})^{r_{1,x}}, kTag_{j,x}; \forall j \in [1, n]\} \forall x \in [1, m]$

- **Decrypt**($SK$, $CT$, $PK$, $A$, $S$) First, the algorithm calculates constants $w_x$ such
  that $\sum_{\rho(x) \in S} w_x A_x = (1, 0, \ldots, 0)$. For each $x \in S$, it calculates

  $$W_{1,x} := e(C_1, D_{1,x}) e(C_2, D_{2,x}) e(C_3, D_{3,x}) \cdot e(C_4, D_{4,x}) e(C_5, D_{5,x}),$$
CHAPTER 6. KP-ABE WITH SHORT CIPHERTEXTS

\[ W_{2,x} := e(C_6, D_{6,x})e(C_7, D_{7,x}). \]

For each attribute, it calculates \( c_0, \ldots, c_n \) by the same manner of Encrypt, then it calculates \( Tag_x := \sum_{j=1}^{n} c_j \cdot kTag_{j,x} - cTag \). If \( Tag_x \neq 0 \),

\[ W_{3,x} = (e(\prod_{j=1}^{n} (K_{j,x})^{c_j}, E_0)/e(D_{7,x}, E_1))^{1/Tag_x} = e(g, w)^{r_1,x}. \]

Otherwise, it aborts. Hence, \( M = C/\prod_{\rho(x) \in S}(W_{3,x}W_{1,x}/W_{2,x})^{w_x} \).

**Correctness**

The computations of \( W_{1,x} \) and \( W_{2,x} \) are similar to \( W_1 \) and \( W_2 \) of our KP-ABE in small universe, and proved also in [Wat09]. It should be noted that \( W_{1,x}/W_{2,x} = e(g, g)^{a_1b_2}e(g, w)^{-r_1,x} \). We only provide the correctness of \( W_{3,x} \).

\[ e(\prod_{j=1}^{n} (K_{j,x})^{c_j}, E_0) \]

\[ = e(\prod_{j=1}^{n} (h_jh_0^{-\rho(x)^j}w^{kTag_{j,x}})^{c_j}g, g) \]

\[ = e(h_1^c \ldots h_n^c, h_0^{-\sum_{j \in [0,n]} c_j \rho(x)^j}w^{\sum_{j \in [0,n]} c_j kTag_{j,x}})g^{r_1,x} \]

\[ = e(h_1^c \ldots h_n^c, g)^{r_1,x}e(h_0^{-\sum_{j \in [0,n]} c_j \rho(x)^j}, g)^{r_1,x} \cdot e(w^{\sum_{j \in [0,n]} c_j kTag_{j,x}})g^{r_1,x} \]

\[ = e(h_1^c \ldots h_n^c, g)^{r_1,x}e(h_0^{-\sum_{j \in [0,n]} c_j \rho(x)^j}, g)^{r_1,x} \cdot e(w, g)^{r_1,x} \]

Also,

\[ e(D_{7,x}, E_1) = e(g^{r_1,x}, (h_0^c \ldots h_n^c w^{cTag})^{l_1}) = e(g, h_0^c \ldots h_n^c)^{r_1,x}e(g, w)^{r_1,x}cTag. \]

Therefore,

\[ e(\prod_{j=1}^{n} (K_{j,x})^{c_j}, E_0)/e(D_{7,x}, E_1) \]

\[ = e(h_0, g)^{-cr_1,x}e(h_0^{-r_1,x} \sum_{j \in [0,n]} c_j \rho(x)^j \cdot e(g, w)^{r_1,x}((\sum_{j \in [0,n]} c_j kTag_{j,x} - cTag)) \]

\[ = e(h_0, g)^{-r_1,x} \sum_{j \in [0,n]} c_j \rho(x)^j \cdot e(g, w)^{r_1,x}((\sum_{j \in [0,n]} c_j kTag_{j,x} - cTag)) \]

\[ = e(g, w)^{r_1,x}((\sum_{j \in [0,n]} c_j kTag_{j,x} - cTag)) \]

The last equality holds because \( \sum_{j \in [0,n]} c_j \rho(x)^j = \sum_{Att_i \in S}(\rho(x) - Att_i) = 0 \) since \( \rho(x) \in S \). \( W_{3,x} = e(g, w)^{r_1,x} \) can be computed since all \( kTags \) and \( cTag \) are given.

Finally,

\[ \prod_{\rho(x) \in S}(W_{3,x}W_{1,x}/W_{2,x})^{w_x} = e(g, g)^{a_1b_2}. \]
because $W_3, x W_1, x / W_2, x = e(g, g)^{\lambda_x a_1 b_2}$, and $\sum_{\rho(x) \in S} w_x \lambda_x = \alpha$.

6.2.2 Security Analysis

We define two semi-functional algorithms $\text{SFKeyGen}$ and $\text{SFEnc}$. We remind that since we prove the security under the semi-adaptive security model of KP-ABE, the simulator always knows the target set $S^*$ for the challenge ciphertext when it creates semi-functional keys.

$\text{SFKeyGen}(\text{MSK}, \text{PK}, S^*, A)$ The algorithm takes as inputs the target set of attribute $S^*$ for the challenge ciphertext and an access structure $A = (A, \rho)$ where $A$ is an $m \times \ell$ matrix. First, to generate the semi-functional key, the algorithm generates a normal key

$$D'_1, x, \ldots, D'_{7}, x, \{ K'_j, x, kTag'_j, x : \forall j \in [1, n] \} \quad \forall x \in [1, m]$$

using $\text{KeyGen}$. Then, it sets

$$D_1, x = D'_1, x, \ldots, D_7, x = D'_7, x,$$

$$\{ K_j, x = K'_j, x, kTag_{j, x} = kTag'_{j, x} : \forall j \in [1, n] \} \quad \forall x \text{ s.t. } \rho(x) \in S^*$$

For the rest key elements, It randomly selects $\gamma_x, \ldots, \gamma_x, \theta$ from $\mathbb{Z}_p$ for each $x$ such that $\rho(x) \notin S^*$ and defines $D_{1, x}, D_{2, x}, D_{4, x}$ as

$$D_{1, x} = D'_1, x \cdot g^{-a_1 a_2 r}, D_{2, x} = D'_2, x \cdot g^{a_2 \gamma_r}, D_4, x = D'_4, x \cdot g^{a_1 \gamma_r} \quad \forall x \text{ s.t. } \rho(x) \notin S^*$$

and sets other elements equal to those of the normal key.

$\text{SFEncrypt}(PK, M, S)$ For a set of attributes $S$, the algorithm generates a normal ciphertext

$$C', C'_1, \ldots, C'_7, E'_0, E'_1, cTag'$$

using $\text{Encrypt}$. Then, it sets a semi-functional ciphertext identically with the normal ciphertext except $C'_4, C'_7$. Then, it randomly selects $\kappa$ from $\mathbb{Z}_p$ and sets $C_4, \ldots, C_7$ as

$$C_4 = C'_4 \cdot g^{a_2 \kappa}, C_5 = C'_5 \cdot g^{a_2 \kappa}, C_6 = C'_6, a_2 \kappa, C_7 = C'_7 b_2 a_2 \kappa.$$

In our security proof, we utilize a hybrid model to convert a normal key to a semi-functional key. Instead of changing a type of the key at once, we change the key elements associated with an attribute which is not included in $S^*$ one-by-one. In order to describe this process, we additionally define a semi-functional key generation algorithm $\text{SFKeyGen}'$. It should be noted that the semi-functional key generation algorithm additionally takes as input an index $\theta$. 
SFKeyGen’(MSK, PK, S*, A, θ)  The algorithm takes as inputs an index θ, the target set of attribute S* for the challenge ciphertext and an access structure A = (A, ρ) where A is an m × ℓ matrix. First, to generate the semi-functional key, the algorithm generates a normal key

\[ D'_1, x, \ldots, D'_{7, x};(K'_j, x, kTag'_j, x; \forall j \in [1, n]) \quad \forall x \in [1, m] \]

using KeyGen. Then, it sets

\[ D_{1, x} = D'_{1, x}; \ldots, D_{7, x} = D'_{7, x}; \]

\[ \{K_j, x = K'_j, x; kTag_j, x = kTag'_j, x; \forall j \in [1, n]\} \quad \forall x \text{ s.t. } \rho(x) \in S^* \]

For the rest key elements, we let \( x_i \) denote the index of the \( i \)th row \( A_x \) of A such that \( \rho(x) \notin S^* \). It randomly selects \( \gamma_{x_1}, \ldots, \gamma_{x_\theta} \) from \( \mathbb{Z}_p \), and defines \( D_{1, x_1}, D_{2, x_1}, D_{4, x_1} \) as

\[
D_{1, x_1} = D'_{1, x_1} \cdot g^{-a_1 g^{a_2 \gamma_{x_1}}}, D_{2, x_1} = D'_{2, x_1} \cdot g^{a_2 \gamma_{x_1}}, D_{4, x_1} = D'_{4, x_1} \cdot g^{a_1 \gamma_{x_1}} \quad \forall x_i \text{ s.t. } i \leq \theta.
\]

Also, it sets other elements equal to those of the normal key.

For the other elements (\( i > \theta \)), it sets,

\[ D_{1, x_i} = D'_{1, x_i}; \ldots, D_{7, x_i} = D'_{7, x_i}; \]

\[ \{K_j, x_i = K'_j, x_i; kTag_j, (x_i) = kTag'_j, (x_i); \forall j \in [1, n]\} \quad \forall x_i \text{ s.t. } i > \theta.
\]

It should be noted that SFKeyGen’(MSK, PK, S*, A, Θ) is identical with SFKeyGen(MSK, PK, S*, A) where Θ is the number of rows, \( A_x \) of A such that \( \rho(x) \notin S^* \).

GameReal This game is identical with the semi-adaptive security model. It should be noted that all keys and the challenge ciphertext are normal in this game.

Game\(_{δ,0}\) is identical with Game\(_{δ−1, Θ_{δ−1}}\) where \( Θ_{δ−1} \) is the number of rows, \( A_x \) of A such that \( ρ(x) \notin S^* \) of the \( δ − 1 \)th key. In this game, the first \( δ − 1 \) keys are generated by SFKeyGen(MSK, PK, S*, A). It should be noted that in Game\(_{0,0}\) all keys are normal, but the challenge ciphertext is semi-functional.

Game\(_{δ,θ}\) We let \( x_i \) denote the index of the \( i \)th row \( A_x \) of A such that \( ρ(x) \notin S^* \) where \( (A, ρ) \) is an access structure for the \( δ \)th key. This game is identical with Game\(_{δ,θ−1}\) except the key elements for \( A_{x_θ} \) of the \( δ \)th key. In this game, the key elements for \( A_{x_1}, \ldots, A_{x_θ} \) are semi-functional. It means SFKeyGen’(MSK, PK, S*, A, θ) are used to generate the \( δ \)th key.

GameFinal This game is identical with Game\(_{q,Θ_q}\) except the message encrypted in the
Table 6.2: Summary of Security Games

<table>
<thead>
<tr>
<th>Game</th>
<th>Key Generation Algorithm</th>
<th>Encryption Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game_Real</td>
<td>KeyGen</td>
<td>Enc</td>
</tr>
<tr>
<td>Game_0,0</td>
<td>KeyGen</td>
<td>SFEnc</td>
</tr>
<tr>
<td>Game_\delta,0</td>
<td>SFKeyGen (&lt; \delta)</td>
<td>SFEnc</td>
</tr>
<tr>
<td>Game_\delta,\theta</td>
<td>SFKeyGen (&lt; \delta)</td>
<td>SFEnc</td>
</tr>
<tr>
<td>Game_\delta,\theta</td>
<td>SFKeyGen’ with \theta (= \delta)</td>
<td>SFEnc</td>
</tr>
<tr>
<td>Game_\delta,\theta</td>
<td>SFKeyGen (&gt; \delta)</td>
<td>SFEnc</td>
</tr>
<tr>
<td>Game_\text{final}</td>
<td>SFKeyGen</td>
<td>SFEnc with a random message</td>
</tr>
</tbody>
</table>

(< \delta): For the first \(\delta - 1\) keys, (= \delta): For the \(\delta\)th key, (> \delta): For all keys except the first \(\delta\) keys

challenge ciphertext where \(q\) is the total number of key queries in Phase I and Phase II. In this game, a random message replaces the message in the challenge ciphertext.

**Theorem 6.2.** *(Informal)* Our KP-ABE scheme with short ciphertexts is semi-adaptively secure under DLIN.

**Proof:** This is proved by Lemmas 6.4, 6.5, and 6.6.

**Lemma 6.4.** *(Semi-functional ciphertext invariance)* Suppose there exists a PPT algorithm \(A\) to distinguish between Game\_Real and Game\_0,0 with a non-negligible advantage \(\epsilon\). Then we can build an algorithm \(B\) breaking DLIN with the advantage, \(\epsilon\), using \(A\).

**Proof:** This proof is similar with the proof of Lemma 6.1. \(B\) takes \((g, f, \nu, g^{c_1}, f^{c_2}, T)\) as an instance from DLIN assumption. It will simulate either Game\_Real or Game\_0,0 based on the value of \(T\).

**Setup:** The algorithm selects \(a_1, b, y_v, y_{v1}, y_{v2}, y_w, h_0, ..., h_n\) from \(\mathbb{Z}_p\), and sets \(g^{a_1} = f, g^{a_2} = \nu\). Then, it publishes the public parameters as follows

\[
g, g^b, g^{b \cdot a_1} = f^b, g^{b \cdot a_2} = (\nu)^b, w = g^{y_w}, g^{h_0}, ..., g^{h_n},
\]

\[
\tau_1 = f^{y_{v1}}, \tau_2 = \nu^{y_{v2}}, \tau_1^b, \tau_2^b, e(g, g)^{\alpha \cdot a_1 \cdot b} = e(g, f)^{\alpha \cdot b}.
\]

It also sets MSK = \(\{g^\alpha, g^{a \cdot a_1} = f^\alpha, v, v_1, v_2\}\).

**Init:** Before it generates any private key, \(B\) requests to the adversary a target set of attributes \(S^*\) which will be used to generate the challenge ciphertext.

**Phase I and II:** To generate normal keys, \(B\) uses the key generation algorithm, KeyGen. It is possible because \(B\) knows all public parameter and MSK.
Challenge: When $A$ requests the challenge ciphertext for $S^* = \{\text{Att}_1, ..., \text{Att}_n\}$ with two messages $M_0$ and $M_1$, $B$ randomly selects $\beta$ from $\{0, 1\}$. Then, it generates a normal ciphertext, $C', C'_1, ..., C'_n, E'_0, E'_2, c\text{Tag}'$ using $\text{Encrypt}$. It sets the challenge ciphertext as follows:

$$C = C' \cdot \left( e(g^{c_1}, f) \cdot e(g, f^{c_2}) \right)^{b_a} C_1 = C'_{1}, (g^{c_1})^b, C_2 = C'_{2} \cdot (f^{c_2})^{-b}, C_3 = C'_{3} \cdot (f^{c_2}),$$

$$C_4 = C'_{4}(T)^b, C_5 = C'_{5} \cdot T, C_6 = C'_{6} \cdot (g^{c_1})^{y_0} \cdot (f^{c_2})^{-y_1} \cdot T^{y_2},$$

$$C_7 = C'_{7} \cdot ((g^{c_1})^{y_0} \cdot (f^{c_2})^{-y_1} \cdot T^{y_2})^b, E_0 = E'_0, E_1 = E'_1, c\text{Tag} = c\text{Tag}'$$

We let $s'_1, s'_2, \tau$ denote the randomization parameters of the normal challenge ciphertext. This implicitly sets $s_1 = -c_2 + s'_1$ and $s_2 = s'_2 + c_1 + c_2$. Therefore, if $T$ equals to $\nu^{c_1+c_2}$, $B$ has properly simulated $\text{Game}_{\text{Real}}$. Otherwise, if $T$ is a random value, it has simulated $\text{Game}_{0,0}$, also properly by letting $\nu^{c_1+c_2} g^\tau$ denote $T$. 

To achieve semi-functional key invariance, we designed a nested duals system encryption. In our proof of lemma 6.5, the independence of tags in the challenge key and the challenge ciphertext is proved by $n$-wise independence [AL10]. In detail, tags for an attribute $\rho(x)$ of $A_x$, $k\text{Tag}_{j, \rho(x)} \forall j \in [1, n]$ and a tag for the challenge ciphertext, $c\text{Tag}$ are generated as

$$
\begin{pmatrix}
-\rho(x) & 1 \\
-\rho(x)^2 & 1 \\
\vdots & \ddots \\
-\rho(x)^n & 1 \\
c_0 & c_1 & c_2 & \cdots & c_n
\end{pmatrix}
\begin{pmatrix}
h'_0 \\
h'_1 \\
\vdots \\
h'_n
\end{pmatrix}
= 
\begin{pmatrix}
k\text{Tag}_{1, \rho(x)} \\
k\text{Tag}_{2, \rho(x)} \\
\vdots \\
k\text{Tag}_{n, \rho(x)}
\end{pmatrix}
\begin{pmatrix}
h_0' \\
h_1' \\
\vdots \\
h_n'
\end{pmatrix}
$$

where $c_j$ is coefficients of $y^j$ of $\prod_{\rho(x) \in S^*} (y - \rho(x))$ and $S^*$ is the target set of attributes for the challenge ciphertext.

To claim $n$-wise independence of tags, we show that they satisfy two conditions following

1. $h'_0, ..., h'_n$ are information theoretically hidden to the adversary.
2. $\text{Att}_i$ is not in $S^*$.

In lemma 6.5, we must show they appear only to suffice key elements for $\rho(x)$ for the first condition. To do this, we isolate $k\text{Tag}_{j, \text{Att}_i} \forall j \in [1, n]$ for $\rho(x)$ from the key elements for other attributes by utilizing a hybrid model. In the security proof, we show the invariance of two games which have different types of key elements for an attribute $\rho(x)$ in the $k^{th}$ key, not for all elements in the key. Also, we do not apply this proof for the elements for attributes shared between the key and the challenge.
ciphertext. Therefore, we leave those key elements as normal type (i.e. without any change of type from the real game) since the correlation can be detected to the adversary by checking \( \sum_{i \in [0,n]} c_i kTag_i, \rho(x) = cTag \) if \( \rho(x) \) is in \( S^* \) as the second condition requires.

**Lemma 6.5. (Semi-functional key invariance)** Suppose there exists a PPT algorithm \( A \) to distinguish \( Game_{k,\theta - 1} \) and \( Game_{k,\theta} \) with a non-negligible advantage \( \epsilon \). Then we can build an algorithm \( B \) breaking DLIN with the advantage, \( \epsilon \), using \( A \).

**Proof:** First, \( B \) takes \( (g, f, \nu, g^{\alpha_1}, f^{v_2}, T) \) as an instance from DLIN. It will simulate either \( Game_{k,\theta - 1} \) or \( Game_{k,\theta} \) based on the value of \( T \).

**Setup:** The algorithm selects \( \alpha, \alpha_1, \alpha_2, y_{v_1}, y_{v_2}, y_w, h_0', ..., h_n', \tilde{h}_0', ..., \tilde{h}_n \) from \( \mathbb{Z}_p \), and sets
\[
\begin{align*}
g &= g, \
g^b &= f, \nu = \nu^{-a_1 - a_2}, v_1 = \nu a_2 \cdot g^{y_{v_1}}, v_2 = \nu a_1 \cdot g^{y_{v_2}}, \
w &= fg^{\nu w}, h_0 = f^{-y_{h_0}} g^{\tilde{y}_{h_0}}, ..., h_n = f^{-y_{h_n}} g^{\tilde{y}_{h_n}}
\end{align*}
\]
This implies that we do not know \( y_{h_i} \), but we can calculate \( h_i \) using \( g, f, y_{h_i}, \) and \( \tilde{y}_{h_i} \). Then it publishes the public parameter following
\[
\begin{align*}
g, g^b, g^{b-a_1} &= f a_1, g^{b-a_2} = f a_2, g^{h_0}, ..., g^{h_n}, w, \
\tau_1 &= g^{y_{v_1} a_1}, \tau_2 = g^{y_{v_2} a_2}, \tau_1^b = f^{y_{v_1} a_1}, \tau_2^b = f^{y_{v_2} a_2}, e(g, g)^{\alpha a_1 b} = e(f, g)^{\alpha a_1}.
\end{align*}
\]

**B** generates \( MSK = \{ g^\alpha, g^{\alpha a_1}, v, v_1, v_2 \} \). This is possible because it knows \( a_1 \) and \( \alpha \). We stress that \( B \) does not require any information of the target set of attributes, \( S^* \) for the challenge ciphertext when it sets all parameters in **Setup**.

**Init:** Before it generates any private key, \( B \) requests to the adversary a target set of attributes \( S^* \) which will be used to generate the challenge ciphertext.

**Phase I and II:** For the first \( k - 1 \) keys (< \( k \)), first it generates a normal key \((D_{1,x}, ..., D_{t,x}, \{ K'_{j,x}, kTag'_{j,x}; \forall j \in [1,n] \}; \forall x \in [1,m]) \) where \( m \) is the number of rows of an access matrix for the key, then, it selects \( \gamma_x \) from \( \mathbb{Z}_p \) for each \( x \) such that \( \rho(x) \notin S^* \). For all rows of \( A \), it sets

1. \( \forall x \text{ s.t. } \rho(x) \in S^* \)
\[
D_{1,x} = D'_{1,x}, D_{2,x} = D'_{2,x}, D_{3,x} = D'_{3,x}, D_{4,x} = D'_{4,x}, D_{5,x} = D'_{5,x}, D_{6,x} = D'_{6,x}, \\
D_{7,x} = D'_{7,x}, \{ K_{j,x} = K'_{j,x}, kTag_{j,x} = kTag'_{j,x}; \forall j \in [1,n] \}
\]
CHAPTER 6. KP-ABE WITH SHORT CIPHERTEXTS

2. \( \forall x \text{ s.t. } \rho(x) \notin S^* \)

\[
D_{1,x} = D'_{1,x} \cdot g^{-a_1 a_2 \gamma_x}, D_{2,x} = D'_{2,x} \cdot g^{a_2 \gamma_x}, D_{3,x} = D'_{3,x}, D_{4,x} = D'_{4,x} \cdot g^{a_1 \gamma_x},
\]

\[
D_{5,x} = D'_{5,x}, D_{6,x} = D'_{6,x}, D_{7,x} = D'_{7,x},
\]

\[
\{K_{j,x} = K'_{j,x}, kTag_{j,x} = kTag'_{j,x}; \forall j \in [1, n]\}).
\]

For the rest keys except the \( k^\text{th} \) key (\( > k \)), \( B \) runs the key generation algorithm to generate normal keys. It is possible since \( B \) knows all public parameters and MSK.

We let \( x_i \) denote the index of the \( i^\text{th} \) row \( A_x \) of \( A \) such that \( \rho(x) \notin S^* \). To generate the \( k^\text{th} \) key with the index \( \theta \), for \( A = (A, \rho) \), it first generates \( \{kTag_{j,x}; \forall j \in [1, n]\} \) randomly from \( \mathbb{Z}_p \) for each \( x \text{ s.t. } x \neq x_\theta \). For \( x_\theta \), it sets

\[
kTag_{j,x_\theta} = y'_{h_j} + y_{h_\theta} \rho(x_\theta)^j \quad \forall j \in [1, n].
\]

Then, using \( \{kTag_{j,x}; \forall j \in [1, n], \forall x \in [1, m]\} \), it generates a normal key

\[
D'_{1,x}, \ldots, D'_{7,x}, \{K'_{j,x}, kTag_{j,x}; \forall j \in [1, n]\} \quad \forall x \in [1, m].
\]

Finally, it selects \( \gamma_{x_i} \) from \( \mathbb{Z}_p \) for each \( x_i \) such that \( i < \theta \), and sets the \( k^\text{th} \) key as follows

1. \( \forall x \text{ s.t. } \rho(x) \in S^* \),

\[
D_{1,x} = D'_{1,x}, \ldots, D_{7,x} = D'_{7,x}, \{K_{j,x} = K'_{j,x}, kTag_{j,x}; \forall j \in [1, n]\},
\]

2. \( \forall x \text{ s.t. } \rho(x) \notin S^* \),

(a) \( \forall x_i \text{ s.t. } i < \theta \),

\[
D_{1,x_i} = D'_{1,x_i} \cdot g^{-a_1 a_2 \gamma_{x_i}}, D_{2,x_i} = D'_{2,x_i} \cdot g^{a_2 \gamma_{x_i}}, D_{3,x_i} = D'_{3,x_i},
\]

\[
D_{4,x_i} = D'_{4,x_i} \cdot g^{a_1 \gamma_{x_i}}, D_{5,x_i} = D'_{5,x_i}, \ldots, D_{7,x_i} = D'_{7,x_i},
\]

\[
\{K_{j,x_i} = K'_{j,x_i}, kTag_{j,x_i}; \forall j \in [1, n]\}).
\]

(b) for \( x_\theta \),

\[
D_{1,x_\theta} = D'_{1,x_\theta} T^{-a_1 a_2}, D_{2,x_\theta} = D'_{2,x_\theta} T^{a_2} (g^{c_1})^{y_{c_1}}, D_{3,x_\theta} = D'_{3,x_\theta} (f^{c_2})^{y_{c_1}},
\]

\[
D_{4,x_\theta} = D'_{4,x_\theta} T^{a_1} (g^{c_1})^{y_{c_2}}, D_{5,x_\theta} = D'_{5,x_\theta} (f^{c_2})^{y_{c_2}}, D_{6,x_\theta} = D'_{6,x_\theta} f^{c_2},
\]

\[
D_{7,x_\theta} = D'_{7,x_\theta} (g^{c_1}), \{K_{j,x_\theta} = K'_{j,x_\theta} (g^{\theta - h_\theta \rho(x_\theta)})^j, kTag_{j,x_\theta}; \forall j \in [1, n]\}),
\]
(c) \( \forall x_i \) s.t. \( i > \theta \),

\[
D_{1,x_i} = D'_{1,x_i}, \ldots, D_{7,x_i} = D'_{7,x_i}, \{K_{j,x_i} = K'_{j,x_i}, kTag_{j,x_i}; \forall j \in [1,n]\}
\]

We let \( z'_{1,x_0}, z'_{2,x_0}, r'_{1,x_0}, r'_{2,x_0}, \mu' \) denote randomized parameters for \( A_{x_0} \) in the normal key. This implicitly sets \( z_{1,x_0} = z'_{1,x_0} - y_{c_2} \) and \( z_{2,x_0} = z'_{2,x_0} - y_{c_2} \). Also, it sets \( r_{2,x_0} = r'_{2,x_0} + c_2 \) and \( r_{1,x_0} = r'_{1,x_0} + c_1 \). If \( T \) equals to \( \nu^{c_1+c_2} \), this has properly simulated the semi-functional key generated by \( S\text{fKeyGen}'(\text{MSK, PK, } S^*, \beta, \theta - 1) \). Otherwise, If \( T \) is a random value, and we denote it as \( \nu^{c_1+c_2}g^{y_{c_2}} \), this is properly simulated the semi-functional key generated by \( S\text{fKeyGen}'(\text{MSK, PK, } S^*, \beta, \theta) \).

Challenge: When \( A \) requests the challenge ciphertext for \( S^* \) with two message, \( M_0 \) and \( M_1 \) \( B \) randomly selects \( \beta \) from \( \{0, 1\} \). Then, it sets \( cTag = c_0h_0' + \ldots + c_nh_n' \). With \( M_\beta \) and \( cTag \), it generates a normal challenge ciphertext,

\[
(C', C'_1, \ldots, C'_7, E_0, E_{\text{Att}_i}; \forall \text{Att}_i \in S^*)
\]

Then, it randomly generates \( \kappa \) from \( \mathbb{Z}_p \) and sets

\[
C = C', C_1 = C'_1, C_2 = C'_2, C_3 = C'_3, C_4 = C'_4 \cdot \nu^{a_2}, C_5 = C'_5 \cdot g^{a_2},
\]

\[
C_6 = C'_6 \cdot \nu^{a_2}, C_7 = C'_7 \cdot \nu^{y_{c_2+p} \cdot \nu^{a_1} \cdot \nu^{y_{c_2}}, E_1 = E'_1 \cdot (\nu^{c_0h_0' + \ldots + c_nh_n' + cTag}) \cdot \nu^{a_2}, cTag.
\]

This implicitly sets \( g' = g'^{y_{c_2}} \cdot \nu^{a_2} \) where \( t' \) is a randomization parameter of a normal key. It should be noted that tags in the challenge ciphertext and the key elements of \( A_{x_0} \) of the \( k \)th key do not correlate to each other because of \( n \)-wise independence. Hence, all tags in the challenge ciphertext and the \( k \)th key are randomly distributed and do not correlate to each other. Therefore, if \( T \) equals to \( \nu^{c_1+c_2} \), \( B \) has properly simulated \( \text{Game}_{k,\theta-1} \). Otherwise, it has simulated \( \text{Game}_{k,\theta} \) also properly.

\[ \square \]

Lemma 6.6. (Semi-functional Security) Suppose there exists a PPT algorithm \( A \) to distinguish \( \text{Game}_{q,\theta} \) and \( \text{Game}_{\text{Final}} \) with a non-negligible advantage \( \epsilon \). Then, we can build an algorithm \( B \) breaking \( \text{DBDH} \) with the advantage, \( \epsilon \), using \( A \).

Proof: \( B \) takes \( (g, g^{c_1}, g^{c_2}, g^{c_3}, T) \) as an instance from \( \text{DBDH} \). It will simulate either \( \text{Game}_{q,\theta} \) or \( \text{Game}_{\text{Final}} \) based on the value of \( T \).

Setup: The algorithm selects \( a_1, b, y_0, y_1, y_2, y_w, h_0, \ldots, h_n \) from \( \mathbb{Z}_p \), and sets

\[
g = g, g^{a_2} = g^{c_2}, v = g^{y_w}, v_1 = g^{y_1}, v_2 = g^{y_2}, w = g^{y_w}, \{h_i = g^{y_i}; i \in [0,n]\}
\]
Then, it publishes the public parameters as follows
\[
g, g^b, g^{b_{a1}}, g^{b_{a2}} = (g^{c_2})^b, w, g^{h_1}, ..., g^{h_n}, \tau_1 = v_1^{a_1},
\]
\[
\tau_2 = (g^{c_2})^{\mu_{12}}, r_1^b, \tau_2^b, e(g, g)^{\alpha a_1 b} = e(g^{c_1}, g^{c_2})^{a_1 b}.
\]
This implicitly sets \(\alpha = c_1 \cdot c_2\) and \(a_2 = c_2\). In this setting, the simulator does not know \(MSK\) since it does not know \(\alpha\).

**Init:** Before it generates any private key, \(B\) requests to the adversary a target set of attributes \(S^*\) which will be used to generate the challenge ciphertext.

**Phase I and II:** For generating a semi-functional key, \(B\) randomly selects \(\bar{\mu}_1\) from \(\mathbb{Z}_p\) such that \(A_x \cdot \bar{\mu}_1 = 0\) for all \(x\) such that \(\rho(x) \in S^*\) and the first coordinate of \(\bar{\mu}_1\) equals to 1. This exists because \(S^*\) does not satisfy an access structure of the semi-functional key. Also, it generate a random vector \(\bar{\mu}_2\) of which the first coordinate equals to 0. It implicitly sets \(\alpha \cdot \bar{\mu} = \alpha \cdot \bar{\mu}_1 + \bar{\mu}_2\). For each \(x \in [1, m]\), it randomly generates \(z_{1,x}, z_{2,x}, r_{1,x}, r_{2,x}, \{kTag_{j,x}; \forall j \in [1, n]\}\) from \(\mathbb{Z}_p\).

Then, for normal type rows, it sets
\[
D_{1,x} = g^{\bar{A}_x \mu_2 a_{12} v_1 r_{1,x}}, D_{2,x} = g^{-A_x \mu_2 v_1 r_{1,x} g^{z_{1,x}}}, D_{3,x} = (g^b)^{-z_{1,x}}, D_{4,x} = v_2^{r_{2,x}} g^{z_{2,x}},
\]
\[
D_{5,x} = (g^b)^{-z_{2,x}}, D_{6,x} = g^{r_{2,x} b}, D_{7,x} = g^{r_{1,x}},
\]
\[
\{K_{j,x} = (h_j h_0^{-\rho(x)})^{w^{kTag_{j,x}}} r_{1,x}, kTag_{j,x}; j \in [1, n]\} \quad \forall x \text{ s.t. } \rho(x) \in S^*
\]

For the rest semi-functional rows, it randomly generate \(\gamma_k'\) for each \(x\) s.t. \(\rho(x) \notin S^*\). It sets
\[
D_{1,x} = g^{A_x \mu_2 a_{12} (g^{c_2})^{-\gamma_k' a_{12} v_1 r_{1,x}}}, D_{2,x} = g^{-A_x \mu_2 (g^{c_2})^{-\gamma_k' v_1 r_{1,x} g^{z_{1,x}}}}, D_{3,x} = (g^b)^{-z_{1,x}},
\]
\[
D_{4,x} = v_2^{r_{2,x}} g^{z_{2,x}} (g^{c_1})^{a_1 A_x \mu_1} g^{a_1 \gamma_k'}, D_{5,x} = (g^b)^{-z_{2,x}}, D_{6,x} = g^{r_{2,x} b}, D_{7,x} = g^{r_{1,x}},
\]
\[
\{K_{j,x} = (h_j h_0^{-\rho(x)})^{w^{kTag_{j,x}}} r_{1,x}, kTag_{j,x}; j \in [1, n]\} \forall x \text{ s.t. } \rho(x) \notin S^*
\]
This implicitly sets \(\gamma_x = c_1 A_x \mu_1 + \gamma_k'\).

**Challenge:** When \(A\) requests the challenge ciphertext for \(S^*\) with two message \(M_0\) and \(M_1\), first, \(B\) randomly selects \(\beta\) from \(\{0, 1\}\). It, then, randomly generates \(s_1, \kappa\) and \(t\) from \(\mathbb{Z}_p\) and sets \(s_2 = c_3\) and \(\kappa = -c_3 + \kappa'\). It, also, sets the challenge ciphertext as follow.
\[
C = M_2 T^{a_{12}}, C_1 = g^{b_{s_1}} (g^{c_3})^b, C_2 = g^{b_{s_1} s_1}, C_3 = g^{a_{12} s_1},
\]
\[
C_4 = (g^{c_2})^{b \cdot \kappa}, C_5 = (g^{c_2})^{\kappa}, C_6 = \tau_1^{s_1} (g^{c_3})^{\mu_{22}} (g^{c_2})^{b_{s_2} \cdot \kappa'},
\]
If $T$ equals to $g_{c_1}g_{c_2}g_{c_3}$, $\mathcal{B}$ has properly simulated Game$_{q,\Theta_q}$. Otherwise, a random will be added in $M_\beta$, and it has simulated Game$_{Final}$, also properly. \hfill \Box
Chapter 7

Conclusion and Future Works

In this work, we introduce techniques to achieve functional encryption schemes using the encoding frameworks. We explore encoding frameworks and contribute to them by suggesting new techniques which results in a number of new adaptively secure functional encryption schemes. Also, a new encoding framework was introduced for efficient functional encryption schemes.

In particular, in Chapter 3, we introduced a new conversion technique which results in more efficient dual schemes of functional encryption schemes for which the adaptive security is proved from the doubly selective security. To clarify the requirements and techniques of this conversion, we utilize the pair encoding framework. Our conversion technique is applicable to all functional encryption schemes. This is proved via doubly selective security and improves the efficiency of the previous best conversion technique.

In Chapter 4, we introduced a new compiler of the pair encoding framework. This compiler is adaptively secure in prime order groups under static assumptions. Prior to our work, all previous works were in composite order groups or do not support encoding schemes which utilize the doubly selective security. Therefore, our compiler improves the efficiency since it realizes functional encryption schemes in prime order groups which are originally introduced in composite order groups. In order to achieve this work, we nested the dual system encryption and generalize the functional encryption from identity based encryption.

In Chapter 5, we presented a new encoding framework which is tag-based encoding. A tag-based encoding results in functional encryption schemes which are adaptively secure in prime order groups. Also, their compilers improve the efficiency of the most efficient compiler of the pair encoding scheme. Therefore, the resulting schemes of our framework are more efficient than those of the pair encoding schemes if their predicates are large. As instances of our new encoding framework, we introduce a number of functional encryption schemes which include inner product encryption and spatial encryption.
In Chapter 6, we proposed two KP-ABE schemes which are semi-adaptively secure. Although those schemes are semi-adaptively secure, they have desirable properties because they allow multi-use of attributes in standard model under the standard assumption with a good efficiency. Particularly, our KP-ABE scheme with short ciphertexts allows a large universe of attributes. Also, the technique for our schemes is related with our tag-based encoding in chapter 5. It shows how the limitation of tag based encoding is easy under the weaker security model.

Through this work, we presented the core technique that modern functional encryption shares and how we improve the previous achievements by utilizing encodings which formalized Waters’ dual system encryption and/or Lewko and Waters’ doubly selective security.

Future Work

All encoding frameworks we presented in this work are equipped with bilinear maps, but there are also functional encryption schemes utilizing multi-linear maps. Since multilinear maps allow functional encryption schemes having more desirable functions such as functional encryption for arithmetic circuits and a homomorphic encryption [BGG+14, Att14b], and multilinear maps become more practical [CLT15, AFH+16, GGH13], it may be interesting to observe the properties of functional encryption schemes in multilinear maps and to achieve similar results like encodings.

In a bilinear map, achieving functional encryption schemes in prime order groups under simple assumptions are actively researched. There introduced versatile tools such as dual system groups [CW13, AC16, Att15] and dual pairing vector spaces [OT09, OT10], but achieving adaptively secure functional encryption in prime order groups with a more plausible property such as allowing multi-use of attributes, better expressiveness (e.g. circuit) or efficient parameters (e.g short ciphertexts) under standard assumption still remains a difficult task.

Tighter security for functional encryption is also interesting topic. The dual system encryption naturally permits a loss in reduction since the advantage of the adversary linearly increases with the number of key queried by the adversary. Improving this disadvantage of the dual system encryption is an interesting future work. There already exist tightly secure identity based encryption schemes [CW13, BKP14, HKS15] and an encoding approach [AHY15]. However, it is unclear whether this encoding is applicable for more complicated functions such as Inner Product Encryption or ABE for tighter security. Introducing a new encoding or instances for tighter security is interesting as complement the previous encoding frameworks.
The recent works show that functional encryption can be reduced in Learning With Error (LWE) problem [BV16, GVW15, AFV11]. Schemes based on LWE problem [Reg09] provide a long-term security since the LWE problem is known to be difficult even under quantum computing. Also, these can provide a better expressiveness such as circuit and homomorphism which are considered to be very difficult in pairing based cryptography. Therefore, functional encryption with LWE may be one of promising directions of functional encryption.
Bibliography


Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Catalano et al. [CFGN11], pages 90–108.


[BW06a] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Juels et al. [JWdV06], pages 211–220.


[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Juels et al. [JWdV06], pages 89–98.


Appendix A

A.1 A Dedicated Construction for Masked Forms

We provide a dedicated construction of Dual-P($R$) which was introduced in chapter 3. The construction is derived by applying a masked form of a pair encoding $P(R)$ to the generic construction in chapter 4. For this construction, we assume that there exists a masked form of $P(R)$ which consists of $\vec{k}(\alpha', x, \vec{h}; \vec{r})$ and $\vec{c}(y, \vec{h}; s, \vec{s})$. Therefore, $P(R)$ can be parsed as $(u, \vec{dk}(\phi u, x, (h_1, ..., h_n); \vec{r}))$ and $(\phi s, \vec{dc}(y, (h_1, ..., h_n); s, \vec{s}))$ respectively when we set $\alpha' = 0$. It should be noted that the way of denoting common value in chapter 4 is slightly different from the syntax of the original pair encoding framework [Att14a]. We follow the original notation of [Att14a] since this difference is only needed for the security proof.

With the masked form of $P(R)$, a functional encryption Dual-FE$(P)$ comprises of four randomized algorithms employing prime order groups $G_1, G_2$ and $G_T$ of order $p$. In the construction, we use the subscripts of group elements to denote the group generators to be used to generate those elements (e.g. $g_1, f_1 \in G_1$ and $g_2, f_2 \in G_2$).

- **Setup($\lambda$) $\rightarrow$ PK, MSK**: The setup algorithm selects a bilinear group $G_1, G_2, G_T$ of order $p$. The algorithm randomly selects $g_1 \in G_1$ and $g_2 \in G_2$. Then, it runs $\text{Param}(\kappa)$ to derive $n$ where $\kappa$ is the index allocated to the function $R$. It randomly generates $\alpha, a, b, y_u, y_v, h_1, ..., h_n \in \mathbb{Z}_p$ and sets $\tau = y_v + a \cdot y_u$ and $\vec{h} = (\phi, h_1, ..., h_n)$. It publishes public parameters as

$$\{g_1, g_1^\vec{h}, g_1^a, g_1^{a \cdot \vec{h}}, g_1^{\tau \cdot \vec{h}}, e(g_1, g_2)^\alpha\}.$$  

It also sets MSK as $\{g_2, g_2^a, g_2^{\vec{h}}, v_2 = g_2^{b y_v}, u_2 = g_2^{b y_u}, f_2 = g_2^b\}$.

- **Encrypt($M, x, PK$) $\rightarrow$ CT$_y$**: The encryption algorithm randomly chooses $r = (u, \vec{r}) \in \mathcal{R}_r$ and runs $\text{Enc}$ to get $\vec{k}(0, x, \vec{h}; \vec{r})$. Then it parse $\vec{k}(0, x, \vec{h}; \vec{r})$ as $u$ and $\vec{dk}(\phi u, x, (h_1, ..., h_n); \vec{r})$. It sets the ciphertext as

$$C = M \cdot e(g_1, g_2)^{\alpha u}, \quad \vec{C}_0 = (g_1^u, g_1^{\vec{dk}(\phi u, x, (h_1, ..., h_n); \vec{r})}),$$

\[118\]
\[
\vec{C}_1 = (g_1^{\alpha u}g_1^{\phi u_x(h_1,\ldots,h_n);\vec{r}}), \quad \vec{C}_2 = (g_1^{\tau u}g_1^{\tau\phi u_x(h_1,\ldots,h_n);\vec{r}}).
\]

- **KeyGen**\((y, MSK, PK) \rightarrow SK_x\): The key generation algorithm chooses a random vector \(s, \vec{s} \in D \times \mathbb{Z}_p \times \mathbb{Z}_p \times R_s\) and runs Enc2 to get \(c(y, \vec{h}; s, \vec{s})\). Then, it parses \(c(y, \vec{h}; s, \vec{s})\) as \(\phi s\) and \(\vec{d}c(y, (h_1, \ldots, h_n); s, \vec{s})\) and randomly selects \(\vec{z} \in \mathbb{Z}_c^{\vec{z}}\) where \(c_m\) is \(|\vec{c}|\). Finally, it generates the private key following

\[
\vec{K}_0 = (g_2^\alpha, g_2^{\phi s}, g_2^{d\vec{c}(y, (h_1, \ldots, h_n); s, \vec{s})})u_2^{\vec{z}}, \quad \vec{K}_1 = u_2^{\vec{z}}, \quad \vec{K}_2 = f_2^{-\vec{z}}.
\]

- **Decrypt**\((PK, x, y, SK_x, CT_y) \rightarrow M\) If \(R(x, y) = 1\), the algorithm computes a reconstruction matrix \(M_{xy}\) such that

\[
(\alpha + \phi s, \vec{d}c(y, \vec{h}; s, \vec{s}))M_{xy}(u, \vec{d}k(\phi u, x, \vec{h}; \vec{r})) = \alpha u.
\]

The decryption algorithm computes

\[
e(g_1, g_2)^{\alpha u} = e(C_0^{M_{xy}}, \vec{K}_0)e(C_1^{M_{xy}}, \vec{K}_1)e(C_2^{M_{xy}}, \vec{K}_2).
\]

Finally, the message can be recovered as \(C/e(g_1, g_2)^{\alpha u}\).

**Correctness** We already prove that a converted encoding using a masked form of a pair encoding scheme is also another pair encoding scheme. Therefore, correctness holds as it is shown in chapter 4.

---

\(^aM_{xy}\) exists since there exists a reconstruction matrix \(M'_{xy}\) of the original pair encoding scheme which satisfies \((\phi s, \vec{d}c(y, \vec{h}; s, \vec{s}))M'_{xy}(u, \vec{d}k(\phi u, x, \vec{h}; \vec{r})) = c(y, \vec{h}; s, \vec{s})M'_{xy}\).

Since \((\alpha + \phi s, \vec{d}c(y, \vec{h}; s, \vec{s}))M_{xy}(u, \vec{d}k(\phi u, x, \vec{h}; \vec{r}))\) is a component-wise operation, \(M_{xy}\) can be derived by rescaling \(M'_{xy}\).