Contributions to functional encryption and its applications

Tran Viet Xuan Phuong
University of Wollongong

COPYRIGHT WARNING

UNIVERSITY OF WOLLONGONG

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author.

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Recommended Citation

Contributions to Functional Encryption and Its Applications

Tran Viet Xuan Phuong

Supervisors:
Professor Willy Susilo & Doctor Guomin Yang

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

Doctorate Degree

The University of Wollongong
School of Computing and Information Technology

August 2016
Declaration

I, Tran Viet Xuan Phuong, declare that this thesis submitted in partial fulfilment of the requirements for the conferral of the degree Doctorate Degree, 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.

Tran Viet Xuan Phuong
January 9, 2017
Abstract

Access control plays an important role in many information systems. Embedding policy-based access control into modern encryption schemes is an interesting but challenging task that has been intensively studied by the cryptographic research community in recent years. Furthermore, most of encryption schemes require not only the guarantee of security, but also the efficiency in terms of computational and communication cost when producing ciphertext and secret key.

In this thesis, we study Functional Encryption comprising its subclasses such as Attribute Based Encryption, Hidden Vector Encryption, and Inner Product Encryption. We boost the advantage of these encryption schemes by improving their performance, which is critical for real applications. We also consider the user anonymity in these encryption systems in order to protect user privacy, which is very important nowadays.

This thesis has five major contributions. First, we construct twoAttribute Based Encryption schemes for achieving the constant ciphertext size and hidden ciphertext policy. Second, by combining Attribute Based Encryption and Broadcast Encryption, we construct Attribute Based Broadcast Encryption schemes with short ciphertext and short decryption key. Third, We also explore the anonymity of Attribute Based Broadcast Encryption supporting multi-gate access structures. Fourth, we propose two ciphertext policy hidden vector encryption schemes with constant-size ciphertext, and attribute hiding. Both of our proposed schemes achieve the efficiency and flexibility. Finally, we construct a new type of fuzzy public key encryption, called Edit Distance-based Encryption, based on the Edit Distance which is a very useful tool to measure the similarity between two strings.

In our constructions, we define the access policy by applying the Boolean AND Gates Access Structure with positive, negative attributes including wildcard; OR-AND Gates with positive, negative attributes. We also develop techniques to bridge Attribute Based Encryption, Attribute Based Broadcast Encryption with Inner Product Encryption, and then use the latter to achieve the goal of hidden access policy. All of our proposed schemes are proven secure under standard assumptions.
Acknowledgement

I would like to express my sincere gratitude to my principal supervisor Prof. Willy Susilo for the constant support of my PhD study and research, for his patience, motivation, and immense knowledge. Willy has been an excellent supervisor who has advised me directions yet enough freedom for me to explore different areas. I would like to thank my co-advisor, Dr. Guomin Yang. He helps me a lot during my first times in Wollongong. I always appreciate about that. He always listens and gives tremendous advices in my research. He has helped me sort out the technical details of my work, and encouraged the use of correct grammar and consistent notation in every manuscript I worked on.

My sincere thanks also goes to Dr. Vincenzo Iovino, Dr. Kaitai Liang, Dr. Fuchuo Guo, Prof. Qiong Huang, for their invaluable advice on my research. I am fortunate to have nice colleagues and friends during my PhD study in Wollongong. Many thanks for their kindness. The non-exhausted list includes Jongkil Kim, Nan Li, Clementine Gritti, Yangguang Tian, Rongmao Chen. It certainly includes all the authors of many papers appeared in the reference list.

I would like to thank University of Wollongong offering me the University Postgraduate Award scholarships. I also thank all staffs from School of Computing and Information Technology for their supports during my study.

Everybody has a starting point, and I know that has changed my life. I wish to take this opportunity to give my heartfelt thanks to Prof. Nguyen Dinh Thuc, my supervisor when I was a teaching assistant in Vietnam National University - Ho Chi Minh City - University of Science, Vietnam. It is Prof. Thuc who brings me into the field of cryptography.

I also sincerely thank Ms Ngoc Tho for her constant sharing and cheering up during my research. And thanks Ngoc To, Mai Huynh, Tuyet Trinh, Van Dang for their encouragements.

Finally, my love and gratitude to Father Hung, Mother Ha and my sister Cathy Phuong for their endless patience, encouragement and love. Without them, all my achievements would never be possible.
Publications

The following papers have been published, and contain materials presented in this thesis:


Other papers not included in this thesis:

1. A DFA-Based Functional Proxy Re-Encryption Scheme for Secure Public Cloud Data Sharing, Kaitai Liang, Man Ho Au, Joseph K. Liu, Willy Susilo, Duncan S. Wong, Guomin Yang, Tran Viet Xuan Phuong and Qi Xie. IEEE Transactions


List of Tables

2.1 List of attributes and policies ........................................ 13
2.2 List of attributes and OR/AND policies ............................ 13
3.1 Comparison among CP-ABE with Hidden AND-Gate access structure 47
4.1 Performance Comparison in CP-ABBE ............................... 64
4.2 Performance Comparison in KP-ABBE ............................... 64
5.1 Comparisons in CP-ABBE ............................................. 87
6.1 Performance Comparison ............................................. 112
# List of Figures

2.1 Checking one root of $\mathcal{P}$ .............................. 14

3.1 Access Policy ................................................. 27

3.2 The vector $\vec{v}$ for access policy $W_2$ ....................... 30

3.3 The vector $\vec{z}$ for Alice and Bob .......................... 30

5.1 The vector $\vec{v}$ for access policy $W_2$ ....................... 68

5.2 The vector $\vec{z}$ for Alice and Bob .......................... 68

7.1 Edit Distance Evaluation using Bilinear Map .................... 115
# Contents

Abstract iii

Acknowledgement iv

Publications v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Background 6

2.1 Miscellaneous Notations ................................. 6
2.2 Abstract Algebra ........................................ 7
  2.2.1 Groups ............................................. 7
  2.2.2 Isomorphisms ....................................... 7
  2.2.3 Cyclic Groups and Generators ...................... 8
  2.2.4 Field ............................................. 8
  2.2.5 Bilinear Map on Prime Order Groups ................ 9
  2.2.6 Bilinear Map on Composite Order Groups .......... 9
2.3 Complexity Assumptions .................................. 10
2.4 Access Structure ...................................... 12
  2.4.1 AND Gates with positive, negative attributes and wildcard 12
  2.4.2 Arbitrary OR/AND Gates with positive and negative attributes 13
2.5 Mathematical Tools .................................... 14
  2.5.1 Polynomial and Roots ............................... 14
  2.5.2 The Viète’s formulas ............................... 15
2.6 Edit Distance Algorithm ................................. 16
  2.6.1 Edit Distance ....................................... 16
  2.6.2 Dynamic Programming for Edit Distance .......... 16
2.7 Cryptography Tools ................................... 17
3 Ciphertext Policy Attribute Based Encryption under Standard Assumptions

3.1 Definition

3.1.1 CP-ABE Definition

3.1.2 Security Definition for CP-ABE with Hidden Access Policy

3.2 Constructions

3.2.1 Efficient Ciphertext Policy Attribute Based Encryption Under Decisional Linear Assumption

3.2.2 Hidden Ciphertext Policy Attribute Based Encryption under Standard Assumptions

3.3 Security Proof

3.3.1 Security Proof-Efficient CP-ABE Under Decisional Linear Assumption

3.3.2 Security Proof-Hidden CP-ABE

3.4 Comparisons

3.5 Summary

4 Attribute Based Broadcast Encryption with Short Ciphertext and Decryption Key

4.1 Definition

4.1.1 KP-ABBE Definition

4.1.2 Security Definition for KP-ABBE

4.1.3 CP-ABBE Definition

4.1.4 Security Definition for CP-ABBE

4.2 Constructions

4.2.1 KP-ABBE Scheme

4.2.2 CP-ABBE Scheme

4.3 Security Proof

4.4 Comparisons
## Contents

4.5 Summary ................................................................. 64

5 Anonymous Attribute Based Broadcast Encryption Supporting Multi-Gate Access Structure .................................................. 65
  5.1 Constructions .......................................................... 66
    5.1.1 Anonymous AND Attribute Based Broadcast Encryption .......... 66
    5.1.2 Security Proof ..................................................... 72
    5.1.3 Anonymous OR-AND-ABBE scheme .................................. 82
    5.1.4 Security Proof ..................................................... 86
  5.2 Comparisons .......................................................... 87
  5.3 Summary ............................................................... 87

6 Hidden Vector Encryption with Constant Ciphertext Policy .......... 88
  6.1 Definition ............................................................. 90
    6.1.1 Ciphertext-Policy Hidden Vector Encryption ....................... 90
    6.1.2 Security Model .................................................... 90
  6.2 Construction .......................................................... 91
    6.2.1 CP-HVE Scheme 1 .................................................. 91
    6.2.2 Security Proof ..................................................... 94
    6.2.3 CP-HVE Scheme 2 .................................................. 107
    6.2.4 Security Proof of CCP-HVE2 ...................................... 109
  6.3 Comparison ........................................................... 111
  6.4 Summary ............................................................... 112

7 Edit Distance Based Encryption .......................................... 113
  7.1 Definition ............................................................. 116
    7.1.1 Edit Distance Based Encryption Definition ....................... 116
    7.1.2 Security Model of EDE scheme ................................... 116
  7.2 Construction .......................................................... 117
    7.2.1 Edit Distance based Encryption .................................. 117
  7.3 Security Proof ........................................................ 121
  7.4 EDE Application-Fuzzy Broadcast Encryption .......................... 124
    7.4.1 Definition ........................................................ 125
    7.4.2 Construction ....................................................... 125
  7.5 Summary ............................................................... 127

8 Concluding Remark ....................................................... 129
  8.1 Summary of our contributions ....................................... 129
    8.1.1 Ciphertext Policy Attribute Based Encryption ................... 129
    8.1.2 Attribute Based Broadcast Encryption ............................. 129
Chapter 1

Introduction

Traditional public key encryption establishes a secure public channel, in which the public key is generated by a party as an encryption key; anyone who knows that public key can use it to encrypt messages and create the corresponding ciphertexts. The private key is used by the party who knows it to recover the original message from any ciphertexts generated using the matching public key. Although the security guarantee can be achieved in rigorousness as indistinguishability under chosen plaintext and chosen ciphertext attacks, the notion of public key encryption is insufficient in many applications. For example, emerging applications such as access control services often require to specify an access policy in the ciphertext, and only individuals who satisfy the policy can decrypt. More generally, the decryptor may be authorized to access a function of the plaintext. Unfortunately, the traditional public key cryptography cannot tackle with such tasks.

As a new broad vision of public key encryption, functional encryption provides a promising solution for many challenging security problems such as expressive access control and searching on encrypted data with a strong combination of flexibility, efficiency, and security.

Functional Encryption. The first functional encryption is the Fuzzy Identity Based Encryption (IBE) from Sahai and Waters [SW05a]. In Fuzzy IBE schemes, the receivers ability to decrypt is merely contingent on his knowledge of a private key associated with an identity that matches a string chosen by the sender. In a functional encryption scheme for functionality $\mathcal{F}(.,.)$, an authority holding a master secret key can generate a key $sk_k$ that enables the computation of the function $\mathcal{F}(k,.)$ on encrypted data. Then using $sk_k$ the decryptor can compute $\mathcal{F}(k,x)$ from an encryption of $x$. Intuitively, the security should guarantees that it cannot learn anything more about $x$. One well-representative of functional encryption is
Attribute Based Encryption (ABE).

**Attribute Based Encryption.** It was introduced by Sahai and Waters [SW05a] and extensively studied in recent years [GPSW06, BSW07, Wat11a, LW12]. ABE provides a fine-grained access control of encrypted data. There are two types of ABE called Ciphertext-Policy ABE (CP-ABE)[BSW07], and Key-Policy ABE (KP-ABE)[GPSW06]. Under the construction of CP-ABE [BSW07], a secret key is associated to a user’s attributes, such as \{“Student”, “Faculty : CS”, “Major : Cryptography”\}, and a ciphertext is associated to access policies by composing multiple attributes through logical operators such as “AND”, “OR”, e.g., “Student” \( \land \) (“Birthday:1988” \( \lor \)“Faculty:CS”). If a decryptor wants to decrypt the message successfully, the attributes embedded in the secret key must satisfy the access policies embedded in the ciphertext. In contrast to CP-ABE [BSW07], in a KP-ABE scheme [GPSW06], an encryptor associates a set of attributes to the message during the encryption. Each user is assigned an access structure which is usually defined as an access tree over data attributes, i.e., interior nodes of the access tree are threshold gates and leaf nodes are associated with attributes. User secret key is defined to reflect the access structure so that the user is able to decrypt a ciphertext if and only if the data attributes satisfy his access structure.

**Predicate Encryption.** As another special sub-class of function encryption systems, predicate encryption (PE) offers a mechanism that provides more fine-grained control over access to encrypted data. In a PE scheme, secret keys \(SK_f\) correspond to predicates or boolean functions \(f\) in some class \(\mathcal{F}\), and a sender associates a ciphertext with an index \(I\) in some set \(\Sigma\). The decryption can be successful if and only if \(f(I) = 1\). There are three typical types of predicate encryption systems including Anonymous Identity Based Encryption, Hidden Vector Encryption, and Inner Product Encryption.

- **Anonymous Identity Based Encryption.** The functionality of Anonymous IBE [BCOP04a] is similar to IBE except that the string representing the ciphertext identity is hidden and one can only determine it if they have the corresponding private key.

- **Hidden Vector Encryption.** Hidden Vector Encryption (HVE) schemes [BW07, KSW08a, SLN+10, HHI+11] allow wildcards to appear in either the encryption attribute vector associated with a ciphertext or the decryption attribute vector associated with a user secret key. Similar to ABE schemes, there are two kinds of HVE: Ciphertext Policy (CP-) HVE schemes and the Key Policy (KP-) HVE schemes. The decryption will work if and only if the
two vectors match. That is, for each position, the two vectors must have the same letter (defined in an alphabet $\Sigma$) unless a wildcard symbol ‘⋆’ appears in one of these two vectors at that position.

- **Inner Product Encryption.** As mentioned in [KSW08a], inner product is considered to be the most important class $\mathcal{F}$ of predicates. The class $\mathcal{F}$ of inner-product predicates is defined as follows: a ciphertext (and a private key) is associated with a vector $\vec{x}$ (and $\vec{y}$), then the ciphertext can be decrypted by the private key $SK_{\vec{y}}$ if and only if $<\vec{x}, \vec{y}> = 0$. The inner-product predicate suffices to express a wide class of predicates including conjunctive normal form/disjoint normal form (CNF/DNF) formulas and polynomial evaluations. Furthermore, it can support the attribute-hiding property and can be also extended to construct PE schemes supporting different types of predicates.

It is worth mentioning that there is another variant of public key encryption called **Broadcast Encryption** (BE). This aims to guarantee the communication services in broadcasting and exchanging information confidentially and efficiently. Berkovits, Fiat and Naor [Ber91, FN93] firstly proposed the notion of Broadcast Encryption (BE). In this setting, a center is allowed to broadcast a secret to any subset of privileged users out of a universe of size $n$ so that conjunctions of $k$ users not in the privileged set cannot learn the secret. The functional encryption schemes reviewed above can naturally be used as broadcast encryption.

In order to be useful in real applications, the efficiency and flexibility of functional encryption is a necessary requirement. For instance, in many ABE schemes, the size of ciphertext will increase as long as the number of attributes increases. It is desirable that even though the number of attributes increases, the size of ciphertext remains constant. Another challenging problem is protect the user privacy in ABE or Predicate Encryption schemes. This means that the user can decrypt the message without knowing access policy or the predicate $\mathcal{F}$.

This thesis focuses on designing the aforementioned types of functional encryption. Our goal is to enhance the ABE, HVE, IPE schemes in terms of efficiency, security and flexibility, so that they can be applicable in the real scenarios. We also investigate the relationships among these types of functional encryption to construct new schemes with stronger security. Beyond that, we also construct new predicate encryption scheme by exploring a real measurement metric to instantiate the predicate.

**Summary of my results**

In this thesis, we present two new Ciphertext Policy Attribute Based Encryption
(CP-ABE) schemes where the access policy is defined by AND-Gate with wildcard in chapter 3. In the first scheme, we introduce a new technique to construct a new CP-ABE scheme with constant ciphertext size. We prove our first scheme is secure under the standard Decisional Linear assumptions. In the second scheme, we propose a new CP-ABE scheme with the property of hidden access policy by extending the technique we used in the construction of our first scheme. We also prove that our second scheme is secure under the standard Decisional Linear (DLIN) and Decisional Bilinear Diffie-Hellman (DBDH) assumptions.

Attribute Based Broadcast Encryption (ABBE) scheme is established by a combination of Attribute-based encryption and Broadcast encryption. We focus on the efficiency of ABBE scheme in terms of the ciphertext size and the key size. We propose two schemes: Key Policy Attribute Based Broadcast Encryption (KP-ABBE) with short key size and Ciphertexts Policy Attribute Based Broadcast Encryption (CP-ABBE) with short ciphertext size. Both of proposed schemes are proved secure under the $n$-Bilinear Diffie Hellman Exponent assumption.

We also aim to achieve the anonymity for ABBE scheme which is desirable in the real applications. Toward this goal, we propose the Anonymous Attribute Based Broadcast Encryption with Multi-Gate Access Structure. The first proposed scheme supports AND gates with positive, negative attributes and wildcard. This is an extension of the aforementioned ABBE scheme. The second proposed scheme supports OR/AND gates with positive, and negative attributes. Specifically, we show a way to bridge ABBE with Inner Product Encryption (IPE) in order to achieve the goal of anonymity. We also prove that both schemes are secure under the standard DLIN and DBDH assumptions.

Next, we introduce two constant size ciphertext policy hidden vector encryption (CP-HVE) schemes. Our first scheme is constructed on composite order bilinear groups, while the second one is built on bilinear groups with prime order. Both schemes are proven secure in a selective security model which captures plaintext (or payload) and attribute hiding. Our schemes are the first HVE constructions that can achieve constant ciphertext size among all the existing HVE schemes.

Finally, a new type of fuzzy public key encryption called Edit Distance-based Encryption (EDE) is proposed. We provide a formal definition and security model for EDE, and propose an EDE scheme that can securely evaluate the edit distance between two strings embedded in the ciphertext and the secret key. We also show an interesting application of our EDE scheme named Fuzzy Broadcast Encryption which is very useful in a broadcasting network.

**Organization of This Thesis**

In Chapter 2, we introduce our notations and review background materials nec-
nessary for understanding the remainder of this thesis. We next present two new Ciphertext Policy Attribute Based Encryption schemes in Chapter 3. In Chapter 4, we present Attribute Based Broadcast Encryption with short ciphertext and decryption key. The anonymous Attribute Based Broadcast Encryption with AND, OR gates access structure is presented in Chapter 5. We introduce two constant size ciphertext policy hidden vector encryption schemes in Chapter 6. A new type of fuzzy public key encryption called Edit Distance-based Encryption is presented in Chapter 7. In Chapter 8, we conclude the thesis and discuss some future research directions.
Chapter 2

Background

In this chapter, we cover the notations and definitions that will be used throughout this thesis. Background materials on the topic of number theory, bilinear maps, mathematical tools and other preliminaries will be presented. More details of theory and mathematical proof can be found in the following book [KL14].

2.1 Miscellaneous Notations

In this thesis, we denote the security parameter by $\lambda$. By $1^\lambda$, we denote the string of $n$ ones. By $r \leftarrow \mathbb{Z}_p$, we denote the value $r$ is randomly selected from the finite field $\mathbb{Z}_p$.

Negligible success probability. A negligible function is one that is asymptotically smaller than any inverse polynomial function.

Definition 1 A function $f$ from the natural numbers to the non-negative real numbers is negligible if for every positive polynomial $p$ there is an $N$ such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

Computational Indistinguishability. We say that two distribution families $\Omega_1(l)$ and $\Omega_2(l)$ are computationally indistinguishability if, for all Probabilistic Polynomial Time (PPT) algorithm $A$:

$$|Pr_{x \in \Omega_1(l)}[A(x) = 1] - Pr_{x \in \Omega_2(l)}[A(x) = 1]| \leq negl(l)$$

Statistical Indistinguishability. We say that two distribution families $\Omega_1(l)$ and $\Omega_2(l)$ are statistical indistinguishability if,

$$\sum_x |Pr_{x \in \Omega_1(l)}[x = z] - Pr_{x \in \Omega_2(l)}[x = z]| \leq negl(l)$$
2.2 Abstract Algebra

2.2.1 Groups

Let $G$ be a set. A binary option $\circ$ on $G$ is a function $\circ(\cdot,\cdot)$ that takes as input two elements of $G$. If $g, h \in G$, then we write $g \circ h$.

**Definition 2** A group is a set $G$ along with a binary option $\circ$ for which the following conditions hold:

- **Closure**: For all $g, h \in G$, $g \circ h \in G$.
- **Existence of an identity**: There exists an identity $e \in G$ such that for all $g \in G$, $e \circ g = g = g \circ e$.
- **Existence of inverses**: For all $g \in G$ there exists an element $h \in G$ such that $g \circ h = e = h \circ g$. Such an $h$ is called an inverse of $g$.
- **Associativity**: For all $g_1, g_2, g_3 \in G$, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$. When $G$ has a finite number of elements, we say $G$ is finite and let $|G|$ denote the order of the group. A group $G$ with operation $\circ$ is abelian if the following holds:
- **Commutativity**: For all $g, h \in G$, $g \circ h = h \circ g$.

**Proposition 1** Let $b, N$ be integers, with $b \leq 1$ and $N \leq 1$. Then $b$ is invertible modulo $N$ if and only if $\gcd(b, N) = 1$.

**Lemma 1** Let $G$ be a group and $a, b, c \in G$. If $ac = bc$, then $a = b$. In particular, if $ac = c$ then $a$ is the identity of $G$.

**Theorem 1** Let $G$ be a finite group with $m = |G|$, the order of the group. Then for any element $g \in G$, $g^m = 1$. ($1$ denote to the identity of group $G$).

2.2.2 Isomorphisms

Two groups are isomorphic if they have the same underlying structure. From a mathematical point of view, an isomorphism of a group $G$ provides an alternate, but equivalent, way of thinking about $G$. From a computational perspective, an isomorphism provides a different way to represent elements in $G$.

**Definition 3** Let $G, H$ be groups with respect to the operations $\circ_G, \circ_H$, respectively. A function $f : G \to H$ is an isomorphism from $G$ to $H$ if:

1. $f$ is a bijection, and

2. all $g_1, g_2 \in G$, we have $f(g_1 \circ_G g_2) = f(g_1) \circ_H f(g_2)$.
2.2.3 Cyclic Groups and Generators

Let $G$ be a finite group of order $m$. For arbitrary $g \in G$, consider the set

$$< g > \overset{\text{def}}{=} \{ g^0, g^1, \ldots \}.$$ 

By Theorem 1, we have $g^m = 1$. Let $i \leq m$ be the smallest positive integer for which $g^i = 1$. Then the above sequence repeats after $i$ terms, and so

$$< g > = \{ g^0, \ldots, g^{i-1} \}.$$ 

We see that $< g >$ contains at most $i$ elements. In fact, it contains exactly $i$ elements since if $g^j = g^k$ with $0 \leq j < k < i$ then $g^{k-j} = 1$ and $0 < k - j < i$, contradicting our choice of $i$ as the smallest positive integer for which $g^i = 1$.

**Theorem 2** Let $G$ be a finite group and $g \in G$. The order of $g$ is the smallest positive integer $i$ with $g^i = 1$.

**Proposition 2** Let $G$ be a finite group and $g \in G$ an element of order $i$. Then for any integer $x$, we have $g^x = g^{[x \mod i]}$.

**Proposition 3** Let $G$ be a finite group, and $g \in G$ an element of order $i$. Then $g^x = g^y$ if and only if $x = y \mod i$.

**Corollary 1** If $G$ is a group of prime order $p$, then $G$ is cyclic. Furthermore, all elements of $G$ except the identity are generators of $G$.

**Theorem 3** If $p$ is prime then $\mathbb{Z}_p^*$ is a cyclic group of order $p - 1$.

2.2.4 Field

Field consists of a set of elements and two operations defined between any two elements in the set.

**Definition 4** A field $\mathbb{F}$ consists of a set $\mathbb{F}$ and two operations: addition $\oplus$, and multiplication $\otimes$, and satisfies the following properties.

- **Addition Group.** $(\mathbb{F}, \oplus)$ is an Abelian group. The identity of group $(\mathbb{F}, \oplus)$ is $0_\mathbb{F}$.

- **Multiplication Group.** Let $\mathbb{F}^* = \mathbb{F} - \{0_\mathbb{F}\}$ $(\mathbb{F}^*, \otimes)$ is an Abelian group. The identity of group $(\mathbb{F}^*, \otimes)$ is $1_\mathbb{F}$.

- **Distributivity.** For all $g_1, g_2, g_3 \in \mathbb{F}$, $(g_1 \oplus g_2) \otimes g_3 = (g_1 \otimes g_3) \oplus (g_2 \otimes g_3)$. 

2.2.5 Bilinear Map on Prime Order Groups

A pairing is a bilinear map from a pair of group elements to an element of a different group. Specifically, let $G_1, G_2, G_T$ be cyclic groups of order $p$. Let $g_1$ and $g_2$ be generators of $G_1$ and $G_2$ respectively. A function $e : G_1 \times G_2 \rightarrow G_T$ is said to be a pairing if it satisfies the following properties:

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

2. Non-degeneracy : $e(g_1, g_2) \neq 1_{G_T}$, where $1_{G_T}$ is the identity element in $G_T$.

3. Efficient Computability: $e(u, v)$ can be computed efficiently (that is, in polynomial time) for all $u \in G_1$ and $v \in G_2$.

4. Unique Representation: All elements in $G_1, G_2$ and $G_T$ have unique binary representation.

Galbraith et al. [GPS08] classify pairing into three types. Specifically, if $G_1 = G_2$ (or there exists efficiently computable isomorphism between the two groups), the pairing $e$ is classified as type 1. If $G_1 \neq G_2$ and there is no efficiently computable homomorphism from $G_1$ to $G_2$ but there exists an efficiently computable homomorphism $\psi : G_2 \rightarrow G_1$, the pairing $e$ is classified as type 2. Finally, if $G_1 \neq G_2$ and there are no efficiently computable homomorphisms between $G_1$ and $G_2$, the pairing $e$ belongs to type 3. Broadly speaking, type 1 is sometimes known as symmetric pairing while type 2 and type 3 are known as asymmetric pairing. Typically, $G_1$ and $G_2$ are subgroups of the group of points on an elliptic curve over a finite field, $G_3$ will be a subgroup of the multiplicative group of a related finite field and the map $e$ will be derived from either the Weil [BF01a] or Tate pairing [BKMX06] on the elliptic curve.

2.2.6 Bilinear Map on Composite Order Groups

From [BGN05, Fre10], let $p, q$ be two large prime numbers and $n = pq$. Let $G, G_T$ be cyclic groups of order $n$, We say $e : G \times G \rightarrow G_T$ is bilinear map on composite order groups if $e$ satisfies the following properties:

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

2. Non-degeneracy : $e(g, g) \neq 1_{G_T}$

Let $G_p$ and $G_q$ be two subgroups of $G$ of order $p$ and $q$, respectively. Then $G = G_p \times G_q$, $G_T = G_{T,p} \times G_{T,q}$. We use $g_p$ and $g_q$ to denote generators of $G_p$ and $G_q$, respectively. $e(h_p, h_q) = 1$ for all elements $h_p \in G_p$ and $h_q \in G_q$ since $e(h_p, h_q) = e(g_p^a, g_q^b) = e(g_q^a, g_p^b) = e(g, g)^{pqab} = 1$ for a generator $g$ of $G$. 
2.3 Complexity Assumptions

Discrete Logarithm Problem. The discrete logarithm problem (DLP) [BL96] forms the basis in the security of many cryptosystems. We restrict ourselves to DLP in cyclic group in this thesis.

The Discrete Logarithm Problem in $\mathbb{G} = \langle g \rangle$ is defined as follows: On input a tuple $(g, Y) \in \mathbb{G}_2$, output $x$ such that $Y = g^x$.

Computational Diffie-Hellman Problem. If we can solve DLP in $\mathbb{G}$, we can also solve the computational Diffie-Hellman (CDH) [DH76] problem although whether the converse is true or not is still an open problem.

The CDH problem in $\mathbb{G} = \langle g \rangle$ is defined as follows: On input a tuple $(g, g^x, g^y) \in \mathbb{G}^3$, output $g^{xy}$.

Decisional Diffie-Hellman Assumption. It is the decisional version of the CDH problem. It was first formally introduced in [Bra93].

The DDH problem in $\mathbb{G} = \langle g \rangle$ is defined as follows: On input a tuple $(g, g^x, g^y, g^z) \in \mathbb{G}^4$, decide if $g^z = g^{xy}$.

Decisional Linear (DLIN) Assumption. It was first introduced in [BBS04].

The Decisional Linear (DLIN) problem in $\mathbb{G}$ defined as follows: given a tuple $(g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}^6 \times \mathbb{G}_T$, decide whether $T = g^{b(c+d)}$ or $Z$ in random in $\mathbb{G}$. An algorithm $A$ has advantage $\epsilon$ in solving the DLIN problem in $\mathbb{G}$ if
\[
\text{Adv}^{\text{DLIN}}_A(k) = \Pr[A(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^{b(c+d)}] - \Pr[A(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^r] \leq \epsilon,
\]
where $a, b, c, d, r \in \mathbb{Z}_p$. We say that the DLIN assumptions holds in $\mathbb{G}$ if $\epsilon$ is negligible for any PPT algorithm $A$.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption. It was introduced by Boneh and Franklin [BF01a] in their Identity Based Encryption (IBE) scheme.

The Decisional Bilinear Diffie-Hellman (DBDH) problem in $\mathbb{G}$ is defined as follows: given a tuple $(g, g^a, g^b, g^c, T) \in \mathbb{G}^4 \times \mathbb{G}_T$, decide whether $T = e(g, g)^{ab}c$ or $T = e(g, g)^r$ where $a, b, c, r$ are randomly selected from $\mathbb{Z}_p$. An algorithm $A$ has advantage $\epsilon$ in solving the DBDH problem in $\mathbb{G}$ if
\[
\text{Adv}^{\text{DBDH}}_A(k) = \Pr[A(1^k, g, g^a, g^b, g^c, T) = 1 | T = e(g, g)^{ab}c] - \Pr[A(1^k, g, g^a, g^b, g^c, T) = 1 | T = g^r] \leq \epsilon.
\]
We say that the DBDH assumptions holds in \( G \) if \( \epsilon \) is negligible for any PPT algorithm \( A \).

**Decision \( L \)-Bilinear Diffie-Hellman Exponent (BDHE) Assumption**

The Decision \( L \)-BDHE problem was introduced by Boneh, Boyen, and Goh [BBG05] to construct a hierarchical identity-based encryption scheme with constant size ciphertext, and later used for a public key broadcast encryption scheme with constant size transmission overhead [BGW05].

The problem is defined as follows: Let \( G \) be a bilinear group of prime order \( p \), and \( g, h \) two independent generators of \( G \). Denote \( \overrightarrow{y}_{g,a,L} = (g_1, g_2, \ldots, g_L, g_{L+2}, \ldots, g_{2L}) \in G^{2L-1} \) where \( g_i = g^{\alpha_i} \) for some unknown \( \alpha \in \mathbb{Z}_p^* \). We say that the \( L \)-BDHE assumption holds in \( G \) if for any probabilistic polynomial-time algorithm \( A \)

\[
|\Pr[A(g, h, \overrightarrow{y}_{g,a,L}, e(g_{L+1}, h)) = 1] - \Pr[A(g, h, \overrightarrow{y}_{g,a,L}, T) = 1]| \leq \epsilon(k),
\]

where the probability is over the random choice of \( g, h \) in \( G \), the random choice \( \alpha \in \mathbb{Z}_p^* \), the random choice \( T \in G_T \), and \( \epsilon(k) \) is negligible in the security parameter \( k \).

**Decisional \( L \)- composite Bilinear Diffie-Hellman Exponent (cBDHE) assumption.** The \( L \)- composite Bilinear Diffie-Hellman Exponent has been used for constructing cryptographic schemes [BBG05] to design HIBE with constant size ciphertexts in [4]

Let \( g_p, h \xleftarrow{\$} G_p, g_q, \xleftarrow{\$} G_q, \alpha \xleftarrow{\$} \mathbb{Z}_n \),

\[
Z = (g_p, g_q^\alpha, g_{pQ}^{\alpha_p}, g_{qL+2}^{\alpha_q}, \ldots, g_{p2L}^{\alpha_p}, g_{q2L}^{\alpha_q}),
\]

\( T = e(g_p, h)^{\alpha_{L+1}} \), and \( R \xleftarrow{\$} G_T \).

We say that the decisional \( L - cBDHE \) assumption holds if for any probabilistic polynomial-time algorithm \( A \)

\[
|\Pr[A(Z, T) = 1] - \Pr[A(Z, R) = 1]| \leq \epsilon(k),
\]

where \( \epsilon(k) \) denotes an negligible function of \( k \).

**\( L \)-composite Decisional Diffie-Hellman (cDDH) assumption [BBG05]**

Let \( g_p, g_q, R_1, R_2, R_3 \xleftarrow{\$} G_q, \alpha, \beta \xleftarrow{\$} \mathbb{Z}_n \),

\[
Z = (g_p, g_q^\alpha, g_p^{\alpha_p}, g_q^{\alpha_q}, g_{pL+1}^{\alpha_p} R_1, g_{qL+1}^{\alpha_q} R_2),
\]

\( T = g_p^\beta R_3 \), and \( R \xleftarrow{\$} G \).
We say that the $L - cDDH$ assumption holds if for any probabilistic polynomial-time algorithm $A$
\[
|\Pr[A(Z, T) = 1] - \Pr[A(Z, R) = 1]| \leq \epsilon(k),
\]
where $\epsilon(k)$ denotes an negligible function of $k$.

**Bilinear Subset Decision assumption (BSD) assumption.** This assumption was introduced by Boneh, Sahai, and Waters [BSW06].

Let $g_p \leftarrow G_p, g_q \leftarrow G_q,$
\[Z = (g_p, g_q),\]
\[T \leftarrow G_{T,p}, \text{ and } R \leftarrow G_{T,p}.\]

We say that the BSD assumption holds if for any probabilistic polynomial-time algorithm $A$
\[
|\Pr[A(Z, T) = 1] - \Pr[A(Z, R) = 1]| \leq \epsilon(k),
\]
where $\epsilon(k)$ denotes an negligible function of $k$.

## 2.4 Access Structure

An access structure is used to define an access policy. We focus on two Boolean gates access structures: the AND Gates Structure, and the OR/AND Gates Structure.

**2.4.1 AND Gates with positive, negative attributes and wildcard**

Let $U = \{Att_1, Att_2, ..., Att_L\}$ be the universe of attributes in the system. Each attribute $Att_i$ has two possible values: positive and negative. Let $W = \{Att_1, Att_2, ..., Att_L\}$ be an AND-gates access policy with wildcards. A wildcard ‘*’ means “don’t care” (i.e., both positive and negative attributes are accepted). We use the notation $S \models W$ to denote that the attribute list $S$ of a user satisfies $W$.

For example, suppose $U = \{Att_1 = CS, Att_2 = EE, Att_3 = Faculty, Att_4 = Student\}$. Alice is a student in the CS department; Bob is a faculty in the EE department; Carol is a faculty holding a joint position in the EE and CS department. Their attribute lists are illustrated in Table 2.2. The access structure $W_1$ can be satisfied by all the CS students, while $W_2$ can be satisfied by all CS people.
Table 2.1: List of attributes and policies

<table>
<thead>
<tr>
<th>Attributes</th>
<th>$Att_1$</th>
<th>$Att_2$</th>
<th>$Att_3$</th>
<th>$Att_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>CS</td>
<td>EE</td>
<td>Faculty</td>
<td>Student</td>
</tr>
<tr>
<td>Alice</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Bob</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Carol</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>$W_1$</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>$W_2$</td>
<td>+</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

2.4.2 Arbitrary OR/AND Gates with positive and negative attributes

Table 2.2: List of attributes and OR/AND policies

<table>
<thead>
<tr>
<th>Attributes</th>
<th>$Att_1$</th>
<th>$Att_2$</th>
<th>$Att_3$</th>
<th>$Att_4$</th>
<th>$Att_5$</th>
<th>$Att_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>CS</td>
<td>EE</td>
<td>Professor</td>
<td>F.Officer</td>
<td>Student</td>
<td>Tutor</td>
</tr>
<tr>
<td>$W_1 - W_{11}$</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$W_1 - W_{12}$</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$W_2 - W_{21}$</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$W_2 - W_{22}$</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$W_2 - W_{23}$</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$W_2 - W_{24}$</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Let $W_1 = ((\text{AND}_{i \in \{1,\ldots,m\}} A_i)_1 \text{OR}_1 (\text{AND}_{i \in \{1,\ldots,m\}} A_i)_2 \text{OR}_2 \ldots \text{OR}_{m-1} (\text{AND}_{i \in \{1,\ldots,m\}} A_i))$ be the disjunctive normal form (DNF) and,

$W_2 = ((\text{OR}_{i \in \{1,\ldots,m\}} A_i)_1 \text{AND}_1 (\text{OR}_{i \in \{1,\ldots,m\}} A_i)_2 \text{AND}_2 \ldots \text{AND}_{m-1} (\text{OR}_{i \in \{1,\ldots,m\}} A_i))$

the conjunctive normal form (CNF). Then $W_1, W_2$ in Table 2.2 can be expressed as:

$W_1 = \underbrace{(Att_1 \text{ AND } Att_3)}_{W_{11}} \text{ OR } \underbrace{(Att_1 \text{ AND } Att_6 \text{ AND } Att_5)}_{W_{12}}$

which can be satisfied by CS Professor or CS Student and Tutor; and.

$W_2 = ((Att_1 \text{ OR } Att_2) \text{ AND } (Att_3 \text{ OR } Att_4))$
which can be satisfied by CS Prof or CS Faculty Officer or EE Prof or EE Faculty Officer. Hence $W_2$ can also be expressed as:

$$W_2 = (\overbrace{\text{Att}_1 \text{ AND } \text{Att}_3}^{W_{21}}) \text{ OR } (\overbrace{\text{Att}_1 \text{ AND } \text{Att}_4}^{W_{22}}) \text{ OR } (\overbrace{\text{Att}_2 \text{ AND } \text{Att}_3}^{W_{23}}) \text{ OR } (\overbrace{\text{Att}_2 \text{ AND } \text{Att}_4}^{W_{24}}).$$

When a user joins the system, the user is tagged with an attribute list defined as $S = \{A_i\}_{i \in \{1, m\}}$. Then $S \models W_1$, if the set of attributes in $S$ satisfy one of AND literals in $W_1$; or $S \models W_2$, if the set of attributes in $S$ satisfy all of OR literals in $W_2$.

### 2.5 Mathematical Tools

#### 2.5.1 Polynomial and Roots

We consider a polynomial $P$ has degree $n$:

$$P = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \quad (2.1)$$

Suppose that $P$ is the set of polynomials $n$-degree. We create a vector $\vec{v}$ from the coefficients of polynomial $P$:

$$\vec{v} = (a_n, a_{n-1}, \ldots, a_1, a_0)$$

With the arbitrary value $x$, we create a vector as:

$$\vec{x} = (x \cdot x \cdots x, \underbrace{x \cdots x}_{n-1}, \ldots, x, 1)$$

If $(\vec{v} \cdot \vec{x}) = 0$, then $x$ is a root of $P$. 

![Figure 2.1: Checking one root of $P$](image)
2.5.2 The Viète’s formulas

Consider two vectors \( \vec{v} = (v_1, v_2, \ldots, v_L) \) and \( \vec{z} = (z_1, z_2, \ldots, z_L) \). Vector \( v \) contains both alphabets and wildcards, and vector \( z \) only contains alphabets. Let \( J = \{j_1, \ldots, j_n\} \subset \{1, \ldots, L\} \) denote the positions of the wildcards in vector \( \vec{v} \).

Then the following two statements are equal:

\[
v_i = z_i \lor v_i = * \quad \text{for} \quad i = 1 \ldots L
\]

\[
\sum_{i=1, i \notin J}^L v_i \prod_{j \in J} (i - j) = \sum_{i=1}^L z_i \prod_{j \in J} (i - j).
\] (2.2)

Expand \( \prod_{j \in J} (i - j) = \sum_{k=0}^n a_k i^k \), where \( a_k \) are the coefficients dependent on \( J \), then (2.2) becomes:

\[
\sum_{i=1, i \notin J}^L v_i \prod_{j \in J} (i - j) = \sum_{k=0}^n a_k \sum_{i=1}^L z_i i^k
\] (2.3)

To hide the computations, we choose random group elements \( H_i \) and put \( v_i, z_i \) as the exponents of group elements: \( H_i^{v_i}, H_i^{z_i} \). Then (2.4) becomes:

\[
\prod_{i=1, i \notin J}^L H_i^{v_i} \prod_{j \in J} (i - j) = \prod_{k=0}^n (\prod_{i=1}^L H_i^{z_i i^k}) a_k
\] (2.4)

Using the consequence of Viète’s formula we can construct the coefficient \( a_k \) in (2.3) by:

\[
\begin{cases}
  x_1 + x_2 + \ldots + x_n &= \left(-\frac{a_{n-1}}{a_n}\right) \\
  (x_1x_2 + x_1x_3 + \ldots + x_1x_n) &= \left(\frac{a_{n-2}}{a_n}\right) \\
  (x_2x_3 + x_3x_4 + \ldots + x_2x_n) + \ldots + x_{n-1}x_n &= \left(-\frac{a_{n-2}}{a_n}\right) \\
  \ldots
\end{cases}
\]

\[
 x_1x_2 \ldots x_n = (-1)^n \frac{a_0}{a_n}
\]

Equivalently, we can write:

\[
\sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq n} x_{i_1}x_{i_2} \ldots x_{i_k} = (-1)^k \frac{a_{n-k}}{a_n}
\] (2.5)

for \( k = 1, 2, \ldots, n \).

As example, Let \( J = \{j_1, j_2, j_3\} \), the polynomial is \( (x - j_1)(x - j_2)(x - j_3) \), then:

\[
\begin{align*}
a_3 &= 1 \\
a_2 &= -(j_1 + j_2 + j_3) \\
a_1 &= (j_1j_2 + j_1j_3 + j_2j_3) \\
a_0 &= -j_1j_2j_3.
\end{align*}
\]
2.6 Edit Distance Algorithm

2.6.1 Edit Distance

Consider a finite alphabet set $\mathcal{A}$ whose elements are used to construct strings. Let $Z_I$, $Z_D$ and $Z_S$ be finite sets of integers. Let the function $I : \mathcal{A} \rightarrow Z_I$ be the insertion cost function, i.e., $I(a)$ is the cost of inserting the element $a \in \mathcal{A}$ into a given string. Similarly, define the deletion cost function as $D : \mathcal{A} \rightarrow Z_D$ so that $D(a)$ is the cost of deleting the element $a \in \mathcal{A}$ from a given string. Finally, define the substitution cost function $S : \mathcal{A} \times \mathcal{A} \rightarrow Z_S$ so that for $a,b \in \mathcal{A}$, $S(a,b)$ is the cost of replacing the element $a$ by the element $b$ in a given string.

Given two strings of length $m$ and $n$, denoted by $X \in \mathcal{A}^m$ and $Y \in \mathcal{A}^n$ respectively, consider the sequence of insertion, deletion and substitution operations needed to transform $X$ into $Y$ and the corresponding aggregate cost of the transformation.

**Definition 5** The edit distance between $X$ and $Y$ is defined as the minimum aggregate cost of transforming $X$ into $Y$.

The general definition of edit distance given above considers different weights for different operations. In this chapter we will consider a simpler definition which is given below.

**Definition 6** For all $a,b \in \mathcal{A}$, let $I(a) = D(a) = 1$, $S(a,b) = 1$ when $a \neq b$, and $S(a,a) = 0$. Then, the edit distance is defined as the minimum number of insertion, deletion and substitution operations required to convert $X$ into $Y$.

2.6.2 Dynamic Programming for Edit Distance

Let $X = X_1X_2...X_m \in \mathcal{A}^m$ and $Y = Y_1Y_2...Y_n \in \mathcal{A}^n$ be two strings. We use $M(i,j)$ to denote the edit distance between the two sub strings $X_1X_2...X_i$ and $Y_1Y_2...Y_j$. The problem of finding the edit distance between $X$ and $Y$ can be solved in $O(mn)$ time via dynamic programming [Gus97], which will be used in our scheme.

Let $M(0,0) = 0$. For $1 \leq i \leq m, 1 \leq j \leq n$, define $M(i,0) = \sum_{k=1}^{i} I(x_k)$, and $M(0,j) = \sum_{k=1}^{j} D(y_k)$.

Then, the edit distance $M(m,n)$ is defined by the following recurrence relation for $1 \leq i \leq m, 1 \leq j \leq n$: $M(i,j) = \min \{M(i-1,j) + D(Y_j), M(i,j-1) + I(X_i), M(i-1,j-1) + S(X_i,Y_j)\}$.
2.7 Cryptography Tools

2.7.1 Public Key Encryption

We present the formal definition of public key encryption

Definition 7 A public-key encryption scheme is a triple of probabilistic polynomial time algorithms \((\text{Gen}, \text{Enc}, \text{Dec})\) such that:

- The key generation algorithm \(\text{Gen}\) takes as input the security parameter \(1^\lambda\) and outputs a pair of keys (public key, private key) \((pk, sk)\). We assume for that \(pk, sk\) each has length at least \(n\), and that \(n\) can be determined from \(pk, sk\).

- The encryption algorithm \(\text{Enc}\) takes input a public key \(pk\) and a message \(m\) from some message space. It outputs a ciphertext \(c\), we write it as \(c := \text{Enc}_{pk}(m)\).

- The deterministic decryption algorithm \(\text{Dec}\) takes as input a private key \(sk\) and a ciphertext \(c\), and outputs a message \(m\) or a special symbol \(\bot\) denoting failure. We write this as \(m := \text{Dec}_{sk}(c)\).

2.7.2 Security against Chosen Plaintext Attacks

Given a public-key encryption scheme \(\Pi = (\text{Gen}, \text{Enc}, \text{Dec})\) and an adversary \(A\), consider the following experiment:

- **Init**: The challenger runs \(\text{Gen}(1^\lambda)\) to obtain keys \(pk, sk\). Then the challenger sends to the adversary \(A\) the public key \(pk\).

- **Setup**: Adversary \(A\) is given \(pk\), and outputs a pair of equal-length messages \(m_0, m_1\) in the message space.

- **Challenge**: A uniform bit \(b \in \{0, 1\}\) is chosen, and then a ciphertext \(c \leftarrow \text{Enc}_{pk}(m_b)\) is computed and given to \(A\). We call \(c\) the challenge ciphertext.

- **Guess**: \(A\) outputs a bit \(b'\). The output of the experiment 1 if \(b' = b\), and 0 otherwise. If \(b' = b\) we say that \(A\) succeeds.

The advantage of \(A\) in this game is defined as \(\text{Adv}_A(\lambda) = |\Pr[b' = b] - \frac{1}{2}|\).

2.7.3 Inner Product Encryption

Let \(\Sigma \in \mathbb{Z}^n\) be the set of vectors \(\vec{v}\) of dimension \(n\), and \(\mathcal{F}\) be the class of predicates involving inner-products over vectors \((\mathcal{F} = f_\vec{v}|\vec{x} \in \Sigma\text{ such that } f_\vec{v}(\vec{y}) = 1 \text{ iff } \langle \vec{x}, \vec{y} \rangle >= 0)\). An inner-product encryption (IPE) scheme for the class of predicates \(\mathcal{F}\) consists of four algorithms as follows:
Chapter 2. Background

2.7.4 Security Model for IPE scheme

Following [KSW08a], we define the security, i.e., attribute-hiding property, of the IPE scheme. The security is defined by the following game interacted between an attacker \( A \) and a challenger \( C \). We assume that \((\Sigma, F)\) are given to both \( A \) and \( C \) in advance.

- **Init**: \( A \) outputs two vectors \( \vec{x}, \vec{y} \in \Sigma \).

- **Setup**: \( C \) runs Setup to obtain the public key \( PK \) and master secret key \( MSK \). \( A \) is given \( PK \).

- **Query Phase 1**: \( A \) adaptively issues private key queries for any vectors \( \vec{v}_1, \ldots, \vec{v}_n \in \Sigma \), subject to the restriction that, \( \forall i, \langle \vec{v}_i, \vec{x} \rangle = 0 \) if and only if \( \langle \vec{v}_i, \vec{y} \rangle = 0 \). \( C \) responds with \( SK_{\vec{v}_i} \leftarrow \text{KeyGen}(SK, \vec{v}_i) \).

- **Challenge**: \( A \) outputs two messages \( M_0, M_1 \) with equal length. If \( M_0 \neq M_1 \), then it is required that \( \langle \vec{v}, \vec{y} \rangle \neq 0 \neq \langle \vec{v}, \vec{x} \rangle \) for any \( \vec{v} \) appeared in Query Phase 1. \( C \) flips a random coin \( b \in \{0,1\} \). If \( b = 0 \), \( C \) returns \( CT \leftarrow \text{Encrypt}(PK, \vec{x}, M_0) \) to \( A \); otherwise, if \( b = 1 \), \( C \) returns \( CT \leftarrow \text{Encrypt}(PK, \vec{y}, M_1) \) to \( A \).

- **Query Phase 2**: Phase 1 is repeatedly, which is required that \( \langle \vec{v}, \vec{y} \rangle \neq 0 \neq \langle \vec{v}, \vec{x} \rangle \) for any \( \vec{v} \) if \( M_0 \neq M_1 \).

- **Guess**: \( A \) outputs a guess bit \( b' \) and succeeds if \( b' = b \).

The advantage of \( A \) in this game is defined as \( \text{Adv}_A(\lambda) = |\Pr[b' = b] - \frac{1}{2}| \).

**Definition 8** We say that an IPE scheme is attribute-hiding if for all polynomial time adversaries \( A \), we have that \( \text{Adv}(A) \) is negligible.
2.7.5 Broadcast Encryption

Let $U$ denote the set of all user indices. A broadcast encryption scheme consists of four algorithms:

- **Setup$(1^\lambda)$**: The setup algorithm takes the security parameter $1^\lambda$ as input and outputs the public parameters $PK$ and a master key $MSK$.

- **Encrypt$(S, M, PK)$**: The encryption algorithm takes as input the public parameters $PK$, a message $M$, a set of user index $S \subseteq U$, and outputs a ciphertext $CT$.

- **Key Generation$(ID, MSK, PK)$**: The key generation algorithm takes as input the master key $MSK$, public parameters $PK$, a user index $ID \in U$, and outputs a private key $SK$.

- **Decrypt$(PK, CT, SK)$**: The decryption algorithm takes as input the public parameters $PK$, a ciphertext $CT$, and a private key $SK$, and outputs a message $M$ if $ID \in S$ or a special symbol ‘⊥’.

2.7.6 Security Definition for Broadcast Encryption

We define the Selective IND-CPA security for Broadcast Encryption via the following game.

- **Init**: The adversary commits to the challenge user indexes $S^*$.

- **Setup**: The challenger runs the Setup algorithm and gives $PK$ to the adversary.

- **Phase 1**: The adversary queries for private keys with the user index $ID$ such that $ID \notin S^*$.

- **Challenge**: The adversary submits messages $M_0, M_1$ to the challenger. The challenger flips a random coin $\beta$ and passes the ciphertext $ct^* = Encrypt(PK, M_\beta, S^*)$ to the adversary.

- **Phase 2**: Phase 1 is repeated.

- **Guess**: The adversary outputs a guess $\beta'$ of $\beta$.

**Definition 9** We say that a Broadcast Encryption scheme is selective IND-CPA secure if for any probabilistic polynomial time adversary

$$Adv_{be}^{s-\text{ind-cpa}}(\lambda) = |\Pr[\beta' = \beta] - 1/2|$$

is negligible.
2.7.7 Key Policy-Attribute based Encryption

Let $N$ the set of all user attributes. A key-policy attribute based encryption scheme consists of four algorithms:

- **Setup($1^\lambda$)**: The setup algorithm takes the security parameter $1^\lambda$ as input and outputs the public parameters $PK$ and a master key $MSK$.

- **Encrypt($M, L, PK$)**: The encryption algorithm takes as input the public parameters $PK$, a message $M$, a set of attributes $L \subseteq N$, and outputs a ciphertext $CT$.

- **Key Generation($W, MSK, PK$)**: The key generation algorithm takes as input the master key $MSK$, public parameters $PK$, and an access structure $W$, then outputs a private key $SK$.

- **Decrypt($PK, CT, SK$)**: The decryption algorithm takes as input the public parameters $PK$, a ciphertext $CT$, and a private key $SK$, and outputs a message $M$ if $L \models W$ or a special symbol $\perp$.

2.7.8 Security Definition for KP-ABE

We define the Selective IND-CPA security for KP-ABE via the following game.

- **Init**: The adversary commits to the challenge set of attributes $L^*$.

- **Setup**: The challenger runs the Setup algorithm and gives $PK$ to the adversary.

- **Phase 1**: The adversary submits the access policy $W$ for a KeyGen query. If $L^* \not\models W$, the challenger gives the adversary the secret key $SK_W$. The adversary can repeat this polynomially many times.

- **Challenge**: The adversary submits messages $M_0, M_1$ to the challenger. The challenger flips a random coin $\beta$ and passes the ciphertext $Encrypt(PK, M_\beta, L^*)$ to the adversary.

- **Phase 2**: Phase 1 is repeated.

- **Guess**: The adversary outputs a guess $\beta'$ of $\beta$.

**Definition 10** We say a KP-ABE scheme is selective IND-CPA secure if for any probabilistic polynomial time adversary

$$Adv_{s-ind-crea}^{kp-abe}(\lambda) = |Pr[\beta' = \beta] - 1/2|$$

is negligible.
2.7.9 Ciphertext Policy-Attribute based Encryption

Let $N$ the set of all user attributes. A ciphertext-policy attribute based encryption (CP-ABE) scheme consists of four algorithms:

- **Setup($\lambda, U$)**: The setup algorithm takes security parameters and attribute universe description as input. It outputs the public parameters $PK$ and a master key $MSK$.

- **Encrypt($PK, M, W$)**: The encryption algorithm takes as input the public parameters $PK$, a message $M$, and access structure $W$ over the universe of attributes, and outputs a ciphertext $CT$.

- **Key Generation($MSK, L$)**: The key generation algorithm takes as input the master key $MSK$ and a set of attributes $L \subset U$, and outputs a private key $SK$.

- **Decrypt($PK, CT, SK$)**: The decryption algorithm takes as input the public parameters $PK$, a ciphertext $CT$, and a private key $SK$, and outputs a message $M$ or a special symbol ‘⊥’.

2.7.10 Security Definition for CP-ABE

We define the Selective IND-CPA security for CP-ABE via the following game.

- **Init**: The adversary commits to the challenge access structure $W^\ast$.

- **Setup**: The challenger runs the Setup algorithm and gives $PK$ to the adversary.

- **Phase 1**: The adversary submits the attribute list $L$ for a KeyGen query. If $L \not\supseteq W^\ast$, the challenger gives the adversary the secret key $SK_L$. The adversary can repeat this polynomially many times.

- **Challenge**: The adversary submits messages $M_0, M_1$ to the challenger. The challenger flips a random coin $\beta$ and passes the ciphertext $Encrypt(PK, M_\beta, W^\ast)$ to the adversary.

- **Phase 2**: Phase 1 is repeated.

- **Guess**: The adversary outputs a guess $\beta'$ of $\beta$.

**Definition 11** We say a CP-ABE scheme is selective IND-CPA secure if for any probabilistic polynomial time adversary $\text{Adv}_{\text{cp-abe}}^{s-\text{ind-cpa}}(\lambda) = |\Pr[\beta' = \beta] - 1/2|$
is negligible.
Chapter 3

Ciphertext Policy Attribute Based Encryption
under Standard Assumptions

Access control (i.e., authentication and authorisation) plays an important role in many information systems. Among all the existing cryptographic tools, Attribute Based Encryption (ABE) has provided an effective way for fine-grained access control. ABE, which is an extension of identity-based encryption (IBE) [Sha84, BF01b], allows an access structure/policy to be embedded into the ciphertext (this is referred to as ciphertext-policy ABE, or CP-ABE) or user secret key (this is referred to as key-policy ABE, or KP-ABE). In a CP-ABE, the user’s attributes used for key generation must satisfy the access policy used for encryption in order to decrypt the ciphertext, while in a KP-ABE, the user can only decrypt ciphertexts whose attributes satisfy the policy embedded in the key. We can see that access control is an inherent feature of ABE, and by using some expressive access structures, we can effectively achieve fine-grained access control.

Related Work

Since its introduction in the seminal work of Sahai and Waters [SW05a], ABE has been extensively studied in recent years (e.g., [GPSW06, BSW07, CN07, GJPS08, Wat11b, ALP11, LW12, DJ12]). There are different ways to define an access structure/policy for ABE. The fuzzy IBE given by Sahai and Waters [SW05a], which can be treated as the first KP-ABE, used a specific threshold access policy. Later, the Linear Secret Sharing Scheme (LSSS) realizable (or monotone) access structure has been adopted by many subsequent ABE schemes [GPSW06, BSW07, GJPS08, Wat11b]. In [CN07], Cheung and Newport proposed another way to define access structure using AND-Gate with wildcard. To be more precise, for each attribute in the universe, there are two possible values: positive and negative. A user’s attributes are then defined by a sequence of positive and negative symbols w.r.t. each attribute.
in the universe (assuming that the attributes are placed in order in the universe). An access structure is also defined by a sequence of positive and negative symbols, plus a special wildcard (i.e., “don’t care”) symbol. Cheung and Newport showed that by using this simple access structure, which is sufficient for many applications, CP-ABE schemes can be constructed based on standard complexity assumptions. Subsequently, several ABE schemes [NYO08, EMN⁺09, ZH10, CZF11] were proposed following this specific access structure.

**Our Contributions**

In this chapter, we explore new techniques for the construction of CP-ABE schemes based on the AND-gate with wildcard access structure. The existing schemes of this type need to use three different elements to represent the three possible values – positive, negative, and wildcard – of an attribute in the access structure. We propose a new construction which uses only one element to represent one attribute. The main idea behind our construction is to use the “positions” of different symbols to perform the matching between the access policy and user attributes. Specifically, We put the indices of all the positive, negative and wildcard attributes defined in an access structure into three sets, and by using the technique of Viète’s formulas [SLN⁺10], we allow the decryptor to remove all the wildcard positions, and perform the decryption correctly if and only if the remaining user attributes match those defined in the access structure. Our new technique leads to a new CP-ABE scheme with constant ciphertext size.

Although a secure ABE can well protect the secrecy of the encrypted data against unauthorised access, it does not protect the privacy of the receivers/decryptors by default. That is, given the ciphertext, an unauthorised user may still be able to obtain some information of the data recipients. For example, a health organization wants to send a message to all the patients that carry certain diseases. Then the attribute universe will contain all the diseases, and an access policy will have the format “+ − + − + − …” where “+” (“−”) indicates positive (negative) for a particular disease. If a CP-ABE cannot hide the access policy, then from the fact whether a person can decrypt the message or not, people can directly learn some sensitive information of the user. Therefore, it is also very important to hide the access policy in such applications. However, most of the existing ABE schemes based on AND-Gate with wildcard cannot achieve this property.

To address this problem, we further study the problem of hiding the access policy for CP-ABE based on AND-Gate with wildcard. As the main contribution of this work, we extend the technique we have used in the first construction to bridge ABE based on AND-Gate with wildcard with Inner Product Encryption (IPE) [KSW08a, SW08, ACP12]. Specifically, we present a way to convert an access
policy containing positive, negative, and wildcard symbols into a vector $\vec{X}$ which is used for encryption, and the user’s attributes containing positive and negative symbols into another vector $\vec{Y}$ which is used in key generation, and then apply the technique of IPE to do the encryption. Again, we use the positions of different symbols and the Viète’s formulas [SLN+10] to perform the conversion. The details are provided in Section 5.1.1.

### 3.1 Definition

#### 3.1.1 CP-ABE Definition

A ciphertext-policy attribute based encryption scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

- **Setup($\lambda, U$)**: The setup algorithm takes security parameters and attribute universe description as input. It outputs the public parameters $PK$ and a master key $MSK$.

- **Encrypt($PK, M, W$)**: The encryption algorithm takes as input the public parameters $PK$, a message $M$, and access structure $W$ over the universe of attributes, and outputs a ciphertext $CT$.

- **Key Generation($MSK, L$)**: The key generation algorithm takes as input the master key MSK and a set of attributes $L \subset U$, and outputs a private key $SK$.

- **Decrypt($PK, CT, SK$)**: The decryption algorithm takes as input the public parameters $PK$, a ciphertext $CT$, and a private key $SK$, and outputs a message $M$ or a special symbol ‘⊥’.

#### 3.1.2 Security Definition for CP-ABE with Hidden Access Policy

We define the Selective IND-CPA security for CP-ABE with hidden access policy via the following game.

- **Init**: The adversary commits to the challenge access policies $W_0, W_1$.

- **Setup**: The challenger runs the Setup algorithm and gives $PK$ to the adversary.
• **Phase 1**: The adversary submits the attribute list $L$ for a KeyGen query. If $(L \models W_0 \land L \models W_1) \lor (L \not\models W_0 \land L \not\models W_1)$, the challenger gives the adversary the secret key $SK_L$. The adversary can repeat this polynomially many times.

• **Challenge**: The adversary submits messages $M_0, M_1$ to the challenger. If the adversary obtained the $SK_L$ whose associated attribute list $L$ satisfies both $W_0$ and $W_1$ in Phase 1, then it is required that $M_0 = M_1$. The challenger flips a random coin $\beta$ and passes the ciphertext $\text{Encrypt}(PK, M_\beta, W_\beta)$ to the adversary.

• **Phase 2**: Phase 1 is repeated. If $M_0 \neq M_1$, the adversary cannot submit $L$ such that $L \models W_0 \land L \models W_1$.

• **Guess**: The adversary outputs a guess $\beta'$ of $\beta$.

We say a CP-ABE scheme with hidden access policy is secure if for any probabilistic polynomial-time adversary $A$,

$$\text{Adv}_{A}^{\text{IND-CPA}}(k) = |\Pr[\beta' = \beta] - \frac{1}{2}|$$

is negligible in the security parameter $k$.

**Full Security.** In the above selective security model, the adversary is required to commit the challenge policy before seeing the system parameters. In the full security model, the adversary can choose the challenge policy in the Challenge phase, which makes the model stronger. However, we cannot directly prove the security of our schemes in the full security model. We should note that there are transformations from the selective security to full security [LDL11], and we can apply the same transformation to our schemes presented in this chapter. However, the transformed schemes will be based on composite order pairing groups, and hence less efficient.

### 3.2 Constructions

#### 3.2.1 Efficient Ciphertext Policy Attribute Based Encryption Under Decisional Linear Assumption

In this section, we present our first scheme based on the AND-Gate with wildcard access policy. Below is the main idea of our construction.

We represent each attribute in the universe by an element $A_i$. Given an access structure $W$, we first define three sets $J, V,$ and $Z$ where $J$ contains the positions of all the wildcard positions, and $V$ and $Z$ contain the positions of all the positive and
negative attributes, respectively. We then represent each positive/negative attribute in an access structure as shown in the following figure.

The set \( J \) is attached to the ciphertext and sent to the decryptor. In the decryption process, based on \( J \), the decryptor can reconstruct the coefficients \( \lambda_{w_j} \), and generate

\[
\prod_{j \in J} (A_i)^{i\lambda_{w_j}} = (A_j)^{\prod_{j \in J} (i-w_j)}
\]

according to the Viète’s formulas, for each positive or negative attribute \( \text{Att}_i \) associated with the secret key. In this way, all the wildcard positions will take no effect during decryption. Below are the details of our construction.

**Setup**\((1^3)\): Let \( N_1, N_2, N_3 \) be three upper bounds defined as \( N_1 \leq L \): the maximum number of wildcard in an access structure; \( N_2 \leq L \): the maximum number of positive attribute in an attribute set \( S \); \( N_3 \leq L \): the maximum number of negative attribute in an attribute set \( S \).

The setup algorithm first generates bilinear groups \( \mathbb{G}, \mathbb{G}_T \) with order \( p \), and selects three random generators \( V_0, V_1, g \in \mathbb{G} \). Then randomly choose \( \alpha, \beta_1, \beta_2, a_1, \ldots, a_L \in_R \mathbb{Z}_p \), and set \( \Omega_1 = e(g, V_0)^{\alpha\beta_1}e(g, V_1)^{\alpha\beta_1}, \Omega_2 = e(g, V_0)^{\alpha\beta_2}e(g, V_1)^{\alpha\beta_2} \). Let \( A_i = g^{a_i} \) for \( 1 = 1, \ldots, L \).

The Public Key and Master Secret Key are defined as:

\[
PK = (e, g, \Omega_1, \Omega_2, g^a, V_0, V_1, A_1, \ldots, A_L),
MSK = (\alpha, \beta_1, \beta_2, a_1, \ldots, a_L).
\]

**Encrypt**\((W, M, PK)\): Suppose that the access structure \( W \) contains: \( n_1 \leq N_1 \) wildcards which occur at positions \( J = \{w_1, \ldots, w_{n_1}\} \); \( n_2 \leq N_2 \) positive attributes which occur at positions \( V = \{v_1, \ldots, v_{n_2}\} \); \( n_3 \leq N_3 \) negative attributes which occur at positions \( Z = \{z_1, \ldots, z_{n_3}\} \). Compute for the wildcard positions \( \{w_j\} \) \( (j = 0, 1, 2, \ldots, n_1) \) \{\( \lambda_{w_j} \)\} and set \( t_w = \sum_{j=0}^{n_1} \lambda_{w_j} \). The encryption algorithm then chooses \( r_1, r_2 \in_R \mathbb{Z}_p \), and creates the ciphertext as:

\[
C_0 = M \Omega_1^{r_1} \Omega_2^{r_2}, C_1 = g^{t_w}, C_2 = g^{t_w},
\]
the secret key as:

\[ C_3 = (V_0 \prod_{i \in V} (A_i^{\lambda_{w_j}})^{r_1 + r_2}, \quad C_4 = (V_1 \prod_{i \in Z} (A_i^{\lambda_{w_j}})^{r_1 + r_2}, \] 

The ciphertext is set as:

\[ CT = (C_0, C_1, C_2, C_3, J = \{w_1, w_2, \ldots, w_n\}). \]

**KeyGen**(*MSK, S*): Suppose that a user joins the system with the attribute list *S*, which contains: \( n'_2 \leq N_2 \) positive attributes which occur at positions \( V' = \{v'_1, \ldots, v'_n\} \); \( n'_3 \leq N_3 \) negative attributes which occur at positions \( Z' = \{z'_1, \ldots, z'_n\} \).

By means of the Viète’s formulas, for all the positive positions \( \{v'_k\} \) (\( k = 0, 1, 2, \ldots, n'_2 \)), calculate \( \{\lambda_{v'_k}\} \) and set \( t'_v = \sum_{k=0}^{n'_2} \lambda_{v'_k} \); and for all the negative positions \( \{z'_\tau\} \) (\( \tau = 0, 1, 2, \cdots, n'_3 \)), calculate \( \{\lambda_{z'_\tau}\} \) and set \( t'_z = \sum_{\tau=0}^{n'_3} \lambda_{z'_\tau} \). The algorithm then chooses \( s \in R \mathbb{Z}_p \) and computes \( s_1 = \beta_1 + s, s_2 = \beta_2 + s \) and creates the secret key as:

\[
L_1 = g^{\frac{a_s}{v}}, 
L_2 = g^{\frac{a_s}{z}}, 
K_1 = \{K_{1,0}, K_{1,1}, \ldots, K_{1,N_1}\} = \{V_0^{s_1} \prod_{i \in V'} g^{s_{a_i}}, V_0^{s_1} \prod_{i \in V'} g^{s_{a_i}^N_i}, \ldots, V_0^{s_1} \prod_{i \in V'} g^{s_{a_i}^{N_1}}\},
\]

\[
K'_1 = \{K'_{1,0}, K'_{1,1}, \ldots, K'_{1,N_1}\} = \{V_0^{s_2} \prod_{i \in V'} g^{s_{a_i}}, V_0^{s_2} \prod_{i \in V'} g^{s_{a_i}^N_i}, \ldots, V_0^{s_2} \prod_{i \in V'} g^{s_{a_i}^{N_1}}\},
\]

\[
K_2 = \{K_{2,0}, K_{2,1}, \ldots, K_{2,N_1}\} = \{V_1^{s_1} \prod_{i \in Z'} g^{s_{a_i}}, V_1^{s_1} \prod_{i \in Z'} g^{s_{a_i}^N_i}, \ldots, V_1^{s_1} \prod_{i \in Z'} g^{s_{a_i}^{N_1}}\},
\]

\[
K'_2 = \{K'_{2,0}, K'_{2,1}, \ldots, K'_{2,N_1}\} = \{V_1^{s_2} \prod_{i \in Z'} g^{s_{a_i}}, V_1^{s_2} \prod_{i \in Z'} g^{s_{a_i}^N_i}, \ldots, V_1^{s_2} \prod_{i \in Z'} g^{s_{a_i}^{N_1}}\}.
\]

The user secret key is set as:

\[ SK = (L_1, L_2, K_1, K'_1, K_2, K'_2). \]

**Decrypt**\((CT, SK)\): The algorithm first identifies the wildcard positions in \( J = \{w_1, \ldots, w_n\} \) and computes \( \{\lambda_{w_j}\} \). Then we have:

\[
e((L_1, C_0)^{e_L} \cdot (L_2, C_4)^{e_L}) = e((L_1, C_0)^{e_L} \cdot (L_2, C_4)^{e_L}) = e((L_1, C_0)^{e_L} \cdot (L_2, C_4)^{e_L}) = e((L_1, C_0)^{e_L} \cdot (L_2, C_4)^{e_L}) = e((L_1, C_0)^{e_L} \cdot (L_2, C_4)^{e_L})
\]
\[ e(g, V_0)^{-\alpha \beta_1 r_1} e(g, V_0)^{-\alpha \beta_2 r_2} e(g, V_1)^{-\alpha \beta_1 r_1} e(g, V_1)^{-\alpha \beta_2 r_2} = \Omega_1^{-r_1} \Omega_2^{-r_2} \]

and \( M \) can be recovered by \( \Omega_1^{-r_1} \Omega_2^{-r_2} \cdot C_0 \).

### 3.2.2 Hidden Ciphertext Policy Attribute Based Encryption under Standard Assumptions

Although the CP-ABE scheme presented in the previous section can achieve constant ciphertext size, it cannot hide the access policy since the wildcard positions need to be included in the ciphertext. In this section, we extend the technique used in our first construction to build another CPA-ABE which can hide the access policy.

One way to achieve the attribute hiding property is to apply the Inner Product Encryption technique in the construction of CP-ABE. Such an approach has been used in previous works on policy hiding CP-ABE [LOS+10],[LDL11],[CCL+13]. However, since our CP-ABE scheme is based on the Viète’s formula, we cannot directly use the previous approach. In this chapter, we propose a new transformation technique which can deal with the Viète’s formula.

**Our Idea:** Our main idea is to convert the access policy and user attributes into two vectors, and then apply the technique of Inner Product Encryption to hide the access policy. Similar to the first scheme, we separate the positive, negative, and wildcard symbols in an access structure into three sets: \( V, Z, \) and \( J \). Based on the set \( J \), by applying the Viète’s formulas, we can construct a polynomial \( \sum_{k=0}^{n} a_k i^k \) with coefficients \( (a_0, a_1, \ldots, a_n) \).

Then we combine the set of positive positions \( V \) as:

\[ \Pi_V = + \sum_{i \in V} \prod_{w_j \in J} (i - w_j) \]

and the set of negative positions \( Z \) as:

\[ \Pi_Z = - \sum_{i \in Z} \prod_{w_j \in J} (i - w_j). \]

We then produce a vector

\[ \vec{v} = (a_0, a_1, \ldots, a_n, 0_{n+1}, \ldots, 0_{N_1}, I_V, \Pi_Z) \]

which will be used for encryption.

In user key generation, we also separate the positive and negative attributes into
two sets and construct two vectors

\[ x_{V'} = (v'_0, v'_1, v'_2, \ldots, v'_{N_1}, 1, 0), \]
\[ x_{Z'} = (z'_0, z'_1, z'_2, \ldots, z'_{N_1}, 0, 1), \]

in which:

\[ v'_k = -\sum_{i \in V'} i^k, k = 0, \ldots, N_1, \]
\[ z'_k = +\sum_{i \in Z'} i^k, k = 0, \ldots, N_1. \]

Notice that we assume there are at most \( N_1 \) wildcard positions in an access policy. The decryption will be based on the inner products of \((\vec{v}, x_{V'})\) and \((\vec{v}, x_{Z'})\), which should both return 0 in order to have a successful decryption.

Let \( L = 4, N_1 = 2 \) and \( W_2 = (+, -, *, *) \) be the access policy. Then we create three sets for wildcard positions \( J = \{3, 4\} \), positive positions \( V = \{1\} \), and negative positions \( Z = \{2\} \). Based on Viète’s formulas, we can calculate

\[ a_2 = 1; a_1 = -7, a_0 = 12 \]

and obtain the vector \( \vec{v} \) for the access policy and the vectors for Alice and Bob as follows.

<table>
<thead>
<tr>
<th>( \vec{v} ) =</th>
<th>0.</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( = ( )</td>
<td>12.</td>
<td>-7.</td>
<td>1.</td>
<td>(+1-3)(1-4),</td>
<td>-(2-3)(2-4) )</td>
</tr>
</tbody>
</table>

**Figure 3.2:** The vector \( \vec{v} \) for access policy \( W_2 \)

<table>
<thead>
<tr>
<th>( \vec{x} ) =</th>
<th>0.</th>
<th>1.</th>
<th>2.</th>
<th>3.</th>
<th>4.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Alice_\vec{v} ) =</td>
<td>( -(1^0 + 4^2) )</td>
<td>( -(1^1 + 4^2) )</td>
<td>( -(1^2 + 4^2) ),</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( Alice_\vec{x} ) =</td>
<td>( (2^0 + 3^2) )</td>
<td>( (2^1 + 3^2) )</td>
<td>( (2^2 + 3^2) ),</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( Bob_\vec{v} ) =</td>
<td>( -(1^0 + 3^2) )</td>
<td>( -(1^1 + 3^2) )</td>
<td>( -(1^2 + 3^2) ),</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( Bob_\vec{x} ) =</td>
<td>( (1^0 + 4^2) )</td>
<td>( (1^1 + 4^2) )</td>
<td>( (1^2 + 4^2) ),</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 3.3:** The vector \( \vec{x} \) for Alice and Bob

If we calculate the inner product of \( \vec{v} \) and the two vectors of Alice, the product will return 0, i.e., Alice’s attributes satisfy the access policy \( W_2 \). On the other hand, the inner product of \((\vec{v}, Bob_\vec{v}) = 8 \) and \((\vec{v}, Bob_\vec{x}) = 4 \), which means Bob’s attributes cannot satisfy \( W_2 \).
**Setup** $(1^A)$: Assume that we have $L$ attributes in the universe, and each attribute has two possible values: positive and negative. In addition, we also consider wildcard (meaning “don’t care”) in access structures. Let $N_1, N_2, N_3$ be three upper bounds defined as:

- $N_1 \leq L$: the maximum number of wildcard in an access structure;
- $N_2 \leq L$: the maximum number of positive attribute in an attribute set $S$;
- $N_3 \leq L$: the maximum number of negative attribute in an attribute set $S$.

The setup algorithm first randomly generates $(g, \mathbb{G}, \mathbb{G}_T, p, e)$ and sets $n = N_1 + 3$. It then chooses randomly $\gamma_1, \gamma_2, \theta_1, \theta_2, \{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n$ in $\mathbb{Z}_p$ and $g_2 \in \mathbb{G}$. Then it selects a random $\Delta \in \mathbb{Z}_p$ and obtains $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, w_2$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i} \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.$$

For $i$ from 1 to $n$, it creates:

- $U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}$,
- $T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, Z_{1,i} = g^{z_{1,i}}$,
- $V_i = g^{\gamma_i}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_i}, V_2 = g^{\theta_2}$.

Next it sets $g_1 = g^A, Y = e(g, g_2)$, and the public key $PK$ and master key $MSK$ as

$$PK = (g, \mathbb{G}, \mathbb{G}_T, p, e, g_1, Y, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i}, W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}_{i=1}^n, \{V_i, X_i\}_{i=1}^2)$$

$$MSK = (g_2, \{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}_{i=1}^n, \{v_i, x_i\}_{i=1}^2).$$

**Encrypt** $(W, M, PK)$: Suppose that the access structure $W$ contains: $n_1 \leq N_1$ wildcards which occur at positions $J = \{w_{1,1}, \ldots, w_{n_1}\}$; $n_2 \leq N_2$ positive attributes which occur at positions $V = \{v_{1,1}, \ldots, v_{n_2}\}$; $n_3 \leq N_3$ negative attributes which occur at positions $Z = \{z_{1,1}, \ldots, z_{n_3}\}$. Based on Viète’s formulas, compute for the wildcard positions $\{w_j\}$ ($j = 0, 1, 2, \ldots, n_1$)

$$a_{n_1} = 1$$
$$a_{n_1-1} = -(w_1 + w_2 + \ldots + w_{n_1})$$
$$a_{n_1-2} = (w_1 w_2 + w_1 w_3 + \ldots + w_{n_1-1} w_{n_1})$$
$$\ldots$$
$$a_0 = -(w_1 \cdot w_2 \cdot \ldots \cdot w_{n_1})$$

Next it computes:

$$\Pi_V = + \sum_{i \in V} \prod_{w_j \in J} (i - w_j)$$

$$\Pi_Z = - \sum_{i \in Z} \prod_{w_j \in J} (i - w_j)$$
It creates a vector $\vec{v} = (v_1, v_2, \ldots, v_n)$ as:

$$\vec{v} = (a_0, a_1, \ldots, a_n, 0_{n_1+1}, \ldots, 0_{N_1}, \Pi_V, \Pi_Z).$$

The encryption algorithm chooses random $s_1, s_2, \alpha, \beta \in \mathbb{Z}_p$ and creates the ciphertext as follows:

$$C_m = M \cdot Y^{s_2}, C_A = g^{s_2}, C_B = g_1^{s_1},$$

$$\{C_{1,i}, C_{2,i}\} = \{U_{1,i}^{s_1}T_{1,i}^{s_2}V_{1,i}^{\alpha}, U_{2,i}^{s_1}T_{2,i}^{s_2}V_{2,i}^{\alpha}\},$$

$$\{C_{3,i}, C_{4,i}\} = \{W_{1,i}^{s_1}Z_{1,i}^{s_2}X_{1,i}^{\beta}, W_{2,i}^{s_1}Z_{2,i}^{s_2}X_{2,i}^{\beta}\},$$

Then ciphertext $CT$ is set as:

$$CT = (C_m, C_A, C_B, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i=1}^n).$$

**KeyGen**(MSK, S): Suppose that a user joins the system with the attribute list $S$, which contains: $n_2' \leq N_2$ positive attributes which occur at positions $V' = \{v_1', \ldots, v_{n_2'}\}$; $n_3' \leq N_3$ negative attributes which occur at positions $Z' = \{z_1', \ldots, z_{n_3}'\}$. By means of the Viète’s formulas, for all the positive positions $\{v_k'\} (k = 0, 1, 2, \ldots, n_2')$, for all the negative positions $\{z_k'\} (\tau = 0, 1, 2, \ldots, n_3')$, it sets:

$$v_k' = -\sum_{i\in V'} i^k, k = 0, \ldots, N_1$$

$$z_k' = +\sum_{i\in Z'} i^k, k = 0, \ldots, N_1$$

It creates vectors $\vec{x_V}$ and $\vec{x_Z}$ as:

$$\vec{x_V} = (v_0', v_1', \ldots, v_{N_1}, 1, 0),$$

$$\vec{x_Z} = (z_0', z_1', \ldots, z_{N_1}, 0, 1).$$

The key generation algorithm chooses randomly $r_{i,1}, r_{i,2}$ for $i = 1$ to $n$, and $f_1, f_2, r_1, r_2 \in \mathbb{Z}_p$, and then creates the secret key as follows:

$$\{K_{1,i}, K_{2,i}\} = \{g^{-x_{r_{i,1}}}, g^{x_{r_{i,2}}v_{u_2}}, g^{x_{r_{i,1}}}, g^{-f_{i}v_{u_1}}\},$$

$$\{K_{3,i}, K_{4,i}\} = \{g^{-x_{r_{i,1}}}, g^{x_{r_{i,2}}w_{u_2}}, g^{x_{r_{i,1}}}, g^{-f_{i}w_{u_1}}\},$$

$$K_A = g^{r_2}, \prod_{i=1}^{n} K_{1,i}^{-r_{i,1}}, K_{2,i}^{-r_{i,2}},$$

$$K_B = \prod_{i=1}^{n} g^{-(r_{i,1}+r_{i,2})}.$$ 

The secret key is set as:

$$SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n).$$

**Decrypt**(SK, CT): The decryption algorithm returns

$$\mathfrak{e}(C_A, K_A)\mathfrak{e}(C_B, K_B)\prod_{i=1}^{n} \mathfrak{e}(C_{ij}, K_{ij}).$$
Correctness:

\[ e(C_{1,i}, K_{1,i}) = e(U_{1,i}^{s_1} T_{2,i}^{s_2} V_{1}^{\alpha}, g^{-\gamma_1 r_{1,i} g f_1 v_i u_2, i}) \]
\[ = e(g, g)^{r_{1,i} s_1 (-u_1, i)} \cdot e(g, g)^{-\gamma_1 v_i \gamma_1 r_{1,i}} \cdot e(g, K_{1,i})^{t_1, s_2} \cdot e(g, g)^{f_1 v_i \alpha u_2, i} \cdot e(g, g)^{f_1 v_i \alpha v_i u_2, i}. \]

\[ e(C_{2,i}, K_{2,i}) = e(U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2}^{\alpha}, g^{-\gamma_1 r_{1,i} g f_1 v_i u_1, i}) \]
\[ = e(g, g)^{r_{1,i} s_1 u_2, i} \cdot e(g, g)^{r_{1,i} v_i \gamma_1 r_{1,i}} \cdot e(g, K_{2,i})^{t_2, s_2} \cdot e(g, g)^{-f_1 v_i \alpha u_2, i} \cdot e(g, g)^{-f_1 v_i \alpha v_i u_2, i}. \]

\[ \prod_{j=1}^{4} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i}) = \prod_{i=1}^{n} e(g, g)^{r_{1,i} s_1 \Delta} \cdot e(g, g)^{f_1 v_i u_n \Delta} \cdot e(g, K_{1,i})^{t_1, s_2} e(g, K_{2,i})^{t_2, s_2} e(g, K_{3,i})^{t_3, s_2} e(g, K_{4,i})^{t_4, s_2}. \]

Then we have:

\[ \prod_{j=1}^{4} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i}) = e(g, g)^{(\sum_i^x v_i) f_1 \alpha \Delta} e(g, g)^{(\sum_i^x z_i) f_2 \alpha \Delta} \prod_{i=1}^{n} e(g, K_{1,i})^{t_1, s_2} e(g, K_{2,i})^{t_2, s_2} e(g, K_{3,i})^{t_3, s_2} e(g, K_{4,i})^{t_4, s_2}. \]

Also, since

\[ e(C_A, K_A) = e(g^{r_2}, g_2 \cdot \prod_{i=1}^{n} K_{1,i}^{-t_{1,i}} K_{2,i}^{-t_{2,i}} K_{3,i}^{-t_{3,i}} K_{4,i}^{-t_{4,i}}) \]
\[ e(C_B, K_B) = e(g^{s_1 \Delta}, \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})}) \]

we have

\[ \frac{e(C_A, K_A) e(C_B, K_B) \prod_{j=1}^{4} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i})}{e(g, g)^{(\sum_i^x v_i) f_1 \alpha \Delta} + (\sum_i^x z_i) f_2 \alpha \Delta}} = \frac{M}{e(g, g)^{(\sum_i^x v_i) f_1 \alpha \Delta} + (\sum_i^x z_i) f_2 \alpha \Delta}}. \]

Therefore, the message \( M \) will be returned iff \( \langle \vec{v}, \vec{x} \rangle = 0 \) and \( \langle \vec{v}, \vec{x} \rangle = 0 \), meaning the attributes list in user key \( SK \) satisfies the access policy in the ciphertext \( CT \).

### 3.3 Security Proof

#### 3.3.1 Security Proof-Efficient CP-ABE Under Decisional Linear Assumption

**Theorem 4** Assume that the DLIN assumption holds in \( \mathbb{G} \), then no polynomial-time adversary can have a non-negligible advantage over random guess in the Select-
active IND-CPA security game.

Proof: Let $B$ denote an algorithm that is given $(g,g^a,g^b,g^{ac},g^d,T) \in \mathbb{G}^6$ as input. $B$'s goal is to decide $T = g^{b(c+d)}$ or $T = g^r$.

Init: The adversary gives $B$ the challenge access structure $W^* = [W^*_1, \ldots, W^*_n]$ which contains $n_1$ wildcards which occur at positions $J = \{w_1, \ldots, w_{n_1}\}$, $n_2$ positive attributes which occur at positions $V = \{v_1, \ldots, v_{n_2}\}$, $n_3$ negative attributes which occur at positions $Z = \{z_1, \ldots, z_{n_3}\}$ at the beginning of the game.

Setup: $B$ chooses an upper bound $n_1 \leq N_1 \leq L$ for the number of wildcard in an access structure, and then selects $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{Z}_p$. $B$ also selects $\gamma_0, \gamma_1, \{a'_i\}_{1 \leq i \leq L} \in \mathbb{R}^{N_p}$, and computes by means of the Viète’s formulas $\{\lambda_{w_j}\}_{w_j \in \mathbb{J}}$ and sets $t_w = \sum_{j=0}^{n_1} \lambda_{w_j}$. $B$ then sets

$$V_0 = (g^b)^{\gamma_0} - \sum_{i \in W^*_1 \in V} a'_i \prod_{j=1}^{(i-w_j)} \frac{1}{t_w}, V_1 = (g^b)^{\gamma_1} - \sum_{i \in W^*_2 \in Z} a'_i \prod_{j=1}^{(i-w_j)} \frac{1}{t_w}$$

and

$$A_1 = g^{a'_i} = \begin{cases} g^{a'_i}, & att_i = W^*_i \\ \frac{\sum_{i \in W^*_i} a'_i \prod_{j=1}^{(m-w_j)}}{g^{a'_i}}, & att_i \neq W^*_i \end{cases}$$

$B$ sets $\Omega_1 = e(g^a, V_0)^{\sigma_2}e(g^a, V_1)^{\sigma_1} - \sigma_2$ and $\Omega_2 = e(g^{a^3}(g^a)^{-\sigma_2}, V_0)e(g^{a^3}(g^a)^{-\sigma_2}, V_1)$, and the public key as:

$$PK = (e, g, \Omega_1, \Omega_2, g^a, V_0, V_1, A_1, \ldots, A_L)$$

and the corresponding master secret key is:

$$MSK = (\alpha = a, \beta_1 = \sigma_1 - \sigma_2, \beta_2 = \frac{\sigma_3}{a} - \sigma_2, a_1, \ldots, a_L)$$

Phase 1: $A$ submits an attribute list $L = [L_1, \ldots, L_L]$ in a secret key query, consisting of $n'_L \leq N_2$ positive attributes at positions $V = \{v_1, \ldots, v_{n'_L}\}$, and $n'_3 \leq N_3$ negative attributes at positions $Z = \{z_1, \ldots, z_{n'_3}\}$. $B$ computes:

- for all the positive positions, $\lambda_{v'_k}$ for $k = 0, 1, 2, \ldots, n'_L$, and $t'_v = \sum_{k=0}^{n'_L} \lambda_{v'_k}$;
- for all the negative positions, $\lambda_{z'_\tau}$ for $\tau = 0, 1, 2, \ldots, n'_3$, and $t'_z = \sum_{\tau=0}^{n'_3} \lambda_{z'_\tau}$. 

34
\( \mathcal{B} \) simulates the private key as:

\[
L_1 = \left( g^a \right)^{\frac{\sigma_2}{p^e}}, \quad L_2 = \left( g^a \right)^{\frac{\sigma_2}{q^e}},
\]

\[
K_1 = \{ K_{1,0}, K_{1,1}, \ldots, K_{1,N_1} \},
\]

\[
\left\{ V_{0}^{\sigma_1} \prod_{att_i \in W^+, i \in V} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in V} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j), \right\} \]

\[
V_{0}^{\sigma_1} \prod_{att_i \in W^+, i \in V} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in V} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
, \ldots, V_{0}^{\sigma_1} \prod_{att_i \in W^+, i \in V} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in V} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
K_1' = \{ K_{1,0}', K_{1,1}', \ldots, K_{1,N_1}' \},
\]

\[
\left\{ (g^b)^{\sigma_2 \gamma_0} g^{\sigma_3} \prod_{att_i \in W^+, i \in V} \sum_{j=1}^{n} a_m' (m-w_j), \right\} \]

\[
(g^b)^{\sigma_2 \gamma_0} g^{\sigma_3} \prod_{att_i \in W^+, i \in V} \sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
, \ldots, \frac{(g^b)^{\sigma_2 \gamma_0} g^{\sigma_3} \prod_{att_i \in W^+, i \in V} \sum_{j=1}^{n} a_m' (m-w_j)}{(g^a)^{\sigma_2 a_i'}} \prod_{att_i \in W^+, i \in V} \right\}
\]

\[
K_2 = \{ K_{2,0}, K_{2,1}, \ldots, K_{2,N_1} \}
\]

\[
= \left\{ V_{1}^{\sigma_1} \prod_{att_i \in W^+, i \in Z} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in Z} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j), \right\}
\]

\[
V_{1}^{\sigma_1} \prod_{att_i \in W^+, i \in Z} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in Z} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
, V_{1}^{\sigma_1} \prod_{att_i \in W^+, i \in Z} g^{\alpha_{2a_i'}} \prod_{att_j \in W^+, j \in Z} g^{\alpha_{2a_j'}} \prod_{att_m \in W^+} \sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
\sum_{j=1}^{n} a_m' (m-w_j)
\]

\[
\sum_{j=1}^{n} a_m' (m-w_j)
\]
Chapter 3. Ciphertext Policy Attribute Based Encryption under Standard Assumptions

\[ V_1^{\sigma_1} \prod_{\text{att}_i \in W^*, i \in Z} g^{\sigma_2 a_i^{N_1}} \prod_{\text{att}_i \notin W^*, i \in Z} g^{a_i^{N_1} \prod_{j=1}^{n_i} (m-w_j)} \]

\[ K'_2 = \{ K'_{2,0}, K'_{2,1}, \ldots, K'_{2,n_1} \} \]

\[ \{(g^b)^{\sigma_3 \gamma_1} g^{-\sigma_3} \prod_{\text{att}_i \in W^*, i \in Z} \prod_{a_i^{N_1} \prod_{j=1}^{n_i} (m-w_j)} (g^a)^{\sigma_2 a_i^{N_1}} \}
\]

\[ \prod_{\text{att}_i \notin W^*, i \in Z} \sum_{a_i^{N_1} \prod_{j=1}^{n_i} (m-w_j)} (g^a)^{\sigma_2 a_i^{N_1}} \]

\[ \sum_{a_i^{N_1} \prod_{j=1}^{n_i} (m-w_j)} (g^a)^{\sigma_2 a_i^{N_1}} \]

which implicitly sets \( s = \sigma_2 \). \( B \) returns to \( A \) the secret key \( SK = (L_1, L_2, K_1, K'_1, K_2, K'_2) \).

\textbf{Challenge:} The adversary gives two messages \( M_0 \) and \( M_1 \) to \( B \). Then \( B \) flips a coin \( \nu \) and outputs:

\[ C_0 = M_\nu e(g^{ac}, g^b)^{\sigma_1 \gamma_0} \cdot e(g^{ac}, g)^{\sigma_1 \gamma_0} \cdot \sum_{\text{att}_i \in W^*, i \in V} a_i^{\prod_{j=1}^{n_i} (i-w_j)} \]

\[ e(g^a, g^d) \sum_{\text{att}_i \in W^*, i \in V} a_i^{\prod_{j=1}^{n_i} (i-w_j)} \cdot e(g^b, g^d)^{\sigma_1 \gamma_0} \]

\[ e(g^{ac}, g^b)^{\sigma_1 \gamma_1} \cdot e(g^{ac}, g)^{\sigma_1 \gamma_0} \cdot \sum_{\text{att}_i \in W^*, i \in Z} a_i^{\prod_{j=1}^{n_i} (i-w_j)} \]

\[ e(g^a, g^d) \sum_{\text{att}_i \in W^*, i \in Z} a_i^{\prod_{j=1}^{n_i} (i-w_j)} \cdot e(g^b, g^d)^{\sigma_1 \gamma_1} \]

\[ e(g^d, g) \sum_{\text{att}_i \in W^*, i \in V} a_i^{\prod_{j=1}^{n_i} (i-w_j)} \cdot e(g^b, g^d)^{\sigma_1 \gamma_1} \]
\[ e(g^a, T_\nu)^{\sigma_2 \gamma_0} \cdot e(g^d, g)^{\sum_{i \in V} a'_i \prod_{j=1}^{n_i} (1-w_j)} \cdot e(g^a, T_\nu)^{\sigma_2 \gamma_1}, \]

\[
C_1 = (g^{ac})^{\frac{1}{tw}}, \quad C_2 = (g^d)^{\frac{1}{tw}},
\]

\[
C_3 = (V_0 \prod_{i \in V} (A_i^{tw})^{r_1+r_2})
\]

\[
= ((g^b)^{\gamma_0 g} = \sum_{i \in V} a'_i \prod_{j=1}^{n_i} (1-w_j) \cdot g^{a'i_{i \in V} c_{i \in V}}^{tw})^{c+d} = T_\nu^{\gamma_0},
\]

\[
C_4 = (V_1 \prod_{i \in Z} (A_i^{tw})^{r_1+r_2})
\]

\[
= ((g^b)^{\gamma_1 g} = \sum_{i \in Z} a'_i \prod_{j=1}^{n_i} (1-w_j) \cdot g^{a'i_{i \in Z} c_{i \in Z}}^{tw})^{c+d} = T_\nu^{\gamma_1},
\]

which implicitly sets \( r_1 = c \) and \( r_2 = d \).

**Phase II:** Same as Phase I.

**Guess:** The adversary will eventually output a guess \( \nu' \) of \( \nu \). The simulator outputs 0 to guess that \( Z = g^{b(c+d)} \) if \( \nu' = \nu \); otherwise, it outputs 1 to guess that \( Z \) is a random group element of \( G \).

If \( Z = g^{b(c+d)} \), the simulator \( B \) gives a perfect simulation so we have:

\[
\Pr[B(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^{b(c+d)}] = \frac{1}{2} + \text{Adv}_A(k).
\]

If \( Z \) is a random group element the message \( M_\beta \) is completely hidden from the adversary and we have:

\[
\Pr[B(1^k, g, g^a, g^b, g^{ac}, g^d, Z) = 1 | Z = g^\beta] = \frac{1}{2}.
\]

Therefore, \( B \) can solve DLIN with non-negligible advantage if \( \text{Adv}_A(k) \) is non-negligible.

### 3.3.2 Security Proof-Hidden CP-ABE

**Theorem 5** Assume the Decision Bilinear Diffie-Hellman assumption and Decisional Linear Assumption hold in group \( G \), then our Hidden CP-ABE scheme is selective IND-CPA secure and policy hiding.

Since our scheme actually uses the vector corresponding to an access policy to do the encryption. In order to prove that our scheme is policy hiding, we only need to prove that the adversary cannot tell which vector, among the two vectors \( \vec{u} \) and \( \vec{x} \)
corresponding to \( W_0 \) and \( W_1 \) respectively, has been used to generate the ciphertext. In our proof we will consider two cases \( M_0 = M_1 \) and \( M_0 \neq M_1 \).

In the case \( M_0 = M_1 \), we only consider the following game sequence from Game_1 to Game_5. In this case, we only prove the property of attribute hiding. For the other case \( M_0 \neq M_1 \), we need to consider the whole proof from Game_0 to Game_6. Below we first give a high level description of each game. In each game, we separate the vector used to generate \((C_A, C_B, C_{1,i}, C_{2,i})\) from the vector for \((C_A, C_B, C_{3,i}, C_{4,i})\). However, the same vector is used for both parts in Game_0 and Game_6.

**Game_0**: The challenge ciphertext \( CT_0 \) is generated under \((\vec{v}, \vec{v})\) and \( M_0 \). The ciphertext \( CT_0 \) is computed as follows:

\[
(M_0 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}V_{1}^{v_i}, U_2^{s_i}T_{2,1}^{i}V_{2}^{v_i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}\gamma_1^{v_i}, W_2^{s_i}Z_{2,1}^{i}\gamma_1^{v_i}\}_{i=1}^{n})
\]

**Game_1**: The challenge ciphertext \( CT_1 \) is generated under \((\vec{v}, \vec{v})\) and a random message \( R \in \mathbb{G}_T \). The ciphertext \( CT_1 \) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}V_{1}^{v_i}, U_2^{s_i}T_{2,1}^{i}V_{2}^{v_i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}\gamma_1^{v_i}W_2^{s_i}Z_{2,1}^{i}\gamma_2^{v_i}\}_{i=1}^{n})
\]

**Game_2**: The challenge ciphertext \( CT_2 \) is generated under \((\vec{v}, \vec{0})\) and a random message \( R \in \mathbb{G}_T \). The ciphertext \( CT_2 \) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}V_{1}^{v_i}, U_2^{s_i}T_{2,1}^{i}V_{2}^{v_i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}, W_2^{s_i}Z_{2,1}^{i}\}_{i=1}^{n})
\]

**Game_3**: The challenge ciphertext \( CT_3 \) is generated under \((\vec{v}, \vec{x})\) and a random message \( R \in \mathbb{G}_T \). The ciphertext \( CT_3 \) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}V_{1}^{v_i}, U_2^{s_i}T_{2,1}^{i}V_{2}^{v_i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}\gamma_1^{x_i}, W_2^{s_i}Z_{2,1}^{i}\gamma_2^{x_i}\}_{i=1}^{n})
\]

**Game_4**: The challenge ciphertext \( CT_4 \) is generated under \((\vec{0}, \vec{x})\) and a random message \( R \in \mathbb{G}_T \). The ciphertext \( CT_4 \) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}, U_2^{s_i}T_{2,1}^{i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}\gamma_1^{x_i}, W_2^{s_i}Z_{2,1}^{i}\gamma_2^{x_i}\}_{i=1}^{n})
\]

**Game_5**: The challenge ciphertext \( CT_5 \) is generated under \((\vec{x}, \vec{x})\) and a random message \( R \in \mathbb{G}_T \). The ciphertext \( CT_5 \) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_1^{s_i}T_{1,1}^{i}V_{1}^{x_i}, U_2^{s_i}T_{2,1}^{i}V_{2}^{x_i}\}_{i=1}^{n}, \{W_1^{s_i}Z_{1,1}^{i}\gamma_1^{x_i}, W_2^{s_i}Z_{2,1}^{i}\gamma_2^{x_i}\}_{i=1}^{n})
\]

**Game_6**: The challenge ciphertext \( CT_6 \) is generated under \((\vec{x}, \vec{x})\) and message \( M_1 \in \)
The ciphertext $\mathbf{CT}_0$ is computed as follows:

$$(M_1 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1},
\{U_{1,i}^{s_1} T_{1,i}^{s_2} V_1^{\gamma_1}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_2^{\gamma_2}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{\gamma_1}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{\gamma_2}\}_{i=1}^n)$$

**Indistinguishability between Game$_0$ and Game$_1$**

Suppose that there exists an adversary $A$ which can distinguish the two games with a non-negligible advantage $\epsilon$, we construct another algorithm $B$ which uses $A$ to solve the Decision Bilinear Diffie-Hellman problem also with advantage $\epsilon$. On input $(g, A = g^a, B = g^b, C = g^c, Z) \in \mathbb{G}_4$, $B$ simulates the game for $A$ as follows.

- **Setup:** $B$ selects random elements $\gamma_1, \gamma_2, \theta_1, \theta_2, \lambda, \{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n,
\{w_{1,i}\}_{i=1}^n, \{v_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n$ in $\mathbb{Z}_p$.

Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.$$

Then for $i = 1$ to $n$, $B$ sets:

$$U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, T_{1,i} = (g^b)^{v_{1,i}} g^{t_{1,i}}, T_{2,i} = (g^b)^{v_{2,i}} g^{t_{2,i}},
W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}, Z_{1,i} = (g^b)^{v_{1,i}} g^{t_{1,i}}, Z_{2,i} = (g^b)^{v_{2,i}} g^{t_{2,i}}.$$ 

and

$$V_1 = g^{v_1}, V_2 = g^{v_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2}, g_1 = g^\lambda, Y = e(g^a, g^b)^{-\Delta} \cdot e(g, g)^\lambda.$$ 

Each public key component is distributed properly following the random exponents:

$$t_{1,i} = v_{1,i} b + t_{1,i}, \quad t_{2,i} = v_{1,i} g_1 b + t_{2,i}, \quad z_{1,i} = v_{1,i} \theta_1 b + z_{1,i}, \quad z_{2,i} = v_{1,i} \theta_2 b + z_{2,i},$$

and

$$g_2 = g^{-ab\Delta} g^\lambda.$$ 

- **Key Generation Phase 1 & 2:** $A$ issues private key queries for the attribute list $L$. Consider a query with two vectors $y^V = (y_{V_1}, \ldots, y_{V_n})$ and $y^Z = (y_{Z_1}, \ldots, y_{Z_n})$. $A$ can request the private key query as long as $(\overrightarrow{V},\overrightarrow{Z}) = (\overrightarrow{V}^*,\overrightarrow{Z}^*) = \gamma_Y \neq 0$.

$B$ picks random exponents $\{r_{1,i}\}_{i=1}^n, \{r_{2,i}\}_{i=1}^n$, and $f'_{1}, f'_{2}, r_{1}, r_{2}$. Then $B$ computes:

$$K_{1,i} = g^{-\gamma_{2r_{1,i}}} g^{\left(\frac{\alpha}{\alpha y} + f'\right)y_{V_1} u_{2,i}}, \quad K_{2,i} = g^{\gamma r_{1,i}} g^{\left(\frac{\alpha}{\alpha y} + f'\right)y_{V_1} u_{1,i}},$$

$$K_{1,i}' = g^{-\gamma_{2r_{1,i}}} g^{\left(\frac{\alpha}{\alpha y} + f'\right)y_{V_1} u_{2,i}}, \quad K_{2,i}' = g^{\gamma r_{1,i}} g^{\left(\frac{\alpha}{\alpha y} + f'\right)y_{V_1} u_{1,i}}.$$
which implicitly sets: \( f_1 = \frac{a}{\overline{f}_y} + f'_1 \). Next \( B \) computes:

\[
K_{3,i} = g^{\theta_2 r_2,i} g^{\left( \frac{a}{\overline{f}_y} + f'_1 \right) y z, w_2,i} = g^{\frac{a}{\overline{f}_y} y z, w_2,i} g^{-\theta_2 r_2,i} g f'_2 y z, w_2,i = g^{\frac{a}{\overline{f}_y} y z, w_2,i} \cdot K'_{3,i}.
\]

\[
K_{4,i} = g^{\theta_1 r_2,i} g^{-\left( \frac{a}{\overline{f}_y} + f'_1 \right) y z, w_1,i} = g^{-\frac{a}{\overline{f}_y} y z, w_1,i} g^{\theta_1 r_1,i} g f'_2 y z, w_1,i = g^{-\frac{a}{\overline{f}_y} y z, w_1,i} \cdot K'_{4,i},
\]

which implicitly sets: \( f_2 = \frac{a}{\overline{f}_y} + f'_2 \). Then \( B \) and \( A \) are computed as:

\[
K_B = \prod_{i=1}^{n_1} g^{-(r_1,i + r_2,i)}
\]

\[
K_A = g_2 \prod_{i=1}^{n_1} K_{1,i}^{-r_1,i} K_{2,i}^{-r_2,i} K_{3,i}^{-\overline{r}_1,i} K_{4,i}^{-\overline{r}_2,i}.
\]

For \( A \), we can compute:

\[
K_{1,i}^{-t_1,i} K_{2,i}^{-t_2,i} = (g^{\frac{a}{\overline{f}_y} y v_i, u_2,i} \cdot K'_{1,i})^{-t_1,i} \cdot (g^{\frac{a}{\overline{f}_y} y v_i, u_1,i} \cdot K'_{2,i})^{-t_1,i}
\]

\[
= (g^{\frac{a}{\overline{f}_y} y v_i, u_2,i})^{-(v_i t_1,i)} \cdot (K'_{1,i})^{-t_1,i} \cdot (K'_{2,i})^{-t_1,i} \cdot (g^{\frac{a}{\overline{f}_y} y v_i, u_1,i})^{-(v_i t_1,i)}
\]

\[
= g^{\frac{a}{\overline{f}_y} v_i y v_i (u_1, t_2,i - u_2, t_1,i)} \cdot (K'_{1,i})^{-t_1,i} \cdot (K'_{2,i})^{-t_1,i}.
\]

Similarly, we can compute:

\[
K_{3,i}^{-\overline{r}_1,i} K_{4,i}^{-\overline{r}_2,i} = g^{\frac{a}{\overline{f}_y} v_i y z_i (u_1, z_2,i - u_2, z_1,i)} \cdot (K'_{3,i})^{-\overline{r}_1,i} \cdot (K'_{4,i})^{-\overline{r}_2,i}.
\]

Since \( g_2 = g^{ab_\overline{y}} \), then \( K_A \) can be computed as:

\[
K_A = g^\lambda \prod_{i=1}^{n_1} g^{\frac{a}{\overline{f}_y} y v_i (u_1, t_2,i - u_2, t_1,i)} g^{\frac{a}{\overline{f}_y} y z_i (u_1, z_2,i - u_2, z_1,i)}
\]

\[
\cdot (K'_{1,i})^{-t_1,i} \cdot (K'_{2,i})^{-t_2,i} \cdot (K'_{3,i})^{-\overline{r}_1,i} \cdot (K'_{4,i})^{-\overline{r}_2,i}.
\]

\( B \) gives \( A \) the private key: \( SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^{n_1}) \) for the queried vector \( \bar{y} \).

- **Challenge Ciphertext**: To generate a challenge ciphertext, \( B \) picks random \( s_1, c, \alpha, \beta \in \mathbb{Z}_p \). \( B \) implicitly sets:

\[
s_1 = s'_1, s_2 = c, \alpha = -bc + \alpha', \beta = -bc + \beta'.
\]

Then \( B \) sets \( A = g^c = g^{s_2}, B = g^{\Delta s_1} = g^{s_1} \). For \( i \) from 1 to \( n \), \( B \) computes:
Next \( B \) computes \( C_m = Z^\Delta \cdot e(g, g')^\lambda \cdot M_0 \). If \( Z = e(g, g')^{abc} \) the challenge ciphertext is distributed in \( \text{Game}_0 \), otherwise if \( Z \) is randomly chosen in \( \mathbb{G}_T \), then the challenge ciphertext is distributed in \( \text{Game}_1 \). Hence, if \( A \) can distinguish these two games, \( B \) can solve the DBDH problem.

**Indistinguishability between Game1 and Game2**

Suppose that there exists an adversary \( A \) which can distinguish these two games with non-negligible advantage \( \epsilon \), we construct another algorithm \( B \) which uses \( A \) to solve the Decision Linear problem with advantage \( \epsilon \). On input \((g, g'^a, g'^b, g'^c, g'^d, Z) \in \mathbb{G}_0\), \( B \) simulates the game for \( A \) as follows.

- **Setup:** \( B \) selects random elements \( \gamma_1, \gamma_2, \theta_1, \theta_2, \lambda, \eta_1, \eta_2 \),
  \( \{u_{1,i}\}^n_{i=1}, \{t_{1,i}\}^n_{i=1}, \{t_{2,i}\}^n_{i=1}, \{w_{1,i}\}^n_{i=1}, \{z_{1,i}\}^n_{i=1}, \{z_{2,i}\}^n_{i=1} \) in \( \mathbb{Z}_p \). Then it selects a random \( \Delta \in \mathbb{Z}_p \) to obtain \( \{u_{2,i}\}^n_{i=1}, \{w_{2,i}\}^n_{i=1}, w_2, w_2 \) under the condition:

\[
\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i},
\]

Then for \( i = 1 \) to \( n \), \( B \) sets:

\[
U_{1,i} = (g')^{u_{1,i}}, U_{2,i} = (g')^{u_{2,i}}, T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}},
\]

\[
W_{1,i} = (g')^{w_{1,i}}, W_{2,i} = (g')^{w_{2,i}}, V_{1,i} = g^{v_{1,i}}, V_{2,i} = g^{v_{2,i}}, \quad \text{for } v_{1,i}, v_{2,i}, g_1 = (g^\lambda)^{\Delta_i}, g_2 = g.
\]

Each public key component is distributed properly following the random exponents:

\[
\overline{u}_{1,i} = au_{1,i}, \quad \overline{u}_{2,i} = au_{2,i}, \quad \overline{w}_{1,i} = aw_{1,i} + \theta_1 b v_i, \quad \overline{w}_{2,i} = aw_{2,i} + \theta_2 b v_i,
\]

\[
\overline{z}_{1,i} = v_i \theta_1 b + z_{1,i}, \quad \overline{z}_{2,i} = v_i \theta_2 b + z_{2,i}.
\]

- **Key Generation Phase 1 & 2:** \( A \) issues private key queries for the attribute list \( L \). Consider a query will be created two vectors \( y'_V = (y_{V_1}, \ldots, y_{V_n}) \) and \( y'_Z = (y_{Z_1}, \ldots, y_{Z_m}) \). \( B \) picks random exponents \( \{r'_{1,i}\}^n_{i=1}, \{r'_{2,i}\}^n_{i=1} \) and \( f_1, f_2 \).

Then \( B \) computes:

\[
K_{1,i} = g^{-\gamma_2 r_{1,i}} g^f_{y_{V_1} u_{2,i}} = g^{-\gamma_2 r_{1,i}} g^f_{y_{V_1} u_{2,i}} g^{-\gamma_2 r_{1,i}} g^f_{y_{V_1} u_{2,i}} = g^{-\gamma_2 r_{1,i}} g^f_{y_{V_1} u_{2,i}} \cdot K_{1,i}^r,
\]

\[
K_{2,i} = g^{-\gamma_2 r_{2,i}} g^{-f_{y_{V_1} u_{1,i}}} = g^{-\gamma_2 r_{2,i}} g^{-f_{y_{V_1} u_{1,i}}} = g^{-\gamma_2 r_{2,i}} g^{-f_{y_{V_1} u_{1,i}}} = g^{-\gamma_2 r_{2,i}} g^{-f_{y_{V_1} u_{1,i}}} \cdot K_{2,i}^r,
\]
which implicitly sets: $r_{1,i} = -y_iv_i b + r_{1,i}'$. Next $\mathcal{B}$ computes:

$$K_{3,i} = g^{-\theta_2(v_i y_i b + ar_{1,i}')} g^{f_2 y_i z_i w_{2,i}} = g^{-\theta_2 v_i y_i b} g^{-\gamma r_{1,i}'} a g^{f_2 y_i z_i w_{2,i}} = g^{-\theta_2 v_i y_i b} \cdot K'_{3,i},$$

$$K_{4,i} = g^{\theta_1(v_i y_i b + ar_{1,i}')} g^{-f_2 y_i z_i w_{1,i}} = g^{\theta_1 v_i y_i b} g^{\theta_2 r_{1,i}} g^{-f_2 y_i z_i w_{1,i}} = g^{\theta_1 v_i y_i b} \cdot K'_{4,i},$$

which implicitly sets: $r_{2,i} = y_i v_i b + ar_{1,i}'$.

Then $K_B$ and $K_A$ are computed as:

$$K_B = \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})} \prod_{i=1}^{n} g^{-(y_i v_i b + r_{1,i}' + y_i v_i b + ar_{2,i}')} = g^{-(r_{1} + ar_{2})} \prod_{i=1}^{n} g^{-(r_{1,i}' + ar_{2,i}')},$$

$$K_A = g_2 \prod_{i=1}^{n} K_{1,i}^{4} K_{2,i}^{4} K_{3,i}^{4} K_{4,i}^{4} K_{5,i}^{4} K_{6,i}^{4} K_{7,i}^{4} K_{8,i}^{4}.$$ 

For $K_A$, we can compute:

$$K_{1,i}^{4} K_{2,i}^{4} = g^{-\gamma r_{1,i}'} g^{\gamma r_{1,i}'} (K'_{1,i}^{4})^{-t_{1,i}} \cdot (K'_{2,i}^{4})^{-t_{2,i}} \cdot K_{3,i}^{4} K_{4,i}^{4}.$$ 

Since $g_2 = g^\lambda$ then $K_A$ is computed as:

$$K_A = g^\lambda \prod_{i=1}^{n} g^{-\gamma v_i y_i b + ar_{1,i}'} g^{\gamma v_i y_i b + ar_{2,i}'} (K'_{1,i}^{4})^{-t_{1,i}} \cdot (K'_{2,i}^{4})^{-t_{2,i}} \cdot g^{-(y_i v_i b + ar_{1,i}') + \lambda} g^{(f_2 y_i z_i w_{2,i})} (K'_{3,i}^{4})^{-t_{1,i}} \cdot (K'_{4,i}^{4})^{-t_{2,i}} \cdot g^{-(f_2 y_i z_i w_{1,i})}.$$ 

$\mathcal{B}$ gives $\mathcal{A}$ the private key: $SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^{n})$ for the queried vector $\vec{y}$.

- **Challenge Ciphertext**: To generate a challenge ciphertext, $\mathcal{B}$ picks random $s_{1}', \alpha' \in \mathbb{Z}_p$. $\mathcal{B}$ implicitly sets:

$$s_1 = c, s_2 = d, \alpha = \alpha'.$$

Then $\mathcal{B}$ sets: $A = g^d = g^s, B = (g^\alpha)\Delta = g_1^{s_1}$. For $i$ from 1 to $n$, $\mathcal{B}$ computes:

$$C_{1,i} = (g^{a_{1,i}})^c (g^d)^{z_{1,i}} g^{a_{1,i} \gamma_1 (\alpha')} = U_{1,i}^c T_{1,i}^{c z_{1,i}},$$

$$C_{2,i} = (g^{a_{2,i}})^c (g^d)^{z_{2,i}} g^{a_{2,i} \gamma_2 (\alpha')} = U_{2,i}^c T_{2,i}^{c z_{2,i}}.$$ 

Next $\mathcal{B}$ computes for $i$ from 1 to $n$:

$$C_{3,i} = (g^{a_{1,i}})^c (g^d)^{z_{1,i}} Z^{\theta_1 v_i}, C_{4,i} = (g^{a_{2,i}})^c (g^d)^{z_{2,i}} Z^{\theta_2 v_i}.$$
If $Z = g^{b(c+d)g^r}$ for $r$ chosen randomly in $\mathbb{Z}_p$, then $B$ is simulating Game$_1$ with $\beta = r$:

\[
\begin{align*}
C_{3,i} &= (g^{aw_1.x})^c (g^d)^{z_1,i} (g^{b(c+d)g^r})^{\theta_1 v_i} = W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,i}^{\beta}, \\
C_{4,i} &= (g^{aw_2.x})^c (g^d)^{z_2,i} (g^{b(c+d)g^r})^{\theta_2 v_i} = W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,i}^{\beta}.
\end{align*}
\]

If $Z = g^{b(c+d)}$, then $B$ is simulating Game$_2$

\[
\begin{align*}
C_{3,i} &= (g^{aw_1.x})^c (g^d)^{z_1,i} (g^{b(c+d)})^{\theta_1 v_i} = W_{1,i}^{s_1} Z_{1,i}, \\
C_{4,i} &= (g^{aw_2.x})^c (g^d)^{z_2,i} (g^{b(c+d)})^{\theta_2 v_i} = W_{2,i}^{s_1} Z_{2,i}.
\end{align*}
\]

Therefore, if $A$ can distinguish the two games, $B$ can solve the DLIN problem.

**Indistinguishability of Game$_2$ and Game$_3$**

Suppose that there exists an adversary $A$ which can distinguish these two games with a non-negligible advantage $\epsilon$, we construct another algorithm $B$ that uses $A$ to solve the Decision Linear problem with advantage $\epsilon$. On input $(g, g^a, g^b, g^c, g^d, Z) \in \mathbb{G}_6$, $B$ simulates the game for $A$ as follows.

- **Setup:** $B$ selects random elements $\gamma_1, \gamma_2, \theta_1, \theta_2, \lambda, \{u_{1,i}\}_{i=1}^n, \{u_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n$ in $\mathbb{Z}_p$. Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_1, w_2$ under the condition:

$$\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.$$

Then for $i = 1$ to $n$, $B$ sets:

\[
\begin{align*}
U_{1,i} &= (g^{a})^{u_{1,i})}, U_{2,i} = (g^{a})^{u_{2,i}}, T_{1,i} = g^{f_{1,i}}, T_{2,i} = g^{f_{2,i}}, \\
W_{1,i} &= (g^{a})^{w_{1,i}} (g^b)^{\theta_1 v_i}, W_{2,i} = (g^{a})^{w_{2,i}} (g^b)^{\theta_2 v_i}, \\
Z_{1,i} &= g^{z_{1,i}} (g^b)^{\theta_1 v_i}, Z_{2,i} = g^{z_{2,i}} (g^b)^{\theta_2 v_i}, V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2}, \\
g_1 &= g^{(a)^{\Delta}}, g_2 = g^{\lambda}.
\end{align*}
\]

Each public key component is distributed properly following the random exponents:

\[
\begin{align*}
\overline{u_{1,i}} &= au_{1,i}, \overline{w_{2,i}} = au_{2,i}, \overline{w_{1,i}} = aw_{1,i} + \theta_1 bv_i, \overline{w_{2,i}} = aw_{2,i} + \theta_2 bx_i, \\
\overline{z_{1,i}} &= v_i \theta_1 b + z_{1,i}, \overline{z_{2,i}} &= v_i \theta_2 b + z_{2,i}.
\end{align*}
\]

- **Key Generation Phase 1 & 2:** $A$ issues private key queries for the attribute list $L$. Consider a query will be created two vectors $\overline{y_V} = (y_{v_1}, \ldots, y_{v_n})$ and $\overline{y_Z} = (y_{z_1}, \ldots, y_{z_n})$. Notice that $A$ obey the restrictions defined in the model. That is $(\overline{v}, \overline{y_V}) = (\overline{v}, \overline{y_Z}) = 0$ mod $p$ if and only if $(\overline{x}, \overline{y_V})$ mod $p$ and $(\overline{x}, \overline{y_Z})$ mod $p$. There are two cases we need to consider.

  - **Case 1:** $(\overline{v}, \overline{y_V}) = 0 = (\overline{x}, \overline{y_Z})$ mod $p$. In this case, $B$ picks random
exponents \( \{r'_{1,i}\}_{i=1}^n, \{r'_{2,i}\}_{i=1}^n \), and \( f_1, f_2 \). Then \( \mathcal{B} \) computes:

\[
\begin{align*}
K_{1,i} &= g^{γ_2(−v_i(y_i)b+r'_{1,i})} g^{f_1(y_i)} u_{2,i} = g^{γ_2v_i(y_i)b} g^{−γ_2r'_{1,i}} g^{f_1(y_i)} u_{2,i} \\
&= g^{γ_2v_i(y_i)b} \cdot K'_{1,i}, \\
K_{2,i} &= g^{γ_2(−v_i(y_i)b+r'_{1,i})} g^{f_1(y_i)} u_{1,i} = g^{γ_2v_i(y_i)b} g^{−γ_2r'_{1,i}} g^{f_1(y_i)} u_{1,i} \\
&= g^{γ_2v_i(y_i)b} \cdot K'_{2,i},
\end{align*}
\]

which implicitly sets: \( r_{1,i} = −y_i v_i b + r'_{1,i} \).

Next \( \mathcal{B} \) computes:

\[
\begin{align*}
K_{3,i} &= g^{−θ_2(x_i(y_i) b + ar'_{2,i})} g^{f_2(y_i)} u_{2,i} = g^{−θ_2x_i(y_i) b} g^{−γ_2r'_{2,i}} g^{f_2(y_i)} u_{2,i} \\
&= g^{−θ_2x_i(y_i) b} \cdot K'_{3,i}, \\
K_{4,i} &= g^{θ_1(x_i(y_i) b + ar'_{2,i})} g^{f_2(y_i)} u_{1,i} = g^{θ_1x_i(y_i) b} g^{θ_2r'_{2,i}} g^{f_2(y_i)} u_{1,i} \\
&= g^{θ_1x_i(y_i) b} \cdot K'_{4,i},
\end{align*}
\]

which implicitly sets: \( r_{2,i} = x_i y_i b + ar'_{2,i} \).

\( \mathcal{B} \) also compute \( K_A \) and \( K_B \) as follows.

\[
\begin{align*}
K_B &= \prod_{i=1}^n g^{−(r_{1,i} + r_{2,i})} = \prod_{i=1}^n g^{−(r'_{1,i} + ar'_{2,i})}, \\
K_A &= g_2 \prod_{i=1}^n K_{1,i}^{−1,i} K_{2,i}^{−2,i} K_{3,i}^{−3,i} K_{4,i}^{−4,i}.
\end{align*}
\]

For \( K_A \), its components are computed as follows:

\[
\begin{align*}
K_{1,i}^{−t_{1,i}, K_{2,i}^{−t_{2,i}}} &= g^{γ_2v_i y_i b d_{2,i}} g^{−γ_1v_i y_i b d_{2,i}} \cdot (K'_{1,i})^{−t_{1,i}} \cdot (K'_{2,i})^{−t_{2,i}}, \\
K_{3,i}^{−3,i}, K_{4,i}^{−4,i} &= g^{−θ_2(x_i y_i b) (−z_{1,i} − θ_1 b x_i)} g^{−γ_2 r'_{2,i} (−z_{1,i} − θ_1 b x_i)} g^{−θ_2 r'_{2,i} (−z_{2,i} − θ_2 b x_i)} \\
&= g^{γ_2 v_i y_i b + ar'_{2,i}} \Delta g^{f_2 y_i w_{2,i}} (−z_{1,i} − θ_1 b x_i) g^{−f_2 y_i w_{1,i}} (−z_{2,i} − θ_2 b x_i),
\end{align*}
\]

Since \( g_2 = g^λ \) then \( K_A \) can be computed as:

\[
\begin{align*}
K_A &= g^λ \prod_{i=1}^n g^{γ_2v_i y_i b d_{2,i}} g^{−γ_1v_i y_i b d_{2,i}} \cdot (K'_{1,i})^{−t_{1,i}} \cdot (K'_{2,i})^{−t_{2,i}} \\
&\quad \cdot g^{−(x_i y_i b + ar'_{2,i})} \Delta g^{f_2 y_i w_{2,i}} (−z_{1,i} − θ_1 b x_i) \cdot g^{−f_2 y_i w_{1,i}} (−z_{2,i} − θ_2 b x_i) .
\end{align*}
\]

\( \mathcal{B} \) gives \( \mathcal{A} \) the private key \( SK = (K_A, K_B; \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\})_{i=1}^n \) for the queried vector \( \vec{y} \).

- **Case 2:** \((\vec{v}, \vec{y}_V) = c_v \neq 0 \) and \((\vec{x}, \vec{y}_Z) = c_x \neq 0 \). In this case, \( \mathcal{B} \) picks
random exponents \( \{r_{1,i}'\}_{i=1}^n, \{r_{2,i}'\}_{i=1}^n \), and \( f_1, f_2 \). Then \( B \) computes:

\[
K_{1,i} = g^{-\gamma_1(c^i v_i y_i b + r_{1,i})} g^{f_1 y_i u_i} = g^{-\gamma_1 c^i v_i y_i b} g^{-r_{1,i}} g^{f_1 y_i u_i}
\]
\[
= g^{-\gamma_1 c^i v_i y_i b} \cdot K_{1,i}'
\]
\[
K_{2,i} = g^{-\gamma_1(c^i v_i y_i b + r_{1,i})} g^{-f_1 y_i u_i} = g^{-\gamma_1 c^i v_i y_i b} g^{-r_{1,i}} g^{-f_1 y_i u_i}
\]
\[
= g^{-\gamma_1 c^i v_i y_i b} \cdot K_{2,i}'
\]

which implicitly sets: \( r_{1,i} = -c^i x_i y_i b + r_{1,i}' \).

Next \( B \) computes:

\[
K_{3,i} = g^{-\theta_2 c^i (x_i y_i z_i b + r_{2,i})} g^{f_2 y_i z_i w_{2,i}} = g^{-\theta_2 c^i x_i y_i b} g^{-r_{2,i}'} a g^{f_2 y_i z_i w_{2,i}}
\]
\[
= g^{-\theta_2 c^i x_i y_i b} \cdot K_{3,i}'
\]
\[
K_{4,i} = g^{\theta_1 c^i (x_i y_i z_i b + r_{2,i})} g^{-f_2 y_i z_i w_{1,i}} = g^{\theta_1 c^i x_i y_i b} g^{r_{2,i}'} g^{-f_2 y_i z_i w_{1,i}}
\]
\[
= g^{\theta_1 c^i x_i y_i b} \cdot K_{4,i}'
\]

which implicitly sets: \( r_{2,i} = c^i x_i y_i z_i b + r_{2,i}' \).

Then \( K_B \) and \( K_A \) are computed as follows:

\[
K_B = g^{-(r_1 + r_2)} \prod_{i=1}^n g^{-(r_{1,i} + r_{2,i})} = \prod_{i=1}^n g^{-(r_{1,i} + r_{2,i})}
\]
\[
K_A = g_2 \prod_{i=1}^n K_{1,i}^{-r_{1,i}} K_{2,i}^{-r_{2,i}} K_{3,i}^{-r_{2,i}} K_{4,i}^{-r_{2,i}}
\]

For \( K_A \), the components are computed as follows:

\[
K_{1,i}^{-r_{1,i}} K_{2,i}^{-r_{2,i}} = g^{-\gamma_2 c^i v_i y_i b} \cdot g^{\gamma_1 c^i v_i y_i b} \cdot (K_{1,i}')^{-r_{1,i}} \cdot (K_{2,i}')^{-r_{2,i}}
\]
\[
= g^{-\theta_2 c^i (x_i y_i z_i b + r_{2,i})} \cdot (K_{1,i}')^{-r_{1,i}} \cdot (K_{2,i}')^{-r_{2,i}}
\]
\[
= g^{(f_2 y_i z_i w_{2,i})(-z_{1,i} - \theta_1 b z_{1,i})} g^{(f_2 y_i z_i w_{1,i})(-z_{2,i} - \theta_2 b z_{2,i})}
\]
\[
= g^{(f_2 y_i z_i w_{2,i})(-z_{2,i} - \theta_2 b z_{2,i})} K_{4,i}^{-r_{2,i}'}
\]
\[
= g^{(f_2 y_i z_i w_{1,i})(-z_{2,i} - \theta_2 b z_{2,i})}
\]

Since \( g_2 = g^\lambda \) then \( K_A \) is computed as:

\[
K_A = g^\lambda \prod_{i=1}^n g^{-\gamma_2 c^i v_i y_i b} \cdot g^{\gamma_1 c^i v_i y_i b} \cdot (K_{1,i}')^{-r_{1,i}} \cdot (K_{2,i}')^{-r_{2,i}}
\]
\[
\cdot g^{(c^i x_i y_i z_i b + r_{2,i}') \Delta} g^{(f_2 y_i z_i w_{2,i})(-z_{1,i} - \theta_1 b z_{1,i})} \cdot g^{(f_2 y_i z_i w_{1,i})(-z_{2,i} - \theta_2 b z_{2,i})}
\]

\( B \) gives \( A \) the private key \( SK = (K_A, K_B, \{K_{1,i}', K_{2,i}', K_{3,i}', K_{4,i}'\}_{i=1}^n) \) for the queried vector \( \vec{y} \).
• **Challenge Ciphertext**: To generate a challenge ciphertext, \( \mathcal{B} \) picks random 
\( s_1', \alpha' \in \mathbb{Z}_p \). \( \mathcal{B} \) implicitly sets:

\[
s_1 = c, s_2 = d, \alpha = \alpha'
\]

Then \( \mathcal{B} \) sets: \( A = g^d = g^{z_2}, \mathcal{B} = (g^{nc})^\Delta = g^{z_1}_1 \). For \( i \) from 1 to \( n \), \( \mathcal{B} \) computes:

\[
\begin{align*}
C_{1,i} &= (g^{aw_1,i})^c(g^d)^{z_1,i}Z_1^{\theta_1,i}, \\
C_{2,i} &= (g^{aw_2,i})^c(g^d)^{z_2,i}Z_2^{\theta_2,i}.
\end{align*}
\]

Next \( \mathcal{B} \) computes for \( i \) from 1 to \( n \):

\[
\begin{align*}
C_{3,i} &= (g^{aw_1,i})^c(g^d)^{z_1,i}Z_1^{\theta_1,i}, \\
C_{4,i} &= (g^{aw_2,i})^c(g^d)^{z_2,i}Z_2^{\theta_2,i}.
\end{align*}
\]

If \( Z = g^{b(c+d)} \) then, \( \mathcal{B} \) is playing \( \text{Game}_2 \) with \( \mathcal{A} \)

\[
\begin{align*}
C_{3,i} &= (g^{aw_1,i})^c(g^d)^{z_1,i}(g^{b(c+d)}g^r)^{\theta_1,i} = W_{1,i}^{s_1}Z_{1,i}^sX_1^{z_1\beta}, \\
C_{4,i} &= (g^{aw_2,i})^c(g^d)^{z_2,i}(g^{b(c+d)}g^r)^{\theta_2,i} = W_{2,i}^{s_1}Z_{2,i}^sX_2^{z_2\beta}.
\end{align*}
\]

Otherwise, if \( Z = g^{b(c+d)}g^r \) for \( r \) chosen randomly in \( \mathbb{Z}_p \), then \( \mathcal{B} \) is playing \( \text{Game}_3 \) with \( \mathcal{A} \) by setting \( \beta = r \)

\[
\begin{align*}
C_{3,i} &= (g^{aw_1,i})^c(g^d)^{z_1,i}(g^{b(c+d)})^{\theta_1,i} = W_{1,i}^{s_1}Z_{1,i}^{s_2}X_1^{z_1i}, \\
C_{4,i} &= (g^{aw_2,i})^c(g^d)^{z_2,i}(g^{b(c+d)})^{\theta_2,i} = W_{2,i}^{s_1}Z_{2,i}^{s_2}X_2^{z_2i}.
\end{align*}
\]

Therefore, if \( \mathcal{A} \) can distinguish \( \text{Game}_2 \) from \( \text{Game}_3 \), then \( \mathcal{B} \) can solve the DLIN problem.

The rest of the proof is similar to the above proofs:

• the indistinguishability between \( \text{Game}_3 \) and \( \text{Game}_4 \) can be proved in the same way as for \( \text{Game}_2 \) and \( \text{Game}_3 \);

• the indistinguishability between \( \text{Game}_4 \) and \( \text{Game}_5 \) can be proved in the same way as for \( \text{Game}_1 \) and \( \text{Game}_2 \);

• the indistinguishability of \( \text{Game}_5 \) and \( \text{Game}_6 \) can be proved in the same way as for \( \text{Game}_0 \) and \( \text{Game}_1 \). \( \square \)

### 3.4 Comparisons

In Table 3.1, we give a detailed comparison among the existing CP-ABE schemes based on the AND-Gate access structure where \( p \) denotes the pairing operation, \( e \)
denotes the exponentiation operation, and \( t \) is the number of attributes involved in the access structure. In Table 3.1, \( n_1 << t \) is the maximum number of wildcard involved in the access structure.

**Table 3.1:** Comparison among CP-ABE with Hidden AND-Gate access structure

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Ciphertext Length</th>
<th>Dec Cost</th>
<th>Wildcard</th>
<th>Assumption</th>
<th>Hidden Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN[CN07]</td>
<td>([G_T] + (t + 1)[G])</td>
<td>((t + 1)p)</td>
<td>(\sqrt{DBDH})</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td>NYO[NYO08]</td>
<td>([G_T] + (2t + 1)[G])</td>
<td>((2t + 1)p)</td>
<td>(\sqrt{DBDH + DLIN})</td>
<td>(\sqrt{\ })</td>
<td></td>
</tr>
<tr>
<td>Emura et al.,[EMN’09]</td>
<td>([G_T] + 2[G])</td>
<td>(2p)</td>
<td>X</td>
<td>(DBDH)</td>
<td>(X)</td>
</tr>
<tr>
<td>ZH[ZH10]</td>
<td>([G_T] + 2[G])</td>
<td>(2p + 1)</td>
<td>(\sqrt{n-BDHE})</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td>CZF[CZF11]</td>
<td>([G_T] + 2[G])</td>
<td>(2p)</td>
<td>X</td>
<td>(n-BDHE)</td>
<td>(X)</td>
</tr>
<tr>
<td>Our first scheme</td>
<td>([G_T] + 4[G])</td>
<td>(6p)</td>
<td>(\sqrt{DLIN})</td>
<td>(X)</td>
<td></td>
</tr>
<tr>
<td><strong>Our second Scheme</strong></td>
<td>([G_T] + (4n_1 + 2)[G])</td>
<td>((4n_1 + 2)p)</td>
<td>(\sqrt{DBDH + DLIN})</td>
<td>(\sqrt{\ })</td>
<td></td>
</tr>
</tbody>
</table>

### 3.5 Summary

In this chapter, we presented two new constructions of Ciphertext Policy Attribute Based Encryption for the AND-Gate with wildcard access policy. Our first scheme achieves constant ciphertext size, but cannot hide the access policy. On the other hand, our second scheme can even hide the access policy against the legitimate decryptors. We proved that our second construction is secure under the Decisional Bilinear Diffie-Hellman and the Decision Linear assumptions.
Broadcast encryption (BE), introduced by Berkovits [Ber91] and Fiat and Naor [FN93], is a very useful tool for securing a broadcast channel. In a traditional BE scheme, a broadcaster can specify a subset of privileged users (out of the user universe) as the legitimate receivers of a message. Due to the practicality of broadcast encryption in real-world applications, many BE schemes have been proposed in various settings since its introduction (e.g., [NNL01, DF03, BGW05, BW06, DPP07, GW09, PPSS12]).

Attribute Based Encryption (ABE), first introduced by Sahai and Waters [SW05a], allows an encrypter to embed a fine-grained access policy into the ciphertext when encrypting a message. There are two types of ABE. In a Ciphertext Policy (CP) ABE system, each user secret key is associated with a set of user attributes, and every ciphertext is associated with an access policy. A ciphertext can be decrypted by a secret key if and only if the attributes associated with the secret key satisfy the access policy in the ciphertext. Key Policy (KP) ABE is the dual form of CP-ABE, where attributes are used in the encryption process, and access policies are used in the user secret key generation. ABE systems can provide fine-grained access control of encrypted data, and has been extensively studied in recent years (e.g., [GPSW06, BSW07, CN07, Wat11a, ALDP11, LW12]).

Since ABE gives a one-to-many relationship between a ciphertext and the corresponding valid decryption keys, it can be considered as a natural broadcast encryption where the legitimate decryptors are defined by the access policies (CP-ABE) or the attributes (KP-ABE) associated with the ciphertext. As pointed out in [GPSW06, JK10], ABE is useful in some broadcasting systems, such as Pay TV, which require dynamic and flexible access control. For example, the broadcasting company can specify an access policy ((Location: City A) AND (Age: > 18)) when
generating an encrypted data stream for a TV program, and the access policy may be changed to \((\text{Location: City A}) \land (\text{Age: } *)\) (here `*` denotes the wildcard symbol, meaning “don’t care”) for the next program. However, one drawback of using ABE for broadcasting is that the cost of revoking a user (e.g., those fail to pay the subscription fee for Pay TV) is very high, since the secret keys of all the other non-revoked users must be updated.

Attribute Based Broadcast Encryption (ABBE) is a combination of ABE and BE. Specifically, in a CP-ABBE scheme, a user secret key \(SK\) is associated with a user identity (or index) \(ID\) and a set of user attributes \(L\), and a ciphertext \(CT\) generated by the broadcaster is associated with a user list \(S\) and an access policy \(W\). The ciphertext \(CT\) can be decrypted using \(SK\) if and only if \(L\) satisfies \(W\) (denoted by \(L \models W\)) and \(ID \in S\). KP-ABBE is the dual form of CP-ABBE where the positions of the attributes and the access policy are swapped. We can see that similar to normal ABE, ABBE also allows fine-grained and flexible access control. On the other hand, ABBE can provide direct revocation, which is difficult or expensive to achieve in normal ABE systems. Direct revocation means the broadcaster can directly exclude some revoked users without affecting any non-revoked users, and ABBE can easily achieve this by removing the revoked users from the receiver set \(S\). As highlighted in [AI09, JK10], direct revocation is important for real-time broadcasting applications such as Pay TV.

**Related Work**

Several ABBE schemes [LS08, AI09, JK10] have been proposed in the literature. In [LS08], Lubicz and Sirvent proposed a CP-ABBE scheme which allows access policies to be expressed in disjunctive normal form, with the OR function provided by ciphertext concatenation. Attrapadung and Imai [AI09] proposed two KP-ABBE and two CP-ABBE schemes, which are constructed by algebraically combining some existing BE schemes (namely, the Boneh-Gentry-Waters BE scheme [BGW05] and the Sahai-Waters BE scheme [SW08]) with some existing ABE schemes (namely, the KP-ABE scheme by Goyal et al. [GPSW06] and the CP-ABE scheme by Waters [Wat11a]). Junod and Karlov [JK10] also proposed a CP-ABBE scheme that supports boolean access policies with AND, OR and NOT gates. Junod and Karlov’s scheme achieved direct revocation by simply treating each user’s identity as a unique attribute in the attribute universe.

**Contributions**

In order to use ABBE in real-time applications such as Pay TV, the bandwidth requirement and the decryption cost are the most important factors to be considered. Unfortunately, the ciphertext size of the existing ABBE schemes reviewed above is
quite high. The motivation of this work is to construct efficient ABBE schemes in
terms of ciphertext and key size, as well as decryption cost.

The contributions of this work are two efficient ABBE schemes allowing ac-
cess policies to be expressed using AND-gate with positive (+), negative (−), and
wildcard (∗) symbols. To give a high-level picture of our constructions, we use
the positions of different symbols (i.e., positive, negative, and wildcard) to do the
matching between the access structure (containing wildcards) and the attribute list
(containing no wildcard) in the ABE underlying ABBE schemes. We put the indices
of all the positive, negative and wildcard attributes defined in an access structure
into three sets. By using the Viète’s formulas [SLN+10], based on the wildcard set,
the decryptor can remove all the wildcard positions, and obtain the correct message
if and only if the remaining positive and negative attributes have a perfect posi-
tion match. We then incorporate the technique of Boneh-Gentry-Waters broadcast
encryption scheme [BGW05] into our ABE scheme to enable direct revocation.

Our first ABBE scheme is key policy based, and achieves constant key size
and short ciphertext size. The second scheme is ciphertext policy based, achieving
constant ciphertext size and short key size. Both schemes require only constant
number of pairing operations in decryption.

4.1 Definition

4.1.1 KP-ABBE Definition

Let \( U \) denote the set of all user indices, and \( N \) the set of all user attributes. A
key-policy attribute based broadcast encryption scheme consists of four algorithms:

- **Setup(1^\lambda)**: The setup algorithm takes the security parameter \( 1^\lambda \) as input and
  outputs the public parameters \( PK \) and a master key \( MSK \).

- **Encrypt(S, L, M, PK)**: The encryption algorithm takes as input the public
  parameters \( PK \), a message \( M \), a set of user index \( S \subseteq U \) and a set of attributes
  \( L \subseteq N \), and outputs a ciphertext \( CT \).

- **Key Generation(ID, W, MSK, PK)**: The key generation algorithm takes as input the master key \( MSK \), public parameters \( PK \), a user index \( ID \in U \),
  and an access structure \( W \), and outputs a private key \( SK \).

- **Decrypt(PK, CT, SK)**: The decryption algorithm takes as input the public
  parameters \( PK \), a ciphertext \( CT \), and a private key \( SK \), and outputs a
  message \( M \) or a special symbol ‘⊥’.
4.1.2 Security Definition for KP-ABBE

We define the Selective IND-CPA security for KP-ABBE via the following game.

- **Init**: The adversary commits to the challenge user indices $S^*$ and target attribute set $L^*$.
- **Setup**: The challenger runs the Setup algorithm and gives $PK$ to the adversary.
- **Phase 1**: The adversary queries for private keys with pairs of user index and access structure $(ID, W)$ such that $L^* \not\models W$ or $ID \notin S^*$.
- **Challenge**: The adversary submits messages $M_0, M_1$ to the challenger. The challenger flips a random coin $\beta$ and passes the ciphertext $ct^* = Encrypt(PK, M_\beta, L^*, S^*)$ to the adversary.
- **Phase 2**: Phase 1 is repeated.
- **Guess**: The adversary outputs a guess $\beta'$ of $\beta$.

**Definition 12** We say a KP-ABBE scheme is selective IND-CPA secure if for any probabilistic polynomial time adversary

\[
Adv_{kp}^{s-ind-cpa}(\lambda) = |Pr[\beta' = \beta] - 1/2|
\]

is a negligible function of $\lambda$.

4.1.3 CP-ABBE Definition

A ciphertext-policy attribute based broadcast encryption scheme consists of four algorithms:

- **Setup($1^\lambda$)** : The setup algorithm takes the security parameter $1^\lambda$ as input and outputs the public parameters $PK$ and a master key $MSK$.
- **Encrypt($S, W, M, PK$)** : The encryption algorithm takes as input the public parameters $PK$, a message $M$, an access structure $W$, a set of user index $S \subseteq U$, and outputs a ciphertext $CT$.
- **Key Generation($ID, L, MSK, PK$)**: The key generation algorithm takes as input the master key $MSK$, public parameters $PK$, a user index $ID \in U$, and a set of attributes $L \subseteq N$, and outputs a private key $SK$.
- **Decrypt($PK, CT, SK$)** : The decryption algorithm takes as input the public parameters $PK$, a ciphertext $CT$, and a private key $SK$, and outputs a message $M$ or a special symbol ‘⊥’.
4.1.4 Security Definition for CP-ABBE

We define the Selective IND-CPA security for CP-ABBE via the following game.

- **Init**: The adversary commits to the challenge user indices \( S^* \) and target access structure \( W^* \).

- **Setup**: The challenger runs the Setup algorithm and gives \( PK \) to the adversary.

- **Phase 1**: The adversary queries for private keys with pairs of user index and a user attribute list \((ID, L)\) such that \( L \neq W^* \) or \( ID \notin S^* \).

- **Challenge**: The adversary submits messages \( M_0, M_1 \) to the challenger. The challenger flips a random coin \( \beta \) and passes the ciphertext \( ct^* = Encrypt(PK, M_\beta, W^*, S^*) \) to the adversary.

- **Phase 2**: Phase 1 is repeated.

- **Guess**: The adversary outputs a guess \( \beta' \) of \( \beta \).

**Definition 13** We say a CP-ABBE scheme is selective IND-CPA secure if for any probabilistic polynomial time adversary \( Adv_{cp}^{s\text{-}ind\text{-}cpa}(\lambda) = |Pr[\beta' = \beta] - 1/2| \) is a negligible function of \( \lambda \).

4.2 Constructions

4.2.1 KP-ABBE Scheme

In our KP-ABBE scheme, we assume that \(|U| \leq n\) and \(|N| \leq n\) where \( n \) is a system parameter. Let \( N_1, N_2, N_3 \) be three upper bounds for the user attributes:

- \( N_1 \): the maximum number of wildcard in an access structure.
- \( N_2 \): the maximum number of positive attribute in an attribute list \( L \).
- \( N_3 \): the maximum number of negative attribute in an attribute list \( L \).

**Setup**\((1^\lambda)\): The setup algorithm first generates bilinear groups \( G, G_T \) with order \( p \), and selects random generators \( g, h_1, \ldots, h_N \in_R G \) and \( \alpha \in_R \mathbb{Z}_p \). Then compute \( g_i = g^{\alpha^i} \in G \) for \( i = 1, 2, \ldots, n, n + 2, \ldots, 2n \), randomly choose \( \gamma, \delta, \theta, x_1, \ldots, x_{N_1} \in_R \mathbb{Z}_p \), and set:

\[
\nu = g^{\gamma}, V_0 = g^\delta, V_1 = g^\theta, V_{01} = (g^\delta)^{x_1}, \ldots, V_{0N_1} = (g^\delta)^{x_{N_1}}.
\]
Chapter 4. Attribute Based Broadcast Encryption with Short Ciphertext and Decryption Key

53

Randomly choose $s$. The ciphertext is $r$ and contains:

Key Generation:

$PK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h_1, \ldots, h_N, \nu, V_0, V_1, V_{01}, \ldots, V_{0N_1}, V_{11}, \ldots, V_{1N_1})$

$MSK = (\alpha, \gamma, \delta, \theta, x_1, \ldots, x_{N_1})$.

Encrypt($S, L, M, PK$): Given a user index set $S \subseteq U$, an attribute list $L$ which contains:

- $n_2 \leq N_2$ positive attributes at positions $V = \{v_1, \ldots, v_{n_2}\}$;
- $n_3 \leq N_3$ negative attributes at positions $Z = \{z_1, \ldots, z_{n_3}\}$;

the algorithm randomly chooses $r \in \mathbb{Z}_p$ and computes:

$$C_0 = M \cdot e(g_n, g_1)^r, C_1 = g^r, C_2 = (\nu \prod_{j \in S} g_{n+1-j})^r,$$

$$\begin{pmatrix}
C_{3,0} = (V_0 \prod_{i \in V} h_i)^r \\
C_{3,1} = (V_{01} \prod_{i \in V} h_i^i)^r \\
\vdots \\
C_{3,N_1} = (V_{0N_1} \prod_{i \in V} h_i^{N_1})^r
\end{pmatrix}, \begin{pmatrix}
C_{4,0} = (V_1 \prod_{i \in Z} h_i)^r \\
C_{4,1} = (V_{11} \prod_{i \in Z} h_i^i)^r \\
\vdots \\
C_{4,N_1} = (V_{1N_1} \prod_{i \in Z} h_i^{N_1})^r
\end{pmatrix}.$$  

The ciphertext is $CT = (C_0, C_1, C_2, C_{3,0}, \ldots, C_{3,N_1}, C_{4,0}, \ldots, C_{4,N_1})$.

Key Generation($ID, W, MSK, PK$): Suppose that the access structure $W$ contains:

- $n_1 \leq N_1$ wildcards at positions $J = \{w_1, \ldots, w_{n_1}\}$;
- $n_2 \leq N_2$ positive attributes at positions $V' = \{v'_1, \ldots, v'_{n_2}\}$;
- $n_3 \leq N_3$ negative attributes at positions $Z' = \{z'_1, \ldots, z'_{n_3}\}$.

Randomly choose $s_1, s_2 \in \mathbb{Z}_p$, and apply the Viete formulas on $J$ to compute $a_k(0 \leq k \leq n_1)$ and set $t = \sum_{k=0}^{n_1} x_k a_k$ where $x_0 = 1$. Then compute

$$D_1 = g^{o_D^j + \delta s_1 + \theta s_2}, D_2 = g^\gamma, D_3 = g^\varphi,$$

$$D_4 = \left(\prod_{i \in V'} h_i^{j_{i^{k=0}}(i-w_j)}\right)^{s_1}, D_5 = \left(\prod_{i \in Z'} h_i^{j_{i^{k=0}}(i-w_j)}\right)^{s_2}.$$  

and set the secret key $SK = (D_1, D_2, D_3, D_4, D_5)$.

Decrypt($PK, CT, SK$): The decryption algorithm first applies the Viete formulas on $J$ included in the secret key to compute $a_k$ for $0 \leq k \leq n_1$, and

$$V_{11} = (g^\theta)^{z_1}, \ldots, V_{1N_1} = (g^\theta)^{z_{N_1}},$$
\[ e(D_1, C_1) = e(g^{α_{1D}γ + δs_1 + θs_2}, g^r) = e(g^{α_{1D}γ}, g^r)e(g, g)^{δs_1r}e(g^θ, g)^{θs_2r} \]

\[ e(D_4, C_1) = e((\prod_{i \in V} h_i^{i - w_j})^{s_1/τ}, g^r) \]

\[ e(D_5, C_1) = e((\prod_{i \in Z} h_i^{i - w_j})^{s_2/τ}, g^r) \]

\[ e(g(ID), C_2) = e(g^{α_{ID}}, (ν \prod_{j \in S} g_{n+1-j})^y) = e(g^{α_{ID}}, ν)^r e(g^{α_{ID}}, \prod_{j \in S} g_{n+1-j})^r \]

\[ e(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, C_1) = e(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, g^r) \]

\[ \Rightarrow e(g(ID), C_2)/e(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, C_1) = e(g^{α_{ID}}, ν)^r \cdot e(g_n, g_1)^r \]

\[ e(D_2, \prod_{k=0}^{n_1} C_{3,k}^{a_k}) = e(g^{s_1/τ}, V_0^{r \sum_{a_k} k \prod_{i \in V} h_i^{i - w_j}}) = e(g, V_0)^{s_1r} e(\prod_{i \in V} h_i^{i - w_j}, g^{s_1/τ}) \]

\[ e(D_3, \prod_{k=0}^{n_1} C_{4,k}^{a_k}) = e(g^{s_2/τ}, V_1^{r \sum_{a_k} k \prod_{i \in Z} h_i^{i - w_j}}) = e(g, V_1)^{s_2r} e(\prod_{i \in Z} h_i^{i - w_j}, g^{s_2/τ}) \]

If \( L \models W \) and \( ID \in S \), then we have:

\[ M = \frac{C_0 e(g^{α_{ID}γ}, g^r) e(g, g)^{δs_1 + θs_2} e((\prod_{i \in V} h_i^{i - w_j})^{s_1/τ}, g^r) e((\prod_{i \in Z} h_i^{i - w_j})^{s_2/τ}, g^r)}{e(g^{α_{ID}}, ν)^r e(g_n, g_1)^r e(g, V_0)^{s_1} e(\prod_{i \in V} h_i^{i - w_j}, g^{s_1/τ}) e(\prod_{i \in Z} h_i^{i - w_j}, g^{s_2/τ})} \]

### 4.2.2 CP-ABBE Scheme

Our CP-ABBE scheme is the dual-form of our KP-ABBE scheme.

**Setup**\((1^k)\): The setup algorithm first generates bilinear groups \( G, G_T \) with order \( p \), and selects random generators \( g, h_1, \ldots, h_N \in_R G \), and \( α \in_R \mathbb{Z}_p \). Then compute \( g_i = g^{α_i} \in G \) for \( i = 1, 2, \ldots, n, n + 2, \ldots, 2n \), randomly choose \( γ, δ, θ \in_R \mathbb{Z}_p \), and...
set:
\[ \nu = g^\gamma, V_0 = g^\delta, V_1 = g^\theta. \]

The public key and master secret key are defined as:
\[
PK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h_1, \ldots, h_N, \nu, V_0, V_1)
\]
\[
MSK = (\alpha, \gamma, \delta, \theta).
\]

Encrypt \((S, W, M, PK)\): Given a user index set \(S \subseteq U\), and an access structure \(W\) containing:
- \(n_1 \leq N_1\) wildcards at positions \(J = \{w_1, \ldots, w_{n_1}\}\);
- \(n_2 \leq N_2\) positive attributes at positions \(V = \{v_1, \ldots, v_{n_2}\}\);
- \(n_3 \leq N_3\) negative attributes at positions \(Z = \{z_1, \ldots, z_{n_3}\}\);
the algorithm randomly chooses \(r \in \mathbb{Z}_p\) and computes:
\[
C_0 = M \cdot e(g_n, g_1)^r, C_1 = g^r, C_2 = (\nu \prod_{j \in S} g_{n+1-j})^r,
\]
\[
C_3 = (V_0 \prod_{i \in V} h_i^{w_i})^{n_1/(i-w_j)}^r, C_4 = (V_1 \prod_{i \in Z} h_i^{z_i})^{n_1/(i-w_j)}^r.
\]
The ciphertext is \(CT = (J, C_0, C_1, C_2, C_3, C_4)\).

Key Generation \((ID, L, MSK, PK)\): Given a user identity \(ID\) and an attribute list \(L\) which contains:
- \(n_2 \leq N_2\) positive attributes at positions \(V' = \{v'_1, \ldots, v'_{n_2}\}\);
- \(n_3 \leq N_3\) negative attributes at positions \(Z' = \{z'_1, \ldots, z'_{n_3}\}\);
randomly choose \(s_1, s_2 \in \mathbb{Z}_p\) and compute:
\[
D_1 = g^{\alpha ID^\gamma + \delta s_1 + \theta s_2}, D_2 = g^{s_1}, D_3 = g^{s_2}
\]
\[
\begin{pmatrix}
D_{4,0} &=& (\prod_{i \in V'} h_i^{s_1})^s_1 \\
D_{4,1} &=& (\prod_{i \in V'} h_i^{s_1})^s_1 \\
\cdots \\
D_{4,N_1} &=& (\prod_{i \in V'} h_i^{N_1})^s_1
\end{pmatrix},
\begin{pmatrix}
D_{5,0} &=& (\prod_{i \in Z'} h_i^{s_2})^s_2 \\
D_{5,1} &=& (\prod_{i \in Z'} h_i^{s_2})^s_2 \\
\cdots \\
D_{5,N_1} &=& (\prod_{i \in Z'} h_i^{N_1})^s_2
\end{pmatrix},
\]
and set the secret key \(SK = (D_1, D_2, D_3, D_{4,0}, \ldots, D_{4,N_1}, D_{5,0}, \ldots, D_{5,N_1})\).

Decrypt \((PK, CT, SK)\): The decryption algorithm first applies the Viete formulas
on $J$ included in the ciphertext to compute $a_k$ for $0 \leq k \leq n_1$:

\[
e(D_1, C_1) = e(g^{a_{ID}^{T} + \delta s_1 + \theta s_2}, g^r)
= e(g^{a_{ID}^{T} + \delta s_1}, g^r) e(g, g)^{\delta s_1 r} e(g, g)^{\theta s_2 r}
\]

\[
e\left(\prod_{k=0}^{n_1} D_{4,k}^{a_k}, C_1\right) = e\left(\prod_{i \in V} h_i^{\sum_{j=0}^{n_1} i^k a_k}, g^r\right)
= e\left(\prod_{i \in V} h_i^{(i-w_j)s_1}, g^r\right)
\]

\[
e\left(\prod_{k=0}^{n_1} D_{5,k}^{a_k}, C_1\right) = e\left(\prod_{i \in Z'} h_i^{\sum_{j=0}^{n_1} i^k a_k}, g^r\right)
= e\left(\prod_{i \in Z'} h_i^{(i-w_j)s_2}, g^r\right)
\]

\[
e(g_{ID}, C_2) = e(g^{a_{ID}}, (\nu \prod_{j \in S} g_{n+1-j})^r)
= e(g^{a_{ID}}, \nu)^r e(g^{a_{ID}}, \prod_{j \in S} g_{n+1-j})^r
\]

\[
e\left(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, C_1\right) = e\left(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, g^r\right)
\]

\[
\Rightarrow e(g_{ID}, C_2)/e\left(\prod_{j \in S, j \neq ID} g_{n+1-j+ID}, C_1\right) = e(g^{a_{ID}}, \nu)^r e(g_n, g_1)^r
\]

\[
e(D_2, C_3) = e(g^{s_1}, (V_0 \prod_{i \in V} h_i^{(i-w_j)})^r)
= e(g^{s_1}, V_0^r) e(g^{s_1}, \prod_{i \in V} h_i^{(i-w_j)})^r
\]

\[
e(D_3, C_4) = e(g^{s_2}, (V_1 \prod_{i \in Z} h_i^{(i-w_j)})^r)
= e(g^{s_2}, V_1^r) e(g^{s_2}, \prod_{i \in Z} h_i^{(i-w_j)})^r
\]

If $L \models W$ and $ID \in S$, then we have

\[
M = \frac{C_0 e(g^{a_{ID} + \gamma} g^r) e(g, g)^{\delta s_1 r} e(g, g)^{\theta s_2 r} e(g_1, V_0^r) e(g^{s_1}, \prod_{i \in V} h_i^{(i-w_j)})^r}{e(g^{a_{ID}}, \nu)^r e(g_n, g_1)^r e(g^{s_1}, V_0^r) e(g^{s_1}, \prod_{i \in V} h_i^{(i-w_j)})^r e(g^{s_2}, V_1^r) e(g^{s_2}, \prod_{i \in Z} h_i^{(i-w_j)})^r}.
\]
4.3 Security Proof

We prove that the proposed KP-ABBE and CP-ABBE schemes are selectively secure under the Decision $n$-BDHE assumption.

Theorem 6 Assume that the Decision $n$-BDHE assumption holds, then no polynomial-time adversary against our KP-ABBE scheme can have a non-negligible advantage over random guess in the Selective IND-CPA security game.

Proof: Suppose that there exists an adversary $\mathcal{A}$ which can attack our scheme with non-negligible advantage $\epsilon$, we construct another algorithm $\mathcal{B}$ which uses $\mathcal{A}$ to solve the Decision $n$-BDHE problem. On input $(g, h, \gamma, g_{a,n} = (g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}), T)$, where $g_i = g^{a_i}$ and for some unknown $\alpha \in \mathbb{Z}_p^n$, the goal of $\mathcal{B}$ is to determine whether $T = \epsilon(g_{n+1}, h)$ or a random element of $\mathbb{G}_T$.

Init: $\mathcal{A}$ gives $\mathcal{B}$ the challenge user indices $S^*$ and the target attribute set $L^*$ with $n_2 \leq N_2$ positive attributes which occur at positions $V^* = \{v_1^*, \ldots, v_{n_2}^*\}$, and $n_3 \leq N_3$ negative attributes which occur at positions $Z^* = \{z_1^*, \ldots, z_{n_3}^*\}$ at the beginning of the game.

Setup: $\mathcal{B}$ chooses $d, v_0, v_1, u_1, \ldots, u_n, x_1, \ldots, x_{N_1} \in \mathbb{Z}_p$ and generates:

$$
\nu = g^d( \prod_{j \in S^*} g_{n+1-j}^{-1} ) = g^{d-\sum_{j \in S^*} a^{n+1-j}} = g^7,
$$

$$
V_{0j} = (g^{v_0})^{x_j} \prod_{i \in V^*} g^{a_{n+1-i}^j} = (g^{v_0})^{x_j} g^{\sum_{i \in V^*} a_{n+1-i}^j}, \text{ for } j = 0, \ldots, N_1
$$

$$
V_{1j} = (g^{v_1})^{x_j} \prod_{i \in Z^*} g^{a_{n+1-i}^j} = (g^{v_1})^{x_j} g^{\sum_{i \in Z^*} a_{n+1-i}^j}, \text{ for } j = 0, \ldots, N_1
$$

where $x_0 = 1$, and $h_i = g^{u_i-a^{n+1-i}}$, then $\mathcal{B}$ sets public key as:

$$
PK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h_1, \ldots, h_N, \nu, V_0, V_1, V_{01}, \ldots, V_{0N_1}, V_{11}, \ldots, V_{1N_1}).
$$

Phase 1: $\mathcal{A}$ submits a pair of user index and access structure $(ID, W)$ in a secret key query, which satisfies $L^* \notin W$ or $ID \notin S^*$. Assume $W$ consists of $n_1 \leq N_1$ wildcards which occur at positions $J = \{w_1, \ldots, w_{n_1}\}$, $n_2 \leq N_2$ positive attributes which occur at positions $V = \{v_1, \ldots, v_{n_2}\}$, and $n_3 \leq N_3$ negative attributes which occur at positions $Z = \{z_1, \ldots, z_{n_3}\}$. $\mathcal{B}$ applies the Viete formulas on $J = \{j_1, \ldots, j_{n_1}\}$ to get $a_k$ and set $t = \sum_{k=0}^{n_1} x_k a_k$. Consider the following two cases in Phase 1:

- **Case 1:** $ID \notin S^*$. 

B first selects a random number \( s_1, s_2 \in \mathbb{Z}_p \), then computes:

\[
D_1 = g^d_{ID} \prod_{j \in S^*} (g_{n+1-j+ID})^{-1} \prod_{i \in V} (g_{n+1-i})^{s_1} \prod_{i \in Z^*} (g_{n+1-i})^{s_2} = g^{\alpha^j d (d-S^{*}) \alpha^{n+1-j}} (g_{y_0}^{\sum_{i \in V^*} \alpha^{n+1-i}}) (g_{y_0}^{\sum_{i \in Z^*} \alpha^{n+1-i}})^{s_1} (g_{y_0}^{\sum_{i \in Z^*} \alpha^{n+1-i}})^{s_2} = g^{\alpha^j d (d-S^{*}) + \delta s_1 + \theta s_2}.
\]

\[
D_2 = g^{s_1},
\]

\[
D_3 = g^{s_2},
\]

\[
D_4 = \left( \prod_{i \in V} (g_{u_i - \alpha^{n+1-i}}) \prod_{j \in J} (i-w_j) \right)^{s_1} = \left( \prod_{i \in V} h_i^{i-j} \right)^{s_1},
\]

\[
D_5 = \left( \prod_{i \in Z} (g_{u_i - \alpha^{n+1-i}}) \prod_{j \in J} (i-w_j) \right)^{s_2} = \left( \prod_{i \in Z} h_i^{i-j} \right)^{s_2}.
\]

**Case 2:** \( ID \in S^* \).

In this case, due to the constraint \( L^* \neq W \), \( W \) has at least one position \( i^* \) which has a different attribute value from \( L^* \), which means \( \{V \cup Z^*\} \neq \emptyset \) or \( \{Z \cup V^*\} \neq \emptyset \).

\[\diamond\] If there exists an \( i^* \in \{V \cup Z^*\} \neq \emptyset \): B selects two random numbers \( s'_1, s'_2 \in \mathbb{Z}_p \) and implicitly sets \( s_1, s_2 \) as:

\[
\begin{cases} 
  s_1 = s'_1 \\
  s_2 = s'_2 + \alpha^{i^*}
\end{cases}
\]

by setting \( D_2 = g^{s'_1} = g^{s_1}, D_3 = g^{s'_2 + \alpha^{i^*}} = g^{s_2} \). Then B can compute \( D_1, D_4, D_5 \) as follows:

\[
D_1 = g^{\alpha^j d \gamma + \delta s_1 + \theta s_2}.
\]

\[
= g^{\alpha^j d (d-S^{*}) \alpha^{n+1-j}} g^{\sum_{i \in V^*} \alpha^{n+1-i}} (g_{y_0}^{\sum_{i \in V^*} \alpha^{n+1-i}}) (g_{y_0}^{\sum_{i \in Z^*} \alpha^{n+1-i}})^{s_1} (g_{y_0}^{\sum_{i \in Z^*} \alpha^{n+1-i}})^{s_2 + \alpha^{i^*}} = g^{\alpha^j d (d-S^{*}) + \delta s_1 + \theta s_2}.
\]

\[
D_4 = \left( \prod_{i \in V} (g_{u_i - \alpha^{n+1-i}}) \prod_{j \in J} (i-w_j) \right)^{s'_1} = \left( \prod_{i \in V} h_i^{i-j} \right)^{s'_1}.
\]

\[
D_5 = \left( \prod_{i \in Z} (g_{u_i - \alpha^{n+1-i}}) \prod_{j \in J} (i-w_j) \right)^{s'_2 + \alpha^{i^*}} = \left( \prod_{i \in Z} h_i^{i-j} \right)^{s'_2}.
\]
We should note that since \( i^* \not\in Z \), the item \( g^{\alpha_{i^*}} \) will not occur in the calculation of \( D_5 \).

\( \diamond \) If there exists an \( i^* \in \{Z \cup V^*\} \neq \emptyset \):

the simulation can be performed in a similar way by choosing two random numbers \( s'_1, s'_2 \in \mathbb{Z}_p \) and implicitly setting \( s_1, s_2 \) as:

\[
\begin{cases}
  s_1 = s'_1 + \alpha_{i^*} \\
  s_2 = s'_2
\end{cases}
\]

by setting \( D_2 = g^{s'_1 + \alpha_{i^*}} = g^{s_1}, D_3 = g^{s'_2} = g^{s_2} \). Then \( B \) can compute \( D_1, D_4, D_5 \) as follows:

\[
D_1 = g^{s_1 d} g^{D_3 \gamma + \delta s_1 + \theta s_2} = g^{s_1 d} \prod_{i \in V^*} (g_{n+1-i})^{s_1} \prod_{i \in Z^*} (g_{n+1-i})^{s_2} = g^{s_1} \prod_{j \in S^*} (g_{n+1-j+ID})^{-1} \prod_{j \in S^*} g^{\alpha_{n+1-j}} (g^{s'_1 + \alpha_{i^*}})\left(g_{\sum_{i \in V^*} \alpha_{n+1-i}}\right)^{s_1'} (g_{\sum_{i \in V^*, i \neq i^*} \alpha_{n+1-i}})^{s_2'},
\]

\[
D_4 = \left( \prod_{i \in V} \left( g_{u_i - \alpha_{n+1-i}} \right) \prod_{j \in J} (i-w_j) \right)^{s'_1 + \alpha_{i^*}} / t = \left( \prod_{i \in V} h_t^{i} \right)^{s_1'},
\]

\[
D_5 = \left( \prod_{i \in Z} \left( g_{u_i - \alpha_{n+1-i}} \right) \prod_{j \in J} (i-w_j) \right)^{s_2'} / t = \left( \prod_{i \in Z} h_t^{i} \right)^{s_2'}.
\]

We should note that since \( i^* \not\in V \), the item \( g^{\alpha_{i^*}} \) will not occur in the calculation of \( D_5 \).

\( B \) returns to \( A \) the secret key \( SK = (D_1, D_2, D_3, D_4, D_5) \).

**Challenge:** The adversary gives two messages \( M_0 \) and \( M_1 \) to \( B \). Then \( B \) flips a coin \( b \) and generates the challenge ciphertext by setting \( C_1 = g^\tau = h \) for some unknown \( \tau \) and

\[
C_2 = h^d = (g^d)^\tau = (g^d \prod_{j \in S^*} (g_{n+1-j})^{-1} \prod_{j \in S^*} (g_{n+1-j})^\tau = (\nu \prod_{j \in S^*} (g_{n+1-j}))^\tau,
\]

\[
C_{3,k} = h \sum_{i \in V^*} u_{ik} = (g \sum_{i \in V^*} u_{ik})^\tau,
\]

\[
C_{4,k} = h \sum_{i \in Z^*} u_{ik} = (g \sum_{i \in Z^*} u_{ik})^\tau.
\]
\(\mathcal{B}\) then sends the following challenge ciphertext to \(\mathcal{A}\)

\[ CT^* = (M_b T, C_1, C_2, \{C_{3,k}\}, \{C_{4,k}\}) \]

**Phase II:** Same as Phase I.

**Guess:** \(\mathcal{A}\) output \(b' \in \{0, 1\}\). If \(b' = b\) then \(\mathcal{B}\) outputs 1, otherwise outputs 0.

**Analysis:** If \(T = e(g_{n+1}, h)\), then the simulation is the same as in the real game. Hence, \(\mathcal{A}\) will have the probability \(\frac{1}{2} + \epsilon\) to guess \(b\) correctly. If \(T\) is a random element of \(G_T\), then \(\mathcal{A}\) will have probability \(\frac{1}{2}\) to guess \(b\) correctly. Therefore, \(\mathcal{B}\) can solve the Decision \(n\)-BDHE assumption also with advantage \(\epsilon\). □

**Theorem 7** Assume that the Decision \(n\)-BDHE assumption holds, then no polynomial-time adversary against our CP-ABBE scheme can have a non-negligible advantage over random guess in the Selective IND-CPA security game.

**Proof:** Suppose that there exists an adversary \(\mathcal{A}\) which can attack our scheme with non-negligible advantage \(\epsilon\), we construct another algorithm \(\mathcal{B}\) which uses \(\mathcal{A}\) to solve the Decision \(n\)-BDHE problem. On input \((g, h, \overline{y}_{g, a, n} = (g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}), T)\), where \(g_i = g^{\alpha_i}\) and for some unknown \(\alpha \in \mathbb{Z}_p^*\), the goal of \(\mathcal{B}\) is to determine whether \(T = e(g_{n+1}, h)\) or a random element of \(G_T\).

**Init:** \(\mathcal{A}\) gives \(\mathcal{B}\) the challenge user indexes \(S^*\) and the challenge access structure \(W^*\) with \(n_1 \leq N_1\) wildcards which occur at positions \(J^* = \{w_{1}^*, \ldots, w_{n_1}^*\}\), \(n_2 \leq N_2\) positive attributes which occur at positions \(V^* = \{v_{1}^*, \ldots, v_{n_2}^*\}\), \(n_3 \leq N_3\) negative attributes which occur at positions \(Z^* = \{z_{1}^*, \ldots, z_{n_3}^*\}\) at the beginning of the game.

**Setup:** \(\mathcal{B}\) chooses \(d, v_0, v_1, u_1, \ldots, u_n \in \mathbb{Z}_p\) and generates:

\[
\nu = g^d \left( \prod_{j \in S^*} g_{n+1-j}^{-1} \right) = g^{d - \sum_{j \in S^*} a^{n+1-j}} = g^\gamma,
\]

\[
V_0 = g^{v_0} \prod_{i \in V^*} g^{a^{n+1-i} \sum_{j \in J^*} (i - w_j)} = g^{v_0 + \sum_{i \in V^*} a^{n+1-i} \sum_{j \in J^*} (i - w_j)} = g^\delta,
\]

\[
V_1 = g^{v_1} \prod_{i \in Z^*} g^{a^{n+1-i} \sum_{j \in J^*} (i - w_j)} = g^{v_1 + \sum_{i \in Z^*} a^{n+1-i} \sum_{j \in J^*} (i - w_j)} = g^\theta,
\]

and \(h_i = g^{n_i - a^{n+1-i}}\), then \(\mathcal{B}\) sets public key as:

\[ PK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, h_1, \ldots, h_N, \nu, V_0, V_1). \]

**Phase 1:** \(\mathcal{A}\) submits \((ID, L)\) in a secret key query, where \(L \not\in W^* \text{ "or" } ID \not\in S^*\). Suppose the attribute set \(L\) contains \(n_2 \leq N_2\) positive attributes which occur at positions \(V = \{v_1, \ldots, v_{n_2}\}\), and \(n_3 \leq N_3\) negative attributes which occur at positions \(Z = \{z_1, \ldots, z_{n_3}\}\). We consider two cases in Phase 1:
• Case 1: \( ID \notin S^* \)

\( B \) first selects random numbers \( s_1, s_2 \in \mathbb{Z}_p \) and computes:

\[
D_1 = g_{ID}^{d} \prod_{j \in S^*} (g_{n+1-j+ID})^{-1}g^{s_0s_1} \prod_{i \in V^*} (g_{n+1-i}^{j})^{s_1}g^{s_2} \prod_{i \in Z^*} (g_{n+1-i}^{j})^{s_2}
\]

\[
= g^{\alpha_{ID}(d-\sum_{j \in S^*} \alpha^{n+1-j}) \sum_{i \in V^*} \alpha^{n+1-i} \sum_{j \in J^*} (i-w_j^*)} g^{s_1}(g \prod_{i \in V^*} (g_{n+1-i}^{j})^{s_1}g^{s_2} \prod_{i \in Z^*} (g_{n+1-i}^{j})^{s_2})
\]

\[
= g^{\alpha_{ID}+\delta s_1+\theta s_2},
\]

\( D_2 = g^{s_1}, \quad D_3 = g^{s_2}, \quad D_{4,k} = \prod_{i \in V} (g^{n+s_{1}^{i}+\alpha^{n+1-i}})^{\delta s_1} = \prod_{i \in V} h_i^{\delta s_1}, \quad D_{5,k} = \prod_{i \in Z} (g^{n+s_{1}^{i}+\alpha^{n+1-i}})^{\delta s_2} = \prod_{i \in Z} h_i^{\delta s_2}.
\]

• Case 2: \( ID \in S^* \)

In this case, due to the constraint \( L \neq W^* \), \( L \) has at least one position \( i^* \) which has a different attribute value from \( W^* \), which means \( \{V \cup Z^*\} \neq \emptyset \) or \( \{Z \cup V^*\} \neq \emptyset \).

\( \diamond \) If there exists \( i^* \in \{V \cup Z^*\} \neq \emptyset \):

\( B \) selects two random numbers \( s'_1, s'_2 \in \mathbb{Z}_p \) and implicitly sets \( s_1, s_2 \) as:

\[
\left\{ \begin{array}{l}
  s_1 = s'_1 \\
  s_2 = s'_2 + \sum_{j \in J^*}(i^*-w_{j}^*) \left( g^{n+s_0}\prod_{i \in V^*} (g_{n+1-i}^{j})^{s_1}g^{s_2} \prod_{i \in Z^*} (g_{n+1-i}^{j})^{s_2} \right) \\
\end{array} \right.
\]

\( g^{s_2}. \) Then \( B \) can compute \( D_1, D_{4,k}, D_{5,k} \) as follows:

\[
D_1 = g^{\alpha_{ID}+\delta s_1+\theta s_2}.
\]

\[
= g^{\alpha_{ID}(d-\sum_{j \in S^*} \alpha^{n+1-j}) \sum_{i \in V^*} \alpha^{n+1-i} \sum_{j \in J^*} (i-w_j^*)} g^{s_1}(g \prod_{i \in V^*} (g_{n+1-i}^{j})^{s_1}g^{s_2} \prod_{i \in Z^*} (g_{n+1-i}^{j})^{s_2})
\]

\[
= g^{\alpha_{ID}} \prod_{j \in S^*} (g_{n+1-j+ID})^{-1}g^{-\alpha^{n+1}} \prod_{j \neq ID} (g_{n+1-j+ID})^{-1}g^{-a_{n+1}} \end{array} \right.
\]

\[
= g^{\alpha_{ID}} \prod_{j \in S^*} (g_{n+1-j+ID})^{-1}g^{-\alpha^{n+1}} \prod_{j \neq ID} (g_{n+1-j+ID})^{-1}g^{-a_{n+1}}
\]

\[
\]
If there exists an $i^* \in \{Z \cup V^*\} \neq \emptyset$, the simulation can be performed in a similar way by choosing two random numbers $s'_1, s'_2 \in \mathbb{Z}_p$ and implicitly setting $s_1, s_2$ as:

$$s_1 = s'_1 + \prod_{j \in J^*}(i^*-w_j)$$
$$s_2 = s'_2$$

setting $D_2 = g^{s_1}, D_3 = g^{s_2} = g^{s_2}$. Then $B$ can compute $D_1, D_{4,k}, D_{5,k}$ as follows:

$$D_1 = g^{\alpha^{ID} + \delta s_1 + \theta s_2}$$
$$= g^{\alpha^{ID} \prod_{j \in S^* \setminus ID} (g_{n+1-j+ID})^{-1} \prod_{i \in V^*} (g_{n+1-i})^{s_1} \prod_{i \in Z^*} (g_{n+1-i})^{s_2}}$$
$$= g^{\alpha^{ID} \prod_{j \in S^* \setminus ID} (g_{n+1-j+ID})^{-1} \prod_{i \in V^*} (g_{n+1-i})^{s_1} \prod_{i \in Z^*} (g_{n+1-i})^{s_2}}$$

$$D_{4,k} = \prod_{i \in V} (g_{u_i - \alpha^{n+1-i}})^{i^k s'_1} = \prod_{i \in V} h_i^{i^k s_1}$$

$$D_{5,k} = \prod_{i \in Z} (g_{u_i - \alpha^{n+1-i}})^{i^k s'_2} = \prod_{i \in Z} h_i^{i^k s_2}$$

$$D_2 = g^{s_1}, D_3 = g^{s_2}$$
\[ D_{4,k} = \prod_{i \in V} (g^{u_i - \alpha^{n+1-i}})^{i \tau'} \prod_{i \in Z} (g^{u_i - \alpha^{n+1-i}})^{i \tau''} = \prod_{i \in V} h_i^{i \tau'} \prod_{i \in Z} h_i^{i \tau''} \]

\[ D_{5,k} = \prod_{i \in V} (g^{u_i - \alpha^{n+1-i}})^{i \tau} \prod_{i \in Z} (g^{u_i - \alpha^{n+1-i}})^{i \tau} = \prod_{i \in V} h_i^{i \tau} \prod_{i \in Z} h_i^{i \tau} \]

\( \mathcal{B} \) returns to \( \mathcal{A} \) the secret key \( SK = (D_1, D_2, D_3, \{D_{4,k}\}, \{D_{5,k}\}) \).

**Challenge:** The adversary gives two messages \( M_0 \) and \( M_1 \) to \( \mathcal{B} \). Then \( \mathcal{B} \) flips a coin \( b \) and generates the challenge ciphertext by setting \( C_1 = g^\tau = h \) for some unknown \( \tau \) and

\[ C_2 = h^d = (g^d)^\tau \]

\[ = (g^d \prod_{j \in S^*} (g^{n+1-j})^{-1} \prod_{j \in S^*} (g^{n+1-j}))^{\tau} \]

\[ = (\nu \prod_{j \in S^*} (g^{n+1-j}))^{\tau} \]

\[ C_3 = h^{v_0 + \sum_{i \in V^*} u_i \prod_{j \in J^*} (i-w_j^*)} = (g^{v_0 + \sum_{i \in V^*} u_i \prod_{j \in J^*} (i-w_j^*)})^{\tau} \]

\[ C_4 = h^{v_1 + \sum_{i \in Z^*} u_i \prod_{j \in J^*} (i-w_j^*)} = (g^{v_1 + \sum_{i \in Z^*} u_i \prod_{j \in J^*} (i-w_j^*)})^{\tau} \]

\( \mathcal{B} \) sends the following challenge ciphertext to \( \mathcal{A} \):

\[ CT^* = (M_b T, C_1, C_2, C_3, C_4) \]

**Phase II:** Same as Phase I.

**Guess:** \( \mathcal{A} \) outputs \( b' \in \{0,1\} \). If \( b' = b \) then \( \mathcal{B} \) outputs 1, otherwise outputs 0.

**Analysis:** If \( T = e(g^{n+1}, h) \), then the simulation is the same as in the real game. Hence, \( \mathcal{A} \) will have the probability \( \frac{1}{2} + \epsilon \) to guess \( b \) correctly. If \( T \) is a random element of \( \mathbb{G}_T \), then \( \mathcal{A} \) will have probability \( \frac{1}{2} \) to guess \( b \) correctly. Therefore, \( \mathcal{B} \) can solve the Decision \( n \)-BDHE assumption also with advantage \( \epsilon \).

### 4.4 Comparisons

We give a comparison among in ABBE schemes in Table 4.1 in CP-ABBE and Table 4.2 in KP-ABBE. The schemes are compared in terms of the order of the
underlying group, ciphertext size, decryption cost, access structure, and security assumption. In the table, \( p \) denotes the pairing operation, \( N \) total attributes in system, \( n \) the number of attributes in access structure, \( t \) the number of attributes in user key, \( l \) total number of users, and \( N << n \) denotes the maximum number of wildcards.

We should note that in our CP-ABBE scheme the wildcard positions should be attached with the ciphertext. A naive way to do this is to include an \( n \)-bit string where a bit “1” indicates wildcard at that position. Similar to the previous works on BE [BGW05] and ABBE [AI09], this information together with the target receiver set \( S \) are not counted when measuring the ciphertext size in Table 4.1 and Table 4.2.

<table>
<thead>
<tr>
<th>CP-ABBE</th>
<th>Ciphertext Size</th>
<th>Private Key Size</th>
<th>Decryption Cost</th>
<th>Access Structure</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LS08]</td>
<td>((n + l)G + tG)</td>
<td>((n + l)G)</td>
<td>(3p)</td>
<td>DNF policy</td>
<td>GHDE</td>
</tr>
<tr>
<td>[AI09]</td>
<td>((n + l)G + tG)</td>
<td>((n + l)G)</td>
<td>((2N \cap n) + 3p)</td>
<td>LSSS</td>
<td>n-BDHE, MEBDH</td>
</tr>
<tr>
<td>[JK10]</td>
<td>((n + l)G + tG)</td>
<td>((n + l)G)</td>
<td>(2N + 3p)</td>
<td>DNF, CNF policy</td>
<td>GHDE</td>
</tr>
<tr>
<td>CP-ABBE</td>
<td>(4G + 16G)</td>
<td>((2N + 3)G)</td>
<td>(7p)</td>
<td>AND Gates + wildcard</td>
<td>n-BDHE</td>
</tr>
</tbody>
</table>

(GHDE = General Diffie Hellman Exponent (GDHE) problem)

<table>
<thead>
<tr>
<th>KP-ABBE</th>
<th>Ciphertext Size</th>
<th>Private Key Size</th>
<th>Decryption Cost</th>
<th>Access Structure</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AI09]</td>
<td>((n + 2)G + 16G)</td>
<td>(5G)</td>
<td>((2N \cap n) + 2p)</td>
<td>LSSS</td>
<td>n-BDHE, MEBDH</td>
</tr>
</tbody>
</table>

Remark: In Table 4.1 and 4.2 we do not count the wildcard positions when measuring the ciphertext size. To indicate those wildcard positions, a naive way is to use an \( L \)-bit string, which has the same size as several group elements when \( L \) is linear in the security parameter.

### 4.5 Summary

We proposed two efficient Attribute Based Broadcast Encryption (ABBE) schemes allowing access policies to be expressed using AND-gate with positive, negative, and wildcard symbols. Our first key policy ABBE scheme achieves constant secret key size, while the second ciphertext policy ABBE scheme achieves constant ciphertext size, and both schemes require only constant number of pairing operations in decryption. We also proved the security of our schemes under the Decision n-BDHE assumption.
Chapter 5

Anonymous Attribute Based Broadcast Encryption Supporting Multi-Gate Access Structure

The protection of sensitive resources is essential in public channel, where everybody can access them without any restriction. By enforcing the access control into the channel, only the authorized individuals can have permission to access those resources. It is important to integrate into broadcasting systems a fine-grained access control mechanism, which based on attributes rather than unique identities. Actually, some distributed systems such as subscription-based channel require both identity and attributes (age, career, address, etc...), in order to an entity to access to system. Among all the existing cryptographic tools, Attribute Based Encryption (ABE) [SW05a, GPSW06, BSW07, CN07, Wat11b, ALP11, LW12] can be used as an efficient mechanism to specify the access control over encrypted data, and it is desirable to combine broadcast encryption (BE) [Ber91, FN93] with ABE to produce a broadcast channel enforcing access policies.

In this work, we address an important problem in ABBE scheme, which is hiding the access structure including the conjunction of the revoked user set and the access policy in a ciphertext. Such an ABBE scheme can be deal with the anonymous recipients when broadcasting in the public channel. One explicit application of our scheme is Pay TV. Suppose that the manager of Pay TV wants to offer attractive channels to his customers with ideal prices. The manager does not want to reveal the identities of registered customers and their access policies such as “(Town A AND Age > 22) OR (Town C AND Male) ”. The reason is that if the policy is hidden, the competitive companies can not see and find out the company’s customer details.

Although a secure ABBE can well protect the secrecy of the encrypted data
against unauthorised access, it does not protect the privacy of the receivers/decryptors by default. That is, given the ciphertext, an unauthorised user may still be able to obtain some information of the data recipients. For example, in our above scenario, if a CP-ABBE cannot hide the group user and access policy, then from the fact whether a person can decrypt the message or not, people can directly learn some sensitive information of the user. Therefore, it is also very important to hide the access policy in such applications. However, the existing ABBE schemes cannot achieve this property.

Our contribution.

Similar to ABE, there are two kinds of ABBE: Key Policy-ABBE and Ciphertext Policy-ABBE as ABE scheme. We focus on CP-ABBE in this research. We first give extensive construction to bridge ABBE based on AND-Gate with wildcard with Inner Product Encryption (IPE) [KSW08a, SW08, ACP12] named Anonymous AND Ciphertext Policy-ABBE (A-AND-ABBE). Specifically, we present a way to convert a group revoker user and an access policy containing positive, negative, and wildcard symbols into a vector $\vec{X}$ which is used for encryption. Next, the user ID and user’s attributes containing positive and negative symbols are converted into another vector $\vec{Y}$ which is used in key generation. Apart from ABBE in chapter 4, we use the positions of different symbols and the Viète’s formulas [SLN+10] to perform the conversion. Similar to the A-AND-ABBE, we then construct Anonymous Attribute Based Broadcast Encryption (AABBE) scheme with supporting boolean gates (AND, OR) with positive, negative attributes named Anonymous OR-AND-ABBE (A-OR-AND-ABBE). Furthermore, we show how to construct our scheme with supporting arbitrary OR/AND gates in access structure. Then we apply the technique of IPE based on the two vectors $\vec{X}$ and $\vec{Y}$. Due to the attribute-hiding property of IPE, we achieve the anonymity for our CP-ABBE. An anonymous KP-ABBE scheme can be constructed in a similar way by snapping the positions of the two vectors.

5.1 Constructions

5.1.1 Anonymous AND Attribute Based Broadcast Encryption

We propose a new transformation technique which can deal with hidden revoke user set in broadcast mechanism and the Viète’s formula to achieve the hidden access policy in CP-ABBE.
Idea

Our main idea is to convert the tuple of revoke user set access policy and the tuple of user ID and user attributes into two vectors, and then apply the technique of Inner Product Encryption to hide the whole information in the ciphertext.

Firstly, we assume the revoke user set as $S = (ID_a, ID_b, ID_c, \ldots, ID_s)$ with the total user set equal to $|U|$, then apply the Viète’s formula to construct:

$$
\begin{align*}
ID_a + ID_b + ID_c + \ldots + ID_s &= a_{|S|} \\
(ID_aID_b + ID_aID_c + \ldots + ID_aID_s) \\
\ldots + ID_{|S|-1}ID_s &= a_{|S|-1} \\
\ldots \\
ID_aID_bID_c\ldots ID_s &= a_0
\end{align*}
$$

Similar to the first scheme, we separate the positive, negative, and wildcard symbols in an access structure into three sets: $V$, $Z$, and $J$. Based on the set $J$, by applying the Viète’s formulas, we can construct a polynomial $\sum_{k=0}^{n} a_k i^k$ with coefficients $(b_0, b_1, \ldots, b'_n)$.

Then we combine the set of positive positions $V$ as:

$$
\Pi_V = + \sum_{i \in V} \prod_{w_j \in J} (i - w_j)
$$

and the set of negative positions $Z$ as:

$$
\Pi_Z = - \sum_{i \in Z} \prod_{w_j \in J} (i - w_j)
$$

We then produce a vector:

$$
\vec{v} = (a_0, a_1, \ldots, a_{|S|}, 0_{|U|}, b_0, b_1, \ldots, b_n', 0_{n'+1}, \ldots, 0_{N_1}, \Pi_V, \Pi_Z)
$$

which will be used for encryption.

In user key generation, each user has a unique $ID$, we set as the root variant of polynomial degree $U$, and we also separate the positive and negative attributes into two sets and construct two vectors

$$
\vec{x}_{V'} = (ID^0, ID^1, ID^2, \ldots, ID^{|U|}, v_0', v_1', v_2', \ldots, v_{N_1}', 1, 0),
$$
$$
\vec{x}_{Z'} = (ID^0, ID^1, ID^2, \ldots, ID^{|U|}, z_0', z_1', z_2', \ldots, z_{N_1}', 0, 1),
$$
in which:
\[
v'_k = -\sum_{i \in V'} i^k, \quad k = 0, \ldots, N_1, \\
z'_k = +\sum_{i \in Z'} i^k, \quad k = 0, \ldots, N_1.
\]

Notice that we assume there are at most \(N_1\) wildcard positions in an access policy. The decryption will be based on the inner products of \((\vec{v}, x_{V'})\) and \((\vec{v}, x_{Z'})\), which should both return 0 in order to have a successful decryption.

Below we give a simple example based on Table II to illustrate our idea. Let \(U = 3, L = 4, N_1 = 2\) then \(S = 2, 3, 5; W_2 = (+, -, *, *)\) be revoke user set and the access policy, respectively. Then we create three sets for wildcard positions \(J = \{3, 4\}\), positive positions \(V = \{1\}\), and negative positions \(Z = \{2\}\). Based on Viète’s formulas, we can calculate

\[
a_0 = -30, \quad a_1 = 31, \quad a_2 = -10, \quad a_3 = 1, \quad b_0 = 12, \quad b_1 = -7, \quad b_2 = 1
\]

and obtain the vector \(\vec{v}\) for the access policy and the vectors for Alice with \(ID = 5\) and Bob with \(ID = 8\) as follows.

<table>
<thead>
<tr>
<th>(\vec{v})</th>
<th>(\Pi_v)</th>
<th>(\Pi_z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_0)</td>
<td>(a_1)</td>
<td>(a_2)</td>
</tr>
<tr>
<td>-30</td>
<td>31</td>
<td>-10</td>
</tr>
</tbody>
</table>

**Figure 5.1:** The vector \(\vec{v}\) for access policy \(W_2\)

<table>
<thead>
<tr>
<th>(\vec{z})</th>
<th>(ID_a^0)</th>
<th>(ID_b^1)</th>
<th>(ID_a^2)</th>
<th>(x_0)</th>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_{\eta_a})</th>
<th>(x_{\eta_z})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5(^0)</td>
<td>5(^3)</td>
<td>5(^3)</td>
<td>(-(1^0 + 4^0))</td>
<td>(-(1^1 + 4^1))</td>
<td>(-(1^2 + 4^2))</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Bob</td>
<td>8(^0)</td>
<td>8(^2)</td>
<td>8(^2)</td>
<td>(-(2^0 + 3^0))</td>
<td>(-(2^1 + 3^1))</td>
<td>(-(2^2 + 3^2))</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 5.2:** The vector \(\vec{z}\) for Alice and Bob

If we calculate the inner product of \(\vec{v}\) and the two vectors of Alice, the product will return 0, i.e., ID of Alice include in \(S\), and Alice’s attributes satisfy the access policy \(W_2\). On the other hand, the inner product of \((\vec{v}, \vec{z}_{Bob}) = 90 + 8 = 98\) and \((\vec{v}, \vec{z}_{Bob}) = 90 + 4 = 94\), which means Bob’s attributes cannot satisfy \(W_2\).
Construction

**Setup**(1^3): Assume that we have \( N \) attributes in the universe, \( U \) set of all user indices, and each attribute has two possible values: positive and negative. In addition, we also consider wildcard (meaning “don’t care”) in access structures. Let \( N_1, N_2, N_3 \) be three upper bounds defined as:
- \( N_1 \): the maximum number of wildcard in an access structure;
- \( N_2 \): the maximum number of positive attribute in an attribute set \( S \);
- \( N_3 \): the maximum number of negative attribute in an attribute set \( S \).

The setup algorithm first randomly generates \((g, G, G_T, p, e)\) and sets \( n = \mid U \mid + N_1 + 3\). It then chooses randomly \( \gamma_1, \gamma_2, \theta_1, \theta_2, \{u_{1,i}\}_{i=1}^n, t_1, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n \in \mathbb{Z}_p \) and \( g_2 \in \mathbb{G} \). Then it selects a random \( \Delta \in \mathbb{Z}_p \) and obtains \( \{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, w_2 \) under the condition:

\[
\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i} \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.
\]

For \( i \) from 1 to \( n \), it creates:

\[
U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}},
T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, Z_{1,i} = g^{z_{1,i}},
V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, V_2 = g^{\theta_2}.
\]

Next it sets \( g_1 = g^\Delta, Y = e(g, g_2) \), and the public key \( PK \) and master key \( MSK \) as

\[
PK = (g, \mathbb{G}, \mathbb{G}_T, p, e, g_1, Y, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i}\}, \{W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}_{i=1}^n, \{V_i, X_i\}_{i=1}^2)
\]

\[
MSK = (g_2, \{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}_{i=1}^n, \{v_i, x_i\}_{i=1}^2).
\]

**Encrypt**\((S, W, M, PK)\): Given a user index set \( S = \{ID_a, ID_b, ID_c, \ldots ID_s\} \subseteq U \) and the access structure \( W \) contains: \( n_1 \leq N_1 \) wildcards which occur at positions \( J = \{w_1, \ldots, w_{n_1}\} \); \( n_2 \leq N_2 \) positive attributes which occur at positions \( V = \{v_1, \ldots, v_{n_2}\} \); \( n_3 \leq N_3 \) negative attributes which occur at positions \( Z = \{z_1, \ldots, z_{n_3}\} \). Based on Viète’s formulas, compute for the wildcard positions \( \{w_j\} \)...
(j = 0, 1, 2, · · · , n):

\[
\begin{align*}
ID_a + ID_b + ID_c + \ldots + ID_s &= a_{|S|} \\
(ID_a ID_b + ID_a ID_c + \ldots + ID_a ID_s) \\
\ldots + ID_{|S|-1} ID_s &= a_{|S|-1} \\
\ldots \\
ID_a ID_b ID_c \ldots ID_s &= a_0.
\end{align*}
\]

Then:

\[
\begin{align*}
b_{n_1} &= 1, \\
b_{n_1-1} &= -(w_1 + w_2 + \ldots + w_{n_1}), \\
b_{n_1-2} &= (w_1 w_2 + w_1 w_3 + \ldots + w_{n_1-1} w_{n_1}), \\
\ldots, \\
b_0 &= -(w_1 \cdot w_2 \ldots w_{n_1}).
\end{align*}
\]

Next it computes:

\[
\begin{align*}
\Pi_V &= + \sum_{i \in V} \prod_{w_j \in J} (i - w_j) \\
\Pi_Z &= - \sum_{i \in Z} \prod_{w_j \in J} (i - w_j)
\end{align*}
\]

It creates a vector \( \vec{v} = (v_1, v_2, \ldots, v_n) \) as:

\[
\vec{v} = (a_0, a_1, \ldots, a_{|S|}, \ldots, 0_{|U|}, b_0, b_1, \ldots, b_{n'}, 0_{n'+1}, \ldots, 0_{N_1}, \Pi_V, \Pi_Z).
\]

The encryption algorithm chooses random \( s_1, s_2, \alpha, \beta \in \mathbb{Z}_p \) and creates the ciphertext as follows:

\[
C_m = M \cdot Y^{s_2}, C_A = g^{s_2}, C_B = g_1^{s_1},
\]

\[
\{C_{1,i}, C_{2,i}\} = \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1}^{w_i,\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2}^{w_i,\alpha}\},
\]

\[
\{C_{3,i}, C_{4,i}\} = \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1}^{v_i,\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2}^{v_i,\beta}\}.
\]

Then ciphertext \( CT \) is set as:

\[
CT = (C_m, C_A, C_B, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i=1}^n).
\]

**KeyGen**\( (MSK, ID, L) \): Suppose that a user joins the system with a given user identity \( ID \) and attribute list \( L \), which contains: \( n'_2 \leq N_2 \) positive attributes which occur at positions \( V' = \{v'_1, \ldots, v'_n'\} \); \( n'_3 \leq N_3 \) negative attributes which occur at positions \( Z' = \{z'_1, \ldots, z'_m'\} \). By means of the Viète’s formulas, for all the positive positions \( \{v'_k\} \) \( (k = 0, 1, 2, \ldots, n'_2) \), for all the negative positions \( \{z'_r\} \)
\( (\tau = 0, 1, 2, \ldots, n^2) \), it sets:

\[
v'_k = - \sum_{i \in V'} i^k, \quad k = 0, \ldots, N_1
\]

\[
z'_k = + \sum_{i \in Z'} i^k, \quad k = 0, \ldots, N_1
\]

It creates vectors \( \vec{x}_V \) and \( \vec{x}_Z \) as:

\[
\vec{x}_V = ((ID^0, ID^1, ID^2, \ldots, ID^{|V'|}, v'_0, v'_1, \ldots, v'_N_1, 1, 0).
\]

\[
\vec{x}_Z = ((ID^0, ID^1, ID^2, \ldots, ID^{|V'|}, z'_0, z'_1, + \ldots, z'_N_1, 0, 1).
\]

The key generation algorithm chooses randomly \( r_{i,1}, r_{i,2} \) for \( i = 1 \) to \( n \), and \( f_1, f_2, r_1, r_2 \in \mathbb{Z}_p \), and then creates the secret key as follows:

\[
\{K_{1,i}, K_{2,i}\} = \{g^{-\gamma r_{i,1}} g^{f_1 x_V u_{2,i}}, g^{\gamma |r_{i,1}|} g^{-f_1 x_V u_{1,i}}\},
\]

\[
\{K_{3,i}, K_{4,i}\} = \{g^{-\theta r_{2,i}} g^{f_2 x_Z u_{2,i}}, g^{\theta |r_{2,i}|} g^{-f_2 x_Z u_{1,i}}\},
\]

\[
K_A = g_2 \cdot \prod_{i=1}^{n} K_{1,i}^{-r_{i,1}} K_{2,i}^{-r_{i,2}} K_{3,i}^{-s_{1,i}} K_{4,i}^{-s_{2,i}}.
\]

\[
K_B = \prod_{i=1}^{n} g^{-(r_{1}+r_{2,i})}.
\]

The secret key is set as:

\[
SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^{n}).
\]

**Decrypt(SK, CT):** The decryption algorithm returns:

\[
C_m = \frac{e(C_A, K_A) \cdot e(C_B, K_B) \prod_{j=1}^{n} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i})}{e(C_{1,i}, K_{1,i})}
\]

**Correctness:**

\[
e(C_{1,i}, K_{1,i})
\]

\[
= e(U_{1,i}^{1/2} V_{1,i}^{1/2}, g^{-\gamma_2 r_{1,i}} g^{f_1 x_V u_{2,i}})
\]

\[
= e(g, g)^{r_{1,i} x_{i} (-u_{1,i} \gamma_2)} \cdot e(g, g)^{-r_{1,i} x_{i} \alpha \gamma_1 \gamma_2} \cdot e(g, K_{1,i})^{l_{1,i}}
\]

\[
\cdot e(g, g)^{f_1 x_V u_{1,i} u_{2,i} x_{i}} \cdot e(g, g)^{f_1 x_V u_{1,i} \gamma_1 u_{2,i}}.
\]

\[
e(C_{2,i}, K_{2,i})
\]

\[
= e(U_{2,i}^{1/2} V_{2,i}^{1/2}, g^\gamma r_{1,i} g^{-f_1 x_V u_{1,i}})
\]

\[
= e(g, g)^{r_{1,i} s_{1,i} u_{2,i} \gamma_1} \cdot e(g, g)^{r_{1,i} x_{i} \gamma_1 \gamma_2} \cdot e(g, K_{2,i})^{l_{2,i}}
\]

\[
\cdot e(g, g)^{-f_1 x_V u_{1,i} u_{2,i} x_{i}} \cdot e(g, g)^{-f_1 x_V u_{1,i} \gamma_2 u_{1,i}}.
\]
Therefore, the message $M$

\[ \prod_{j=1}^{2} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i}) \]

\[ = \prod_{i=1}^{n} e(g, g)^{r_{1,i}s_1} \cdot e(g, g)^{f_{1,i}v_i x_i \alpha \Delta} \cdot e(g, K_{1,i})^{f_{1,i}s_2} e(g, K_{2,i})^{f_{2,i}s_2}. \]

Then we have:

\[ \prod_{j=1}^{4} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i}) \]

\[ = e(g, g)^{\left(\sum v_i x_i \right) f_1 \alpha \Delta} e(g, g)^{\left(\sum v_i x_i z_i \right) f_2 \beta \Delta} \]

\[ \prod_{i=1}^{n} e(g, K_{1,i})^{f_{1,i}s_2} e(g, K_{2,i})^{f_{2,i}s_2} e(g, K_{3,i})^{z_{1,i}s_2} e(g, K_{4,i})^{z_{2,i}s_2}. \]

Also, since

\[ e(C_A, K_A) = e(g^{r_2}, g^2 \cdot \prod_{i=1}^{n} K_{1,i}^{-f_{1,i}} K_{2,i}^{-f_{2,i}} K_{3,i}^{-z_{1,i}} K_{4,i}^{-z_{2,i}}) \]

\[ e(C_B, K_B) = e(g^{s_1 \Delta}, \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})}) \]

we have

\[ \frac{e(C_A, K_A) e(C_B, K_B)}{\prod_{j=1}^{M} \prod_{i=1}^{n} e(C_{j,i}, K_{j,i})} = \frac{C_m}{e(g, g)^{\left(\sum v_i x_i \right) f_1 \alpha \Delta} + \left(\sum v_i x_i z_i \right) f_2 \beta \Delta}. \]

Therefore, the message $M$ will be returned iff $(\vec{v}, \vec{x}_1^2) = 0$ and $(\vec{v}, \vec{x}_2^2) = 0$, meaning the attributes list in user key $SK$ satisfies the access policy in the ciphertext $CT$.

* **Hidden Policy of Key Policy-ABBE**: By using this technique, we can apply the KP-ABBE with also supporting hidden policy.

### 5.1.2 Security Proof

**Theorem 8** Assume the Decision Bilinear Diffie-Hellman assumption and Decisional Linear Assumption hold in group $\mathbb{G}$, then our Anonymous-AND-ABBE scheme is selective IND-CPA secure and policy hiding.

Since our scheme actually uses the vector corresponding to an access policy to do the encryption. In order to prove that our scheme is policy hiding, we only need to prove that the adversary cannot tell which vector, among the two vectors $\vec{v}$ and $\vec{x}$ corresponding to $W_0$ and $W_1$ respectively, has been used to generate the ciphertext. In our proof we will consider two cases $M_0 = M_1$ and $M_0 \neq M_1$.

In the case $M_0 = M_1$, we only consider the following game sequence from Game$_1$ to Game$_5$. In this case, we only prove the property of attribute hiding. For the other case $M_0 \neq M_1$, we need to consider the whole proof from Game$_6$ to Game$_{10}$.
Below we first give a high level description of each game. In each game, we separate the vector used to generate \((C_A, C_B, C_{1,i}, C_{2,i})\) from the vector for \((C_A, C_B, C_{3,i}, C_{4,i})\). However, the same vector is used for both parts in \textbf{Game}_0 and \textbf{Game}_6.

\textbf{Game}_0 : The challenge ciphertext \(\text{CT}_0\) is generated under \((\vec{v}, \vec{v})\) and \(M_0\). The ciphertext \(\text{CT}_0\) is computed as follows:

\[
(M_0 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,v}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,v}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,v}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,v}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_1 : The challenge ciphertext \(\text{CT}_1\) is generated under \((\vec{v}, \vec{v})\) and a random message \(R \in \mathbb{G}_T\). The ciphertext \(\text{CT}_1\) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,v}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,v}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,v}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,v}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_2 : The challenge ciphertext \(\text{CT}_2\) is generated under \((\vec{v}, \vec{v})\) and a random message \(R \in \mathbb{G}_T\). The ciphertext \(\text{CT}_2\) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,v}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,v}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,v}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,v}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_3 : The challenge ciphertext \(\text{CT}_3\) is generated under \((\vec{v}, \vec{x})\) and a random message \(R \in \mathbb{G}_T\). The ciphertext \(\text{CT}_3\) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,v}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,v}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,v}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,v}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_4 : The challenge ciphertext \(\text{CT}_4\) is generated under \((\vec{v}, \vec{x})\) and a random message \(R \in \mathbb{G}_T\). The ciphertext \(\text{CT}_4\) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,v}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,v}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,v}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,v}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_5 : The challenge ciphertext \(\text{CT}_5\) is generated under \((\vec{x}, \vec{x})\) and a random message \(R \in \mathbb{G}_T\). The ciphertext \(\text{CT}_5\) is computed as follows:

\[
(R', g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,x}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,x}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,x}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,x}^{\beta}\}_{i=1}^n).
\]

\textbf{Game}_6 : The challenge ciphertext \(\text{CT}_6\) is generated under \((\vec{x}, \vec{x})\) and message \(M_1 \in \mathbb{G}_T\). The ciphertext \(\text{CT}_6\) is computed as follows:

\[
(M_1 \cdot Y^{-s_2}, g^{s_2}, g_1^{s_1}, \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,x}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,x}^{\alpha}\}_{i=1}^n, \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,x}^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,x}^{\beta}\}_{i=1}^n).
\]

\textbf{Indistinguishability between Game}_0 and \textbf{Game}_1

Suppose that there exists an adversary \(A\) which can distinguish the two games with a non-negligible advantage \(\epsilon\), we construct another algorithm \(B\) which uses \(A\)
to solve the Decision Bilinear Diffie-Hellman problem also with advantage $\epsilon$. On input $(g, A = g^a, B = g^b, C = g^r, Z) \in \mathbb{G}_2$, $\mathcal{B}$ simulates the game for $\mathcal{A}$ as follows.

- **Setup:** $\mathcal{B}$ selects random elements $\gamma_1, \gamma_2, \theta_1, \theta_2, \lambda$, 
  \[
  \{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \\
  \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n, \text{ in } \mathbb{Z}_p.
  \]
  Then it selects a random $\Delta \in \mathbb{Z}_p$ to obtain $\{u_{2,i}\}_{i=1}^n$, $\{w_{2,i}\}_{i=1}^n$ under the condition:
  \[
  \Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i} \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.
  \]
  Then for $i = 1$ to $n$, $\mathcal{B}$ sets:
  \[
  U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, T_{1,i} = (g^b)^{v_{1,i}} g^{t_{1,i}}, T_{2,i} = (g^b)^{v_{2,i}} g^{t_{2,i}}, \\
  W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}}, Z_{1,i} = (g^b)^{v_{1,i}} g^{z_{1,i}}, Z_{2,i} = (g^b)^{v_{2,i}} g^{z_{2,i}}.
  \]
  and
  \[
  V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2}, g_1 = g^\Delta, Y = e(g^a, g^b)^{-\Delta} \cdot e(g, g)^\lambda.
  \]
  Each public key component is distributed properly following the random exponents:
  \[
  \overline{t}_{1,i} = v_i \gamma_1 b + t_{1,i}, \overline{t}_{2,i} = v_i \gamma_2 b + t_{2,i}, \overline{z}_{1,i} = v_i \theta_1 b + z_{1,i}, \overline{z}_{2,i} = v_i \theta_2 b + z_{2,i}, \\
  g_2 = g^{-ab\Delta} g^\lambda.
  \]

- **Key Generation Phase 1 & 2:** $\mathcal{A}$ issues private key queries for the attribute list $L$. Consider a query with two vectors $\vec{y}_v = (y_{v_1}, \ldots, y_{v_n})$ and $\vec{y}_z = (y_{z_1}, \ldots, y_{z_n})$. $\mathcal{A}$ can request the private key query as long as $(\vec{v}, \vec{y}_v) = (\vec{v}, \vec{y}_z) = c_y \neq 0$.
  $\mathcal{B}$ picks random exponents $\{r_{1,i}\}_{i=1}^n$, $\{r_{2,i}\}_{i=1}^n$, and $f_1', f_2', r_1, r_2$. Then $\mathcal{B}$ computes:
  \[
  K_{1,i} = g^{-\gamma r_{1,i}} g^{\frac{\alpha}{2\gamma} + f_1'} y_{v_{1,i}} y_{u_{2,i}} = g^{\frac{\alpha}{2\gamma} y_{v_{1,i}} y_{u_{2,i}}} g^{-\gamma r_{1,i}} g^{f_1'} y_{v_{1,i}} y_{u_{2,i}} = g^{\frac{\alpha}{2\gamma} y_{v_{1,i}} y_{u_{2,i}}} \cdot K'_{1,i}, \\
  K_{2,i} = g^{\gamma r_{1,i}} g^{-\left(\frac{\alpha}{2\gamma} + f_1'\right) y_{v_{1,i}}} = g^{-\frac{\alpha}{2\gamma} y_{v_{1,i}} y_{u_{1,i}}} g^{\gamma r_{1,i}} g^{-f_1' y_{v_{1,i}}} = g^{-\frac{\alpha}{2\gamma} y_{v_{1,i}} y_{u_{1,i}}} \cdot K'_{2,i},
  \]
  which implicitly sets: $f_1 = \frac{\alpha}{2\gamma} + f_1'$. Next $\mathcal{B}$ computes:
  \[
  K_{3,i} = g^{-\theta r_{2,i}} g^{\left(\frac{\alpha}{2\gamma} + f_2'\right) y_{Z_{1,i}} y_{Z_{2,i}}} = g^{\frac{\alpha}{2\gamma} y_{Z_{1,i}} y_{Z_{2,i}}} g^{-\theta r_{2,i}} g^{f_2' y_{Z_{1,i}} y_{Z_{2,i}}} = g^{\frac{\alpha}{2\gamma} y_{Z_{1,i}} y_{Z_{2,i}}} \cdot K'_{3,i}, \\
  K_{4,i} = g^{\theta r_{2,i}} g^{-\left(\frac{\alpha}{2\gamma} + f_2'\right) y_{Z_{1,i}} y_{Z_{2,i}}} = g^{-\frac{\alpha}{2\gamma} y_{Z_{1,i}} y_{Z_{2,i}}} g^{\theta r_{2,i}} g^{-f_2' y_{Z_{1,i}} y_{Z_{2,i}}} = g^{-\frac{\alpha}{2\gamma} y_{Z_{1,i}} y_{Z_{2,i}}} \cdot K'_{4,i},
  \]
  which implicitly sets: $f_2 = \frac{\alpha}{2\gamma} + f_2'$. In particular, $f_1 = f_2 = 0$, yielding $\vec{f}' = 0$.
which implicitly sets: \( f_2 = \frac{a}{2^{v_y}} + f'_2 \). Then \( K_B \) and \( K_A \) are computed as:

\[
K_B = \prod_{i=1}^n g^{-(r_1,i + r_2,i)} \\
K_A = g_2 \prod_{i=1}^n K_{1,i}^{-r_{1,i}} \cdot K_{2,i}^{-r_{2,i}} \cdot K_{3,i}^{-s_{1,i}} \cdot K_{4,i}^{-s_{2,i}}.
\]

For \( K_A \), we can compute:

\[
K_{1,i}^{-r_{1,i}} \cdot K_{2,i}^{-r_{2,i}} = (g^{\frac{a}{2^{v_y}}} y_i^{u_{1,i}} \cdot K'_1)_{1,i} \cdot (g^{\frac{a}{2^{v_y}}} y_i^{u_{1,i}} \cdot K'_2)_{1,i} \cdot (g^{\frac{a}{2^{v_y}}} y_i^{u_{1,i}} \cdot K'_3)_{1,i} \cdot (g^{\frac{a}{2^{v_y}}} y_i^{u_{1,i}} \cdot K'_4)_{1,i}.
\]

Similarly, we can compute:

\[
K_{3,i}^{-s_{1,i}} \cdot K_{4,i}^{-s_{2,i}} = g^{\frac{ab}{2^{v_y}}} y_i^{z_i} g^{\frac{a}{2^{v_y}}} y_i^{(u_{1,i} + u_{2,i})} \cdot (K_3)_{1,i}^{-s_{1,i}} \cdot (K_4)_{1,i}^{-s_{2,i}}.
\]

Since \( g_2 = g^{ab} g^{\lambda} \) then \( K_A \) can be computed as:

\[
K_A = g^{\lambda} \prod_{i=1}^n g^{\frac{a}{2^{v_y}}} y_i^{u_{1,i} + u_{2,i} - u_{1,i}} \cdot (K'_1)_{1,i}^{-r_{1,i}} \cdot (K'_2)_{1,i}^{-r_{2,i}} \cdot (K'_3)_{1,i}^{-s_{1,i}} \cdot (K'_4)_{1,i}^{-s_{2,i}}.
\]

\( B \) gives \( A \) the private key: \( SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n) \) for the queried vector \( \vec{y} \).

- **Challenge Ciphertext**: To generate a challenge ciphertext, \( B \) picks random \( s'_1, \alpha', \beta' \in \mathbb{Z}_p \). \( B \) implicitly sets:

\[
s_1 = s'_1, s_2 = c, \alpha = -bc + \alpha', \beta = -bc + \beta'.
\]

Then \( B \) sets \( A = g^c = g^{s_2}, B = g^{\Delta s_1} = g_{1,i}^{s_1} \). For \( i \) from 1 to \( n \), \( B \) computes:

\[
C_{1,i} = (g^{s_{1,i}})^{s_1} g^{v_{1,i} \gamma_1} g^{t_{1,i}} g^{v_{1,i} \gamma_1 (-bc + \alpha')} = U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,i}^{v_{1,i} \gamma_1},
\]

\[
C_{2,i} = (g^{s_{2,i}})^{s_1} g^{v_{2,i} \gamma_2} g^{t_{2,i}} g^{v_{2,i} \gamma_2 (-bc + \alpha')} = U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,i}^{v_{2,i} \gamma_2},
\]

\[
C_{3,i} = (g^{s_{1,i}})^{s_1} g^{v_{3,i}} g^{z_{1,i}} g^{v_{3,i} \gamma_1 (-bc + \beta')} = W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,i}^{v_{3,i}},
\]

\[
C_{4,i} = (g^{s_{2,i}})^{s_1} g^{v_{4,i}} g^{z_{2,i}} g^{v_{4,i} \gamma_2 (-bc + \beta')} = W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,i}^{v_{4,i}}.
\]

Next \( B \) computes \( C_m = Z^\Delta \cdot e(g, g^\lambda) \cdot M_0 \). If \( Z = e(g, g)^{abc} \) the challenge ciphertext is distributed in \( \text{Game}_0 \), otherwise if \( Z \) is randomly chosen in \( \mathbb{G}_T \), then the challenge ciphertext is distributed in \( \text{Game}_1 \). Hence, if \( A \) can
distinguish these two games, \( \mathcal{B} \) can solve the DBDH problem.

**Indistinguishability between Game\(_1\) and Game\(_2\)**

Suppose that there exists an adversary \( \mathcal{A} \) which can distinguish these two games with non-negligible advantage \( \epsilon \), we construct another algorithm \( \mathcal{B} \) which uses \( \mathcal{A} \) to solve the Decision Linear problem with advantage \( \epsilon \). On input \( (g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}_0 \), \( \mathcal{B} \) simulates the game for \( \mathcal{A} \) as follows.

- **Setup**: \( \mathcal{B} \) selects random elements \( \gamma_1, \gamma_2, \theta_1, \theta_2, \lambda \),

\[
\{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n
\]

in \( \mathbb{Z}_p \). Then it selects a random \( \Delta \in \mathbb{Z}_p \) to obtain \( \{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2 \) under the condition:

\[
\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i} \quad , \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}
\]

Then for \( i = 1 \) to \( n \), \( \mathcal{B} \) sets:

\[
U_{1,i} = (g^\gamma_{u_{1,i}}), U_{2,i} = (g^\gamma_{u_{2,i}}), T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}},
\]

\[
W_{1,i} = (g^\gamma_{w_{1,i}}), W_{2,i} = (g^\gamma_{w_{2,i}}), Z_{1,i} = g^{z_{1,i}}, Z_{2,i} = g^{z_{2,i}}
\]

\[
V_i = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, X_2 = g^{\theta_2}, g_1 = (g^\theta), g_2 = g^\lambda.
\]

Each public key component is distributed properly following the random exponents:

\[
\overline{u}_{1,i} = au_{1,i}, \overline{w}_{2,i} = aw_{2,i}, \overline{t}_{1,i} = aw_{1,i} + \theta_1 v_i, \overline{t}_{2,i} = aw_{2,i} + \theta_2 v_i,
\]

\[
\overline{z}_{1,i} = v_i \theta_1 b + \gamma_1 i, \overline{z}_{2,i} = v_i \theta_2 b + z_{2,i}.
\]

- **Key Generation Phase 1 & 2**: \( \mathcal{A} \) issues private key queries for the attribute list \( L \). Consider a query will be created two vectors \( \tilde{y}_V = (y_{v_1}, \ldots, y_{v_n}) \) and \( \tilde{y}_Z = (y_{z_1}, \ldots, y_{z_n}) \) following (5). \( \mathcal{B} \) picks random exponents \( \{r_{1,i}\}_{i=1}^n, \{r_{2,i}\}_{i=1}^n, \) and \( f_1, f_2 \). Then \( \mathcal{B} \) computes:

\[
K_{1,i} = g^{-\gamma_1(\gamma_{v_i} + \theta_1 r_{1,i})} g^{f_1 y_{v_i} u_{2,i}} = g^{-\gamma_1 \gamma_{v_i} v_i b} g^{-\gamma_1 r_{1,i}} g^{f_1 y_{v_i} u_{2,i}} = \gamma_v y_{v_i} b \cdot K'_{1,i},
\]

\[
K_{2,i} = g^{\gamma_1(\gamma_{v_i} + \theta_1 r_{1,i})} g^{-f_1 y_{v_i} u_{2,i}} = g^{-\gamma_1 \gamma_{v_i} v_i b} g^{\gamma_1 r_{1,i}} g^{-f_1 y_{v_i} u_{2,i}} = \gamma_v y_{v_i} b \cdot K'_{2,i},
\]

which implicitly sets: \( r_{1,i} = -y_{v_i} v_i b + r_{1,i} \). Next \( \mathcal{B} \) computes:

\[
K_{3,i} = g^{-\theta_1(\gamma_{y_{v_{i}} b + \theta_1 r_{1,i}})} g^{f_2 y_{z_i} w_{2,i}} = g^{-\theta_2 \gamma_{y_{v_{i}} b}} g^{-\theta_1 r_{1,i}} g^{f_2 y_{z_i} w_{2,i}} = \gamma_{y_{v_{i}} b} \cdot K'_{3,i},
\]

\[
K_{4,i} = g^{\theta_1(\gamma_{y_{z_i} b + \theta_1 r_{1,i}})} g^{-f_2 y_{z_i} w_{2,i}} = g^{\theta_1 \gamma_{y_{z_i} b}} g^{\theta_2 r_{2,i}} g^{-f_2 y_{z_i} w_{2,i}} = \gamma_{y_{z_i} b} \cdot K'_{4,i},
\]

which implicitly sets: \( r_{2,i} = y_{z_i} v_i b + a r_{2,i} \).

Then \( K_B \) and \( K_A \) are computed as:

76
Chapter 5. Anonymous Attribute Based Broadcast Encryption Supporting Multi-Gate Access Structure

\[ K_B = \prod_{i=1}^n g^{-(r_{i,1} + r_{i,2})} \prod_{i=1}^n g^{-(y_{i,1}v_i b + r_{i,1}' + y_{i,2} v_i b + ar_{i,2}')} \]
\[ = g^{-(r_{i,1}' + ar_{i,2}'')} \prod_{i=1}^n g^{-(r_{i,1}' + ar_{i,2}'')} . \]
\[ \tilde{K}_A = g_2 \prod_{i=1}^n K_{1,1,i}^{-1} K_{1,2,i}^{-1} K_{3,1,i}^{-1} K_{4,1,i}^{-1} . \]

For \( K_A \), we can compute:

\[ \tilde{K}_{1,1,i} K_{1,2,i}^{-1} \]
\[ = g^{-\gamma_2 v_i y_i b_{1,i}} g^{\gamma_1 v_i y_i b_{2,i}} \cdot (K_{1,1,i}')^{-t_{1,i}} \cdot (K_{2,1,i}')^{-t_{2,i}} \cdot K_{3,1,i}^{-1} K_{4,1,i}^{-1} \]
\[ = g^{\theta_2(v_i y_i b_{1,i})(-z_{1,i} - \theta_1 b_{v_i})} g^{(\theta_2 a_{r_{1,2}'},)(-z_{1,i} - \theta_1 b_{v_i})} \]
\[ = g^{\theta_1(v_i y_i b_{2,i})(-z_{2,i} - \theta_2 b_{v_i})} g^{(\theta_1 a_{r_{1,2}'},)(-z_{2,i} - \theta_2 b_{v_i})} \]
\[ = g^{-(v_i y_i b_{1,i} + ar_{1,2}'')} \Delta g^{(f_{2 y_i z_i z_i} w_{2,i})(-z_{1,i} - \theta_1 b_{v_i})} g^{(-f_{2 y_i z_i z_i} w_{1,i})(-z_{2,i} - \theta_2 b_{v_i})} . \]

Since \( g_2 = g^\lambda \) then \( \tilde{K}_A \) is computed as:

\[ \tilde{K}_A = g^\lambda \prod_{i=1}^n g^{-\gamma_2 v_i y_i b_{1,i}} g^{\gamma_1 v_i y_i b_{2,i}} \cdot (K_{1,1,i}')^{-t_{1,i}} \cdot (K_{2,1,i}')^{-t_{2,i}} \]
\[ = g^{-(v_i y_i b_{1,i} + ar_{1,2}'')} \Delta g^{(f_{2 y_i z_i z_i} w_{2,i})(-z_{1,i} - \theta_1 b_{v_i})} g^{(-f_{2 y_i z_i z_i} w_{1,i})(-z_{2,i} - \theta_2 b_{v_i})} . \]

\( B \) gives \( A \) the private key: \( SK = (K_A, K_B, \{K_{1,1,i}, K_{2,1,i}, K_{3,1,i}, K_{4,1,i}\}_{i=1}^n) \) for the queried vector \( \vec{y} \).

- **Challenge Ciphertext:** To generate a challenge ciphertext, \( B \) picks random \( s_1', \alpha' \in \mathbb{Z}_p \). \( B \) implicitly sets:

\[ s_1 = c, s_2 = d, \alpha = \alpha' . \]

Then \( B \) sets:

\[ A = g^d = g^{s_2}, B = (g^\alpha)^\Delta = g_1^{s_1} . \] For \( i \) from 1 to \( n \), \( B \) computes:

\[ C_{1,i} = (g^{s_1 a_{1,i}})^c (g^d)^{t_{1,i}} g^{\gamma_i (\alpha')} = U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,i}^{v_i \alpha'}, \]
\[ C_{2,i} = (g^{s_2 a_{2,i}})^c (g^d)^{t_{2,i}} g^{\gamma_i (\alpha')} = U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,i}^{v_i \alpha'}. \]

Next \( B \) computes for \( i \) from 1 to \( n \):

\[ C_{3,i} = (g^{s_1 a_{1,i}})^c (g^d)^{z_{1,i}} Z^{v_i \alpha}, C_{4,i} = (g^{s_2 a_{2,i}})^c (g^d)^{z_{2,i}} Z^{v_i \alpha} . \]

If \( Z = g^{(c+d)g^r} \) for \( r \) chosen randomly in \( \mathbb{Z}_p \), then \( B \) is simulating \textbf{Game} \( 1 \) with \( \beta = r \):

\[ C_{3,i} = (g^{s_1 a_{1,i}})^c (g^d)^{z_{1,i}} (g^{(c+d)g^r})^{v_i \alpha} = W_{1,i}^{s_1} Z_{1,i}^{s_2} X_{1,i}^{v_i \beta}, \]
\[ C_{4,i} = (g^{s_2 a_{2,i}})^c (g^d)^{z_{2,i}} (g^{(c+d)g^r})^{v_i \alpha} = W_{2,i}^{s_1} Z_{2,i}^{s_2} X_{2,i}^{v_i \beta} . \]
If \( Z = g^{b(c+d)} \), then \( \mathcal{B} \) is simulating \( \text{Game}_2 \)

\[
\begin{align*}
C_{3,i} &= (g^{aw_{1,i}})^c (g^{d})^{z_{1,i}} (g^{b(c+d)})^{\theta_1v_i} = W_{1,\bar{z}}^s Z_{1,i}^s, \\
C_{4,i} &= (g^{aw_{2,i}})^c (g^{d})^{z_{2,i}} (g^{b(c+d)})^{\theta_2v_i} = W_{2,\bar{z}}^s Z_{2,i}^s.
\end{align*}
\]

Therefore, if \( \mathcal{A} \) can distinguish the two games, \( \mathcal{B} \) can solve the DLIN problem. **Indistinguishability of Game\(_2\) and Game\(_3\)**

Suppose that there exists an adversary \( \mathcal{A} \) which can distinguish these two games with a non-negligible advantage \( \epsilon \), we construct another algorithm \( \mathcal{B} \) that uses \( \mathcal{A} \) to solve the Decision Linear problem with advantage \( \epsilon \). On input \( (g, g^a, g^b, g^{ac}, g^d, Z) \in \mathbb{G}_6 \), \( \mathcal{B} \) simulates the game for \( \mathcal{A} \) as follows.

- **Setup**: \( \mathcal{B} \) selects random elements \( \gamma_1, \gamma_2, \theta_1, \theta_2, \lambda, \{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n \) in \( \mathbb{Z}_p \). Then it selects a random \( \Delta \in \mathbb{Z}_p \) to obtain \( \{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2 \) under the condition:

\[
\Delta = \gamma_1 u_{2,i} - \gamma_2 u_{1,i}, \Delta = \theta_1 w_{2,i} - \theta_2 u_{1,i}.
\]

Then for \( i = 1 \) to \( n \), \( \mathcal{B} \) sets:

\[
\begin{align*}
U_{1,i} &= (g^a)^{u_{1,i}}, U_{2,i} = (g^a)^{u_{2,i}}, T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, \\
W_{1,i} &= (g^a)^{w_{1,i}} (g^b)^{\theta_1v_i}, W_{2,i} = (g^a)^{w_{2,i}} (g^b)^{\theta_2v_i}, \\
Z_{1,i} &= g^{z_{1,i}} (g^b)^{\theta_1v_i}, Z_{2,i} = g^{z_{2,i}} (g^b)^{\theta_2v_i}, V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^\lambda, X_2 = g^\theta_2, \\
g_1 &= (g^a)^{\Delta}, g_2 = g^\lambda.
\end{align*}
\]

Each public key component is distributed properly following the random exponents:

\[
\begin{align*}
\overline{u}_{1,i} &= au_{1,i}, \overline{u}_{2,i} = au_{2,i}, \overline{w}_{1,i} = aw_{1,i} + \theta_1 b v_i, \overline{w}_{2,i} = aw_{2,i} + \theta_2 b x_i, \\
\overline{z}_{1,i} &= v_i \theta_1 b + z_{1,i}, \overline{z}_{2,i} = v_i \theta_2 b + z_{2,i}.
\end{align*}
\]

- **Key Generation Phase \( 1 \) \& \( 2 \)**: \( \mathcal{A} \) issues private key queries for the attribute list \( L \). Consider a query will be created two vectors \( \vec{y}_V = (y_{V_1}, \ldots, y_{V_n}) \) and \( \vec{y}_Z = (y_{Z_1}, \ldots, y_{Z_n}) \) following (5). Notice that \( \mathcal{A} \) obey the restrictions defined in the model. That is \( \langle \vec{v}, \vec{y}_V \rangle - \langle \vec{v}, \vec{y}_Z \rangle = 0 \mod p \) if and only if \( \langle \vec{x}, \vec{y}_V \rangle \mod p \) and \( \langle \vec{x}, \vec{y}_Z \rangle \mod p \). There are two cases we need to consider.

  - **Case 1**: \( \langle \vec{v}, \vec{y}_V \rangle = 0 \mod p = \langle \vec{x}, \vec{y}_Z \rangle \mod p \). In this case, \( \mathcal{B} \) picks random
exponents \( \{r'_{1,i}\}_{i=1}^{n}, \{r'_{2,i}\}_{i=1}^{n}, \) and \( f_1, f_2. \) Then \( B \) computes:

\[
K_{1,i} = g^{\gamma_2(-v_i(y_i) v + r'_{1,i})} g^{f_1(y_i) w_{1,i}} = g^{\gamma_2 v_i(y_i) b} g^{\gamma_2 r'_{1,i}} g^{f_1(y_i) w_{2,i}} = g^{\gamma_2 v_i(y_i) b} \cdot K'_{1,i}.
\]

\[
K_{2,i} = g^{\gamma_1(-v_i(y_i) v + r'_{1,i})} g^{f_1(y_i) w_{1,i}} = g^{\gamma_1 v_i(y_i) b} g^{\gamma_2 r'_{1,i}} g^{f_1(y_i) w_{1,i}} = g^{\gamma_1 v_i(y_i) b} \cdot K'_{2,i},
\]

which implicitly sets: \( r_{1,i} = -y_i v_i b + r'_{1,i}. \)

Next \( B \) computes:

\[
K_{3,i} = g^{\theta_2(x_i y_i z_i b + a r'_{1,i})} g^{f_2(y_i) w_{2,i}} = g^{\theta_2 x_i (y_i z_i b)} g^{\gamma_2 r'_{2,i}} g^{f_2(y_i) w_{2,i}} = g^{\theta_2 x_i (y_i z_i b)} \cdot K'_{3,i}.
\]

\[
K_{4,i} = g^{\theta_1(x_i (y_i z_i b) + a r'_{1,i})} g^{f_2(y_i) w_{1,i}} = g^{\theta_1 x_i (y_i z_i b) g^{\theta_2 r'_{1,i}}} g^{f_2(y_i) w_{1,i}} = g^{\theta_1 x_i (y_i z_i b)} \cdot K'_{4,i},
\]

which implicitly sets: \( r_{2,i} = x_i y_i z_i b + a r'_{2,i} \)

\( B \) also compute \( K_A \) and \( K_B \) as follows.

\[
K_B = \prod_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})} = \prod_{i=1}^{n} g^{-(r'_{1,i} + a r'_{2,i})}.
\]

\[
K_A = g_2 \prod_{i=1}^{n} K_{1,i}^{-l_{1,i}} K_{2,i}^{-l_{2,i}} K_{3,i}^{-\gamma_{3,i}} K_{4,i}^{-\gamma_{4,i}}.
\]

For \( K_A, \) its components are computed as follows:

\[
K_{1,i}^{-l_{1,i}} K_{2,i}^{-l_{2,i}} = g^{\gamma_2 v_i y_i b_{l_{1,i}}} g^{\gamma_1 v_i y_i b_{l_{2,i}}} \cdot (K'_{1,i})^{-l_{1,i}} \cdot (K'_{2,i})^{-l_{2,i}}.
\]

\[
K_{3,i}^{-\gamma_{3,i}} K_{4,i}^{-\gamma_{4,i}} = g^{\theta_2 (x_i y_i z_i b (-z_{1,i} - \theta_1 b z_i)) g^{(\theta_2 a r'_{1,i}) (-z_{1,i} - \theta_1 b z_i)}}
\]

\[
g^{f_2 y_i z_i w_{2,i}) (-z_{1,i} - \theta_1 b x_i) g^{f_2 y_i z_i w_{1,i}) (-z_{1,i} - \theta_2 b x_i)}
\]

\[
g^{\theta_1 (x_i y_i z_i b (-z_{2,i} - \theta_1 b x_i)) g^{(\theta_1 a r'_{1,i}) (-z_{2,i} - \theta_2 b x_i)}}
\]

\[
g^{\gamma_{3,i} (v_i y_i z_i b + a r'_{2,i})} \Delta g^{f_2 y_i z_i w_{2,i}) (-z_{1,i} - \theta_1 b x_i) g^{f_2 y_i z_i w_{1,i}) (-z_{2,i} - \theta_2 b x_i)}
\]

Since \( g_2 = g^\lambda \) then \( K_A \) can be computed as:

\[
\lambda \prod_{i=1}^{n} g^{\gamma_2 v_i y_i b_{l_{1,i}}} g^{\gamma_1 v_i y_i b_{l_{2,i}}} \cdot (K'_{1,i})^{-l_{1,i}} \cdot (K'_{2,i})^{-l_{2,i}}
\]

\[
\cdot g^{(x_i y_i z_i b + a r'_{1,i}) \Delta g^{f_2 y_i z_i w_{2,i}) (-z_{1,i} - \theta_1 b x_i) \cdot g^f f_2 y_i z_i w_{1,i}) (-z_{2,i} - \theta_2 b x_i)}.
\]

\( B \) gives \( A \) the private key \( SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^{n}) \) for the queried vector \( \vec{y}. \)

- Case 2: \((\vec{v}, \vec{y}_V) = c_V \neq 0 \) and \((\vec{x}, \vec{y}_Z) = c_x \neq 0. \) In this case, \( B \) picks
random exponents \( \{r'_{1,i}\}^n_{i=1}, \{r'_{2,i}\}^n_{i=1} \), and \( f_1, f_2 \). Then \( B \) computes:

\[
K_{1,i} = g^{-\gamma_2(-c_x v_i y_i b + r'_{1,i})} g^{f_1 y_i v_i u_{2,i}} = g^{\gamma_2 c_x v_i y_i b} g^{-\gamma_2 r'_{1,i}} g^{f_1 y_i v_i u_{2,i}} \\
K_{2,i} = g^{\gamma_1 (c_x - v_i y_i b + r'_{1,i})} g^{f_1 y_i v_i u_{1,i}} = g^{-\gamma_1 c_x v_i y_i b} g^{-\gamma_2 r'_{1,i}} g^{-f_1 y_i v_i u_{1,i}}
\]

which implicitly sets: \( r_{1,i} = -c_x x_i y_i b + r'_{1,i} \).

Next \( B \) computes:

\[
K_{3,i} = g^{-\theta_2 c_x (x_i y_z b + a r'_{2,i})} g^{f_2 y_z w_{2,i}} = g^{-\theta_2 c_x y_z b} g^{-\gamma_2 r'_{2,i}} g^{f_2 y_z w_{2,i}} \\
K_{4,i} = g^{\theta_1 c_x (x_i y_z b + a r'_{2,i})} g^{-f_2 y_z w_{1,i}} = g^{\theta_1 c_x y_z b} g^\theta g^{-2 f_2 y_z w_{1,i}}
\]

which implicitly sets: \( r_{2,i} = c_x x_i y_z b + a r'_{2,i} \).

Then \( K_B \) and \( K_A \) are computed as follows:

\[
K_B = g^{-(r_1 + r_2)} \prod_{i=1}^n g^{-(r_{1,i} + r_{2,i})} = \prod_{i=1}^n g^{-(r'_{1,i} + a r'_{2,i})}.
K_A = g_2 \prod_{i=1}^n K_{1,i}^{-1} K_{2,i}^{-1} K_{3,i}^{-1} K_{4,i}^{-1}.
\]

For \( K_A \), the components are computed as follows:

\[
K_{1,i}^{-1}, K_{2,i}^{-1} = g^{-\gamma_2 c_x v_i y_i b} g^{\gamma_1 c_x v_i y_i b} \cdot (K_{1,i}^{-1})^{-1} \cdot (K_{2,i}^{-1})^{-1}
K_{3,i}^{-1} K_{4,i}^{-1} = g^{-\theta_2 c_x (x_i y_z b)(-z_{1,i} - \theta_1 b z_{1,i})} \cdot g^{-\theta_2 a r'_{2,i}} (-z_{1,i} - \theta_1 b x_{1,i}) \cdot g^{f_2 y_z w_{2,i}}(-z_{2,i} - \theta_2 b x_{2,i}) \cdot g^{\theta_1 c_x (x_i y_z b)(-z_{2,i} - \theta_2 b x_{2,i})} \cdot g^{\gamma_1 c_x v_i y_i b} \cdot g^{\gamma_2 r_1} \cdot g^{f_1 y_i v_i u_{2,i}} \cdot g^{f_1 y_i v_i u_{1,i}} \cdot g^{f_2 y_z w_{1,i}}(-z_{2,i} - \theta_2 b x_{2,i})
\]

Since \( g_2 = g^\lambda \) then \( K_A \) is computed as:

\[
K_A = g^\lambda \prod_{i=1}^n g^{-(\gamma_2 c_x v_i y_i b) \cdot (K_{1,i}^{-1})^{-1} \cdot (K_{2,i}^{-1})^{-1} \cdot (K_{3,i}^{-1})^{-1} \cdot (K_{4,i}^{-1})^{-1}} \cdot g^{-\gamma_1 c_x v_i y_i b} \cdot g^{f_2 y_z w_{2,i}}(-z_{1,i} - \theta_1 b x_{1,i}) \cdot g^{f_2 y_z w_{1,i}}(-z_{2,i} - \theta_2 b x_{2,i})
\]

\( B \) gives \( A \) the private key \( SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}^n_{i=1}) \) for the queried vector \( \vec{y} \).
• **Challenge Ciphertext:** To generate a challenge ciphertext, \( B \) picks random \( s_1, \alpha' \in \mathbb{Z}_n \). \( B \) implicitly sets:

\[
s_1 = c, s_2 = d, \alpha = \alpha'.
\]

Then \( B \) sets: \( A = g^d = g^{s_2}, B = (g^{nc})^\Delta = g_1^{s_1} \). For \( i \) from 1 to \( n \), \( B \) computes:

\[
C_{1,i} = (g^{au_{1,i}})^c(g^d)^{z_{1,i}}g^{nu_{1,i}(\alpha')} = U_{1,i}^{s_1}T_{1,i}^{s_2}V_{1,i}^{v_{i,\alpha}},
\]

\[
C_{2,i} = (g^{au_{2,i}})^c(g^d)^{z_{2,i}}g^{nu_{2,i}(\alpha')} = U_{2,i}^{s_1}T_{2,i}^{s_2}V_{2,i}^{v_{i,\alpha}}.
\]

Next \( B \) computes for \( i \) from 1 to \( n \):

\[
C_{3,i} = (g^{au_{1,i}})^c(g^d)^{z_{1,i}}Z_{b_{1,i}},
\]

\[
C_{4,i} = (g^{au_{2,i}})^c(g^d)^{z_{2,i}}Z_{b_{2,i}}.
\]

If \( Z = g^{b(c+d)} \) then, \( B \) is playing **Game**\(_2\) with \( A \)

\[
C_{3,i} = (g^{au_{1,i}})^c(g^d)^{z_{1,i}}(g^{b(c+d)}g^r)^{b_{1,i}} = W_{1,i}^{s_1}Z_{1,i}^{s_2}X_{1,i}^{x_{1,i}b},
\]

\[
C_{4,i} = (g^{au_{2,i}})^c(g^d)^{z_{2,i}}(g^{b(c+d)}g^r)^{b_{2,i}} = W_{2,i}^{s_1}Z_{2,i}^{s_2}X_{2,i}^{x_{2,i}b}.
\]

Otherwise, if \( Z = g^{b(c+d)}g^r \) for \( r \) chosen randomly in \( \mathbb{Z}_p \), then \( B \) is playing **Game**\(_3\) with \( A \) by setting \( \beta = r \)

\[
C_{3,i} = (g^{au_{1,i}})^c(g^d)^{z_{1,i}}(g^{b(c+d)})^{b_{1,i}} = W_{1,i}^{s_1}Z_{1,i},
\]

\[
C_{4,i} = (g^{au_{2,i}})^c(g^d)^{z_{2,i}}(g^{b(c+d)})^{b_{2,i}} = W_{2,i}^{s_1}Z_{2,i}.
\]

Therefore, if \( A \) can distinguish **Game**\(_2\) from **Game**\(_3\), then \( B \) can solve the DLIN problem.

The rest of the proof is similar to the above proofs:

• the indistinguishability between **Game**\(_3\) and **Game**\(_4\) can be proved in the same way as for **Game**\(_2\) and **Game**\(_3\);

• the indistinguishability between **Game**\(_4\) and **Game**\(_5\) can be proved in the same way as for **Game**\(_1\) and **Game**\(_2\);

• the indistinguishability of **Game**\(_5\) and **Game**\(_6\) can be proved in the same way as for **Game**\(_0\) and **Game**\(_1\).

\[\square\]

**Theorem 9** Assume the Decision Bilinear Diffie-Hellman assumption and Decisional Linear Assumption hold in group \( \mathbb{G} \), then our Anonymous OR-AND-ABBE scheme is selective IND-CPA secure and policy hiding.
5.1.3 Anonymous OR-AND-ABBE scheme

In this scheme, we consider two values for each attribute. In our construction, we require an \((n + m)\)—dimensional IPE scheme. We then present the construction for DNF access structure, since the CNF form can be conversed to the DNF.

**Setup**\((1^\lambda, N, U)\): Assume that we have \(N\) attributes in the universe, \(U\) denote the set of all user indices, and each attribute has two possible values: positive and negative access structures. Let \(N_2, N_3\) be three upper bounds defined as:

- \(N_2 \leq N\): the maximum number of positive attribute in an attribute set \(S\);
- \(N_3 \leq N\): the maximum number of negative attribute in an attribute set \(S\). The master entity choose a suitable encoding \(\tau\) sending each of the \(n\) indices \(ID \in U\) onto element \(\tau(ID) = x_1 \in \mathbb{Z}_p\), and choose \(k_1, \ldots, k_{2n}\) randomly in \(\mathbb{Z}_p\).

Next, the setup algorithm randomly generates \((g, \mathcal{G}, \mathcal{G}_T, p, e)\) and sets \(n = |U| + |N|\). It then chooses randomly \(\gamma_1, \gamma_2, \theta_1, \theta_2, \{u_{1,i}\}_{i=1}^n, \{t_{1,i}\}_{i=1}^n, \{t_{2,i}\}_{i=1}^n, \{w_{1,i}\}_{i=1}^n, \{z_{1,i}\}_{i=1}^n, \{z_{2,i}\}_{i=1}^n\) in \(\mathbb{Z}_p\) and \(g_2\) in \(\mathcal{G}\). Then it selects a random \(\Delta \in \mathbb{Z}_p\) and obtains \(\{u_{2,i}\}_{i=1}^n, \{w_{2,i}\}_{i=1}^n, w_2, u_2\) under the condition:

\[
\Delta = \gamma_1 w_{2,i} - \gamma_2 u_{1,i} \quad \Delta = \theta_1 w_{2,i} - \theta_2 w_{1,i}.
\]

For \(i\) from 1 to \(n\), it creates:

\[
U_{1,i} = g^{u_{1,i}}, U_{2,i} = g^{u_{2,i}}, W_{1,i} = g^{w_{1,i}}, W_{2,i} = g^{w_{2,i}},
\]

\[
T_{1,i} = g^{t_{1,i}}, T_{2,i} = g^{t_{2,i}}, Z_{1,i} = g^{z_{1,i}},
\]

\[
V_1 = g^{\gamma_1}, V_2 = g^{\gamma_2}, X_1 = g^{\theta_1}, V_2 = g^{\theta_2}.
\]

Next it sets \(g_1 = g^\Delta, Y = e(g, g_2)\), and the public key \(PK\) and master key \(MSK\) as

\[
PK = (g, \mathcal{G}, \mathcal{G}_T, p, e, g_1, Y, \{U_{1,i}, U_{2,i}, T_{1,i}, T_{2,i},
W_{1,i}, W_{2,i}, Z_{1,i}, Z_{2,i}\}_{i=1}^n, \{V_i, X_i\}_{i=1}^2)
\]

\[
MSK = (g_2, \{u_{1,i}, u_{2,i}, t_{1,i}, t_{2,i}, w_{1,i}, w_{2,i}, z_{1,i}, z_{2,i}\}_{i=1}^n,
\{v_i, x_i\}_{i=1}^2).
\]

**Encrypt**\((S, W, M, PK)\): Given a user index set \(S = \{ID_a, ID_b, ID_c, \ldots, ID_s\} \subseteq U\) and the access structure \(W\).

The access structure include the revoke user set as \(S = \{ID_a, ID_b, ID_c, \ldots, ID_s\} \subseteq U\).
with the total user set equal to \(|U|\), then apply the Viètè’s formula to construct:

\[
\begin{align*}
\tau_1(ID_a) + \tau_1(ID_b) + \tau_1(ID_c) + \ldots + \tau_1(ID_s) &= a_{|S|} \\
(\tau_1(ID_a)\tau_1(ID_b) + \tau_1(ID_a)\tau_1(ID_c) + \ldots + \tau_1(ID_a)\tau_1(ID_s)) &= a_{|S| - 1} \\
\ldots \\
\tau_1(ID_a)\tau_1(ID_b)\tau_1(ID_c)\ldots\tau_1(ID_s) &= a_0
\end{align*}
\]

(5.1)

We assume that the access structures can be store \(m_1 \leq m\) OR literals as:

\[
W = (\bigwedge_{i \in \{1, \ldots, m\}} A_i) \bigvee_{w_1} (\bigwedge_{i \in \{1, \ldots, m\}} A_i) \bigvee_{w_2} \ldots \bigvee_{w_{m_1}} (\bigwedge_{i \in \{1, \ldots, m\}} A_i)
\]

is computed as follows:

Each AND Gates +/- access structure \(W\) is computed as follows:

\[
\begin{align*}
\text{If } \begin{cases} 
\text{att}_j \text{ is } + : b_j = k_i \\
\text{att}_i \text{ is } - : b_j = k_{2i}
\end{cases} \text{ then } b_0 = \sum_{\text{att}_i \in \mathcal{W}} b_i. 
\end{align*}
\]

(5.2)

Then set each \(W_i = \sum_{\text{att}_i \in \mathcal{W}_i} b_j\).

Next apply the Consequence of Viètè’s formula to computes the whole access structure \(W\):

\[
\begin{align*}
W_1 + W_2 + \ldots + W_{m_1} &= b_{m_1} \\
W_1W_2 + W_1W_3 + \ldots + W_{m_1-1}W_{m_1} &= b_{m_1-1} \\
\ldots \\
W_1W_2 \ldots W_{m_1} &= b_0
\end{align*}
\]

(5.3)

We then produce a vector:

\[
\vec{v} = (a_0, a_1, \ldots, a_{|S|}, \ldots, 0_n, b_0, \ldots, b_{m_1}, \ldots, 0_m)
\]

which will be used for encryption.

The encryption algorithm chooses random \(s_1, s_2, \alpha, \beta \in \mathbb{Z}_p\) and creates the ciphertext as follows:

\[
\begin{align*}
C_m &= M \cdot Y^{s_2}, C_A = g^{s_2}, C_B = g_1^{s_1}, \\
\{C_{1,i}, C_{2,i}\} &= \{U_{1,i}^{s_1} T_{1,i}^{s_2} V_{1,i}^{\alpha}, U_{2,i}^{s_1} T_{2,i}^{s_2} V_{2,i}^{\alpha}\}, \\
\{C_{3,i}, C_{4,i}\} &= \{W_{1,i}^{s_1} Z_{1,i}^{s_2} X_1^{\beta}, W_{2,i}^{s_1} Z_{2,i}^{s_2} X_2^{\beta}\}.
\end{align*}
\]
Then ciphertext $CT$ is set as:

$$CT = (C_m, C_A, C_B, \{C_{1,i}, C_{2,i}, C_{3,i}, C_{4,i}\}_{i=1}^n).$$

**Key Gen**($MSK, ID, L$): Suppose that a user joins the system with the given user identity $ID$ and attribute list $L$. Each user joining the system has a unique $ID$, $\tau_1(ID) = x_0 \in \mathbb{Z}_p^*$ and an attribute list $L$.

For an attribute user list $L$, it computes:

If $\begin{cases} \text{att}_i \text{ is } + & r'_i = k_i \\ \text{att}_i \text{ is } - & r'_i = k_{2i} \end{cases}$ \hspace{1cm} (5.4)

Then set $b' = \sum_{\text{att}_i \in L} r'_i$, and it computes based on $b'$:

$$\begin{aligned}
 b'_0 &= b'^0 = 1 \\
 b'_1 &= b'^1 = b' \\
 b'_2 &= b'^2 \\
 \vdots \\
 b'_n &= b'^m
\end{aligned} \hspace{1cm} (5.5)$$

Then, we set as the root variant of polynomial degree $U$ and computes for $ID$, then produce a vector:

$$\vec{z} = ((ID^0, ID^1, ID^2, \ldots, ID^U), a'_0, a'_1, \ldots, a'_n, b'_0, \ldots, b'_m)$$

which will be used for key generation. The key generation algorithm chooses randomly $r_{i,1}, r_{i,2}$ for $i = 1$ to $n$, and $f_1, f_2, r_1, r_2 \in \mathbb{Z}_p$, and then creates the secret key as follows:

$$\begin{aligned}
\{K_{1,i}, K_{2,i}\} &= \{g^{-\theta_1 r_{i,1}}, g^{f_1 z_{i} w_{2,i}}, g^{\gamma_1 r_{i,1}}, g^{-f_1 z_{i} u_{1,i}}\}, \\
\{K_{3,i}, K_{4,i}\} &= \{g^{-\theta_2 r_{i,2}}, g^{f_2 z_{i} w_{2,i}}, g^{\theta_1 r_{i,2}}, g^{-f_2 z_{i} u_{1,i}}\}, \\
K_A &= g_2 \cdot \prod_{i=1}^n K_{1,i}^{-(r_{i,1} + 2r_{i,2})} K_{2,i}^{-r_{i,1}} K_{3,i}^{-r_{i,2}} K_{4,i}^{-(r_{i,1} + 2r_{i,2})}, \\
K_B &= \prod_{i=1}^n g^{-(r_{i,1} + r_{i,2})}.
\end{aligned}$$

The secret key is set as:

$$SK = (K_A, K_B, \{K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}\}_{i=1}^n).$$

**Decrypt**($SK, CT$): The decryption algorithm returns:

$$C_m \frac{e(C_A, K_A) \cdot e(C_B, K_B) \prod_{j=1}^4 \prod_{i=1}^n e(C_{j,i}, K_{j,i})}{e(C_{1,i}, K_{1,i}) \cdot e(C_{2,i}, K_{2,i}) \cdots e(C_{4,i}, K_{4,i})}$$
Correctness:

\[
e(C_{1,i}, K_{1,i}) = e(U_{1,i}^{-1}T_{1,i}V_{1,i}^{v_{1,i}}, g^{-\gamma_1 r_{1,i} s_{1,i}} f_{1} x_{1,i} u_{1,i})
\]

\[
e(g, g) \gamma_1 r_{1,i} s_{1,i} \cdot e(g, g)^{-\gamma_1 r_{1,i} s_{1,i}} \cdot e(g, K_{1,i})^{t_{1,i} s_{2,i}}
\]

\[
e(g, g) f_{1} x_{1,i} u_{1,i} s_{1,i} \cdot e(g, g) f_{1} x_{1,i} ^\alpha r_{1,i} s_{2,i}.
\]

\[
e(C_{2,i}, K_{2,i}) = e(U_{2,i}^{-1} T_{2,i} V_{2,i}^{v_{2,i}}, g^{-\gamma_1 r_{2,i} s_{2,i}} f_{1} x_{2,i} u_{1,i})
\]

\[
e(g, g) \gamma_1 r_{2,i} s_{2,i} \cdot e(g, g)^{-\gamma_1 r_{2,i} s_{2,i}} \cdot e(g, K_{2,i})^{t_{2,i} s_{2,i}}
\]

\[
e(g, g) f_{1} x_{2,i} u_{1,i} s_{1,i} \cdot e(g, g) f_{1} x_{2,i} ^\alpha r_{2,i} s_{2,i}.
\]

\[
\Pi_{j=1}^{2} \Pi_{i=1}^{n} e(C_{j,i}, K_{j,i})
\]

\[
= \Pi_{j=1}^{n} e(g, g)^{r_{1,i} s_{1,i}} \cdot e(g, g) f_{1} x_{1,i} ^\alpha r_{1,i} s_{2,i}
\]

\[
\cdot e(g, K_{1,i})^{t_{1,i} s_{2,i}} e(g, K_{2,i})^{t_{2,i} s_{2,i}}.
\]

Then we have:

\[
\Pi_{i=1}^{n} e(g, K_{3,i})^{z_{1,i} s_{2,i}} e(g, K_{4,i})^{z_{2,i} s_{2,i}}
\]

\[
= e(g, g)^{(\sum v_{i} x_{i}) f_{1} \alpha} e(g, g)^{(\sum v_{i} x_{i}) f_{2} \beta}
\]

\[
\cdot e(g, K_{1,i})^{t_{1,i} s_{2,i}} e(g, K_{2,i})^{t_{2,i} s_{2,i}} e(g, K_{3,i})^{z_{1,i} s_{2,i}}
\]

\[
e(g, K_{4,i})^{z_{2,i} s_{2,i}} e(g, g)^{r_{1,i} s_{1,i}} e(g, g)^{r_{2,i} s_{1,i}}.
\]

Also, since

\[
e(C_{A}, K_{A}) = e(g^{s_{2,i}} g_{2} \cdot \Pi_{i=1}^{n} K_{1,i}^{t_{1,i}} K_{2,i}^{t_{2,i}} K_{3,i}^{z_{1,i}} K_{4,i}^{z_{2,i}})
\]

\[
e(C_{B}, K_{B}) = e(g^{s_{1,i}} g_{1} \cdot \Pi_{i=1}^{n} g^{-(r_{1,i} + r_{2,i})})
\]

we have

\[
\frac{e(C_{A}, K_{A}) e(C_{B}, K_{B})}{\Pi_{i=1}^{n} e(C_{j,i}, K_{j,i})} = \frac{e(g, g)^{(\sum v_{i} x_{i}) f_{1} + f_{2} \alpha}}{M}.
\]

For the vector \( \vec{v} = (a_0, a_1, \ldots, a_{|S|}, 0_a, b_0, \ldots, b_m, 0_b) \) corresponding to the set of user indices \( S \) and access structure \( \mathbb{W} \) in the ciphertext \( CT \) and the vector \( \vec{z} = (ID^0, ID^1, ID^2, \ldots, ID^{|U|}, a_0, a_1, \ldots, a_n, b_0, \ldots, b_m) \) corresponding to the secret
key $SK$ in the OR-AND-AABBE, we have:

\[
\sum_{i=0}^{n} v_i z_i = \sum_{i=0}^{n} a_i z_i, \quad \sum_{i=n}^{n+m} v_i z_i = \sum_{i=n}^{n+m} b_{i-n} b'_{i-n} = \sum_{i=n+1}^{n+m} b^{-n} b'_{i-n}
\]

Therefore, the message $M$ will be returned if $(i) = 0$ meaning $ID$ belongs to the user indices $S$, and attribute list $L$ in $SK$ satisfies the access policy in the ciphertext $CT$, which imply $<v, z> = 0$

* By using this technique, we can also construct the other inversion scheme, in which access structure $\mathcal{W}$ is associated to the secret key and attributes list $L$ is embedded to the ciphertext.

**Constructions of secret keys:** Each AND access structure with positive and negative attributes in our second scheme, we assume $\sum_{\text{att} \in L} k_i \neq \sum_{\text{att} \in L^c} k_i$. If there exists $L$ and $L' (L \neq L')$ such that $\sum_{\text{att} \in L} k_i = \sum_{\text{att} \in L^c} k_i$, a user with attribute list $L$ can decrypt a ciphertext associated with $W$, where $L' \not\models W$ and $L \models W$.

Hence, the assumption holds with overwhelming probability:

\[
p(p-1)(p-(N-1)) \frac{p-N+1}{p^n} \frac{N}{p^N} = (1 - \frac{N-1}{p})^N > 1 - \frac{N(N-1)}{p} > 1 - \frac{N^2}{p}
\]

, where $p$ is the prime number which chosen in the first step, $N = \prod_{i=1}^{2n} \gamma_i$. If each secret key $k_i$ is chosen at random from $\mathbb{Z}_p$, then our assumption is natural.

### 5.1.4 Security Proof

**Security Analysis:** Assume that $\mathcal{A}$ announces the $\vec{v} = (a_0, a_1, \ldots, a_n, b_0, \ldots, b_{m_1}, \ldots, b_m)$ by (5.1), (5.2), (5.3) corresponding to the set of user indices $S^*$ and access structure $\mathcal{W}^*$, and $\mathcal{B}$ is a simulator, which use $\mathcal{A}$ to solve the security problem. When the adversary $\mathcal{A}$ issues private key query for the index $ID$, and the attribute list $L$, a query query will be created a vector $\vec{y} = (ID^0, ID^1, ID^2, \ldots, ID^{(U)}, a'_0, a'_1, \ldots, a'_n, b'_0, \ldots, b'_m)$ by (5.4) under restriction $<v, y> \neq 0$.

The computation to check the inner product of two vectors $\vec{v}$ and $\vec{z}$:

\[
\sum_{i=0}^{n} v_i y_i = \sum_{i=0}^{n} a_i \cdot y_i, \quad \sum_{i=n}^{n+m} v_i y_i = \sum_{i=n}^{n+m} b_{i-n} b'_{i-n} = \sum_{i=n+1}^{n+m} b^{-n} b'_{i-n}
\]

- **Case 1:** If $(i') = 0 \land (i'') = 0$, then $<v, y> = 0$ holds. $B$ aborts meaning
\[ ID \in S \text{ and } L \models W^* . \]

- **Case 2:** If \((i') \neq 0 \land (i'') \neq 0\), then \(< v, y >= 0\) still holds. There exists \(L\) and \(L'(L \neq L')\) such that \(\sum_{i \in L} k_i = \sum_{i \in L'} k_i\). Therefore, this probability is at most \(\frac{N^2}{p}\).

- **Case 3:** If \((i') \neq 0 \land (i'') \neq 0\), then \(< v, y > \neq 0\). \(B\) gives \(A\) the private key \(SK\) for the queried vector \(\vec{y}\).

The security proof can be obtained similar as the Anonymous-AND-ABBE scheme.

### 5.2 Comparisons

We give a comparison among in CP-ABBE schemes in Table 5.1. The schemes are compared in terms of the order of the underlying group, ciphertext size, decryption cost, access structure, and the anonymity. In the table, \(N\) - number of clauses in a policy, \(M\) - maximum number of attributes in given clause, \(k\) - number of attributes for given user, \(r\) - number of revoked users, \(k_{\max}\) - maximum number of attributes in access policy, \(n\) - total of indexed user, \(m\) - number of universe attributes, and \(N_1\) denotes the maximum number of wildcards.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Ciphertext</th>
<th>Decryption</th>
<th>Access Structure</th>
<th>Anonymity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[LS08]</td>
<td>(O(N \cdot R))</td>
<td>(O(N,k))</td>
<td>DNF policy</td>
<td>X</td>
</tr>
<tr>
<td>[AI09]</td>
<td>(O(k_{\max} + r))</td>
<td>(O(k_{\max} + r))</td>
<td>LSSS</td>
<td>X</td>
</tr>
<tr>
<td>[JK10]</td>
<td>(O(N \cdot M))</td>
<td>(O(M + m))</td>
<td>DNF, CNF policy</td>
<td>X</td>
</tr>
<tr>
<td>[PYSC15]</td>
<td>(O(1))</td>
<td>(O(3))</td>
<td>AND Gates +/-</td>
<td>(\sqrt{\ })</td>
</tr>
<tr>
<td>A-AND-ABBE</td>
<td>(O(4r + 4N_1 + 2))</td>
<td>(O(4r + 4N_1 + 2))</td>
<td>AND Gates +/- with wildcard</td>
<td>(\sqrt{\ })</td>
</tr>
<tr>
<td>A-OR-AND-ABBE</td>
<td>(O(n + m))</td>
<td>(O(n + m))</td>
<td>OR/ AND Gates +/-</td>
<td>(\sqrt{\ })</td>
</tr>
</tbody>
</table>

### 5.3 Summary

The first scheme HCP-ABBE can even hide the access policy against the legitimate decryptors. We proved that our second construction is secure under the Decisional Bilinear Diffie-Hellman and the Decision Linear assumptions. We then present Anonymous Attribute Based Broadcast Encryption for the OR/AND Gates positive, negative attribute access policy.
Embedding policy-based access control into modern encryption schemes is an interesting but challenging task that has been intensively studied by the cryptologic research community in recent years. Typical examples of such encryption schemes include Attribute-based Encryption (ABE) [GPSW06, LOS+10, Wat11b, LW12] and Predicate Encryption [BW07, KSW08a] schemes, which can be treated as special instances of a more general notion called Functional Encryption which was formalized by Boneh, Sahai, and Waters [BSW11].

As a special type of functional encryption, Hidden Vector Encryption (HVE) schemes [BW07, KSW08a, SLN+10, HHI+11] allow wildcards to appear in either the encryption attribute vector associated with a ciphertext or the decryption attribute vector associated with a user secret key. Similar to ABE schemes, we name the former Ciphertext Policy (CP-) HVE schemes and the latter Key Policy (KP-) HVE schemes. The decryption will work if and only if the two vectors match. That is, for each position, the two vectors must have the same letter (defined in an alphabet $\Sigma$) unless a wildcard symbol ‘⋆’ appears in one of these two vectors at that position. In this research, we focus on the construction of CP-HVE schemes.

**Related Work**

All the recent development on functional encryptions can be traced back to the earlier work on identity-based encryption which was introduced by Shamir [Sha84] and first realized by Boneh and Franklin [BF01b] and Cocks [Coc01]. One important extension of IBE is hierarchical IBE (HIBE) [BBG05], which allows users at a level to issue keys to those on the level below.

The notion of Anonymous IBE was introduced by Boneh et al. [BCOP04b] and later formalized by Abdalla et al. [ABC+08]. Compared with the normal IBE,
anonymous IBE supports the additional feature of identity/attribute hiding. That is, except the user holding the correct decryption key, no one is able to link a ciphertext with the identity string used to create that ciphertext.

In [ACD+06], Abdalla et al. also proposed another extension of IBE called Wildcared IBE (or WIBE for short). WIBE is closely related to CP-HVE except that the former does not consider the property of identity/attribute hiding when it was introduced in [ACD+06]. Abdalla et al. proposed several WIBE constructions based on the Waters HIBE [Wat05], the Boneh-Boyen HIBE [BB04], and the Boneh-Boyen-Goh HIBE [BBG05]. Recently, to address the identity hiding problem, Abdalla et al. also proposed an anonymous WIBE in [ABC+11].

In a predicate encryption system [BW07, KSW08a] for a (polynomial-time) predicate $P$, two inputs (besides some public parameters) are required in the encryption process, one is the message $M$ to be encrypted, and the other one is an index string $i$. A decryption key is generated based on a master secret and a key index $k$. The decryption key can successfully decrypt a valid encryption of $(i, M)$ if and only if $P(k, i) = 1$. IBE can be treated as a special type of predicate encryption where the predicate function simply performs an equality test, while for HVE the predicate function will ignore the positions where wildcard symbols ‘⋆’ have occurred when doing an equality test.

After the notion of hidden vector encryption was first proposed by Boneh and Waters in [BW07], several HVE schemes [KSW08a, SW08, IP08, BIP09], [SLN+10, Par11, HHI+11] have been proposed, most of which are key policy based (i.e., the wildcards ‘⋆’ appear in the decryption attribute vector). One common drawback in many early HVE schemes (e.g. [BW07, KSW08a, IP08, BIP09]) is that the ciphertext size and the decryption key size are large (linear in the length of the vector). In [SLN+10], Sedghi et al. proposed an HVE scheme that has constant decryption key size and short (but still not constant) ciphertext size. In [HHI+11], Hattori et al. introduced a formal definition for CP-HVE and proposed a CP-HVE scheme based on the anonymous HIBE proposed in [SKOS09] and the wildcared IBE proposed in [ACD+06]. Hattori et al.’s CP-HVE scheme also has a linear ciphertext size. To the best of our knowledge, there is no HVE scheme proposed in the literature that can achieve constant size ciphertext.

**Our Contributions**

We propose two ciphertext policy hidden vector encryption schemes with constant size ciphertext.

- Our first proposed scheme (CP-HVE1) is construct on bilinear groups with composite order $n = pq$ where $p, q$ are prime numbers. The security of the scheme is proven in the standard model under three complexity assumptions:
the Decisional \( L \)-composite Bilinear Diffie-Hellman Exponent (\( L \)-cBDHE) assumption, the \( L \)-composite Decisional Diffie-Hellman (\( L \)-cDDH) assumption, and the Bilinear Subspace Decision (BSD) assumption.

- Apart from first proposed scheme, we construct our second scheme (CP-HVE2) built on bilinear groups with prime order. Although, there is a tool to convert from composite order to prime order, we still represent our CP-HVE2 construction more efficient in constant size ciphertext policy compare to using that tool. Then our second scheme is proven under the Decisional \( L \)-Bilinear Diffie-Hellman Exponent (\( L \)-BDHE) assumption.

### 6.1 Definition

#### 6.1.1 Ciphertext-Policy Hidden Vector Encryption

A ciphertext-policy hidden vector encryption (CP-HVE) scheme consists of the following four probabilistic polynomial-time algorithms:

- **Setup**\((1^k, \Sigma, L)\): on input a security parameter \( 1^k \), an alphabet \( \Sigma \), a vector-length \( L \), the algorithm outputs a public key \( PK \) and master secret key \( MSK \).

- **Encryption**\((PK, \vec{v}, M)\): on input a public key \( PK \), a message \( M \), a vector \( \vec{v} \in \Sigma_L^* \) where \( \Sigma^* \) denotes \( \Sigma \cup \{\ast\} \), the algorithm outputs a ciphertext \( CT \).

- **KeyGen**\((MSK, \vec{x})\): on input a master secret key \( MSK \), a vector \( \vec{x} \in \Sigma_L \), the algorithm outputs a decryption key \( SK \).

- **Decryption**\((CT, SK)\): on input a ciphertext \( CT \) and a secret key \( SK \), the algorithm outputs either a message \( M \) or a special symbol \( \bot \).

#### 6.1.2 Security Model.

The security model for a CP-HVE scheme is defined via the following game between an adversary \( A \) and a challenger \( B \).

- **Init**: The adversary \( A \) chooses two target patterns,

\[
\vec{v}_0 = (v_{0,1}, v_{0,2}, \ldots, v_{0,L}) \quad \text{and} \quad \vec{v}_1 = (v_{1,1}, v_{1,2}, \ldots, v_{1,L})
\]

under the restriction that the wildcards \( \ast \) must appears at the same positions.

- **Setup**: The challenger \( B \) run **Setup**\((k, \Sigma, L)\) to generate the \( PK \) and \( MSK \). \( PK \) is then passed to \( A \).
• **Query Phase 1**: A adaptively issues key queries for \( \vec{\sigma} = (\sigma_1, \ldots, \sigma_L) \in \Sigma_L \) under the restriction that \( \vec{\sigma} \) does not match \( \vec{v}_0 \) or \( \vec{v}_1 \). That is, there exist \( i, j \in \{1, \ldots, L\} \) such that \( v^*_{0i} \neq * \wedge v^*_{0i} \neq \sigma_i \), and \( v^*_{1j} \neq * \wedge v^*_{1j} \neq \sigma_j \). The challenger runs KeyGen(\( MSK, \vec{\sigma} \)) and returns the corresponding decryption key to A.

• **Challenge**: A outputs two equal-length messages \( M^*_0, M^*_1 \). B picks \( \beta \leftarrow \{0, 1\} \) and runs Encrypt(\( PK, \vec{v}_\beta, M^*_\beta \)) to generate a challenge ciphertext \( C^* \). B then passes \( C^* \) to A.

• **Query Phase 2**: same as Query Phase 1.

• **Output**: A outputs a bit \( \beta' \) as her guess for \( \beta \).

Define the advantage of A as

\[
\text{Adv}^\text{CP-HVE}_A(k) = \Pr[\beta' = \beta] - 1/2.
\]

### 6.2 Construction

#### 6.2.1 CP-HVE Scheme 1

In this section, we present our first CP-HVE under composite order bilinear groups. Let \( \vec{v} \) denote the attribute vector associated with the ciphertext and \( \vec{z} \) the attribute vector associated with the user secret key. The expression of these two vectors is designed based on the idea The Viète’s formulas. To do encryption, we represent each component of the vector \( \vec{v} \) by \((g^v_i) \prod_{j \in J} (i-j)\) where \( J \) denotes all the wildcard positions and is attached to the ciphertext. Notice that \( \prod_{j \in J} (i-j) = \sum_{k=0}^{n} a_k i^k \) according to the Viète’s formulas. In the decryption process, based on \( J \), the decryptor can reconstruct the coefficients \( a_k \), and generate \( \prod_{j \in J} (i-j) \). In this way, whether \( v_i = z_i \) will not affect the decryption if \( i \in J \).

▶ **Setup** \((1^\lambda, \Sigma, L)\): The setup algorithm first chooses \( N < L \) where \( N \) is the maximum number of wildcards that are allowed in an encryption vector. It then picks large primes \( p, q \), generates bilinear groups \( G, G_T \) of composite order \( n = pq \), and selects generators \( g_p \in G_p, g_q \in G_q \). After that, it selects random elements:

- \( g, f, v, v', h_1, \ldots, h_L, h'_1, \ldots, h'_L, w \in G_p \)
- \( R_q, R_f, R_v, R_{v'}, R_{h_1}, \ldots, R_{h_L}, R'_{h_1}, \ldots, R'_{h'_L} \in G_q \)
Chapter 6. Hidden Vector Encryption with Constant Ciphertext Policy

and computes:

\[ G = gR_g, \ F = fR_f, \ V = vR_v, \ V' = v'R_{v'}, \]
\[ H_1 = h_1R_{h_1}, \ldots, \ H_L = h_LR_{h_L}, \]
\[ H'_1 = h'_1R_{h'_1}, \ldots, \ H'_L = h'_L R_{h'_L}, \]
\[ E = e(g, w). \]

Then it creates the public key and master secret key as:

\[ PK = \{g_p, g_q, G, F, V, V', (H_1, \ldots, H_L), (H'_1, \ldots, H'_L), E\}, \]
\[ MSK = \{p, q, g, f, v, v', (h_1, \ldots, h_L), (h'_1, \ldots, h'_L), w\}. \]

**Encrypt** \((PK, M, \vec{v}) = (v_1, \ldots, v_L) \in \Sigma_L^*\): Suppose that \(\vec{v}\) contains \(\tau \leq N\) wildcards which occur at positions \(J = \{j_1, \ldots, j_{\tau}\}\). The encryption algorithm first chooses:

\[ s \in \mathbb{Z}_n, \text{ and } Z_1, Z_2, Z_3, Z_4 \in \mathbb{G}_q. \]

Using Viète’s formulas, compute \(a_k\) for \(k = 1, 2, \ldots, \tau\), and \(t = a_0\). Then set

\[ C_0 = M \cdot E^s, \ C_1 = G^s Z_1, \ C_2 = F^s Z_2, \]
\[ C_3 = (\prod_{i=1}^{L} V^i H_i^\nu)_{i=1}^{L} (i-j_k) \cdot Z_3, \ C_4 = (\prod_{i=1}^{L} V'^i (H'_i)^\nu)_{i=1}^{L} (i-j_k) \cdot Z_4, \]
\[ J = \{j_1, j_2, \ldots, j_{\tau}\}, \]

and ciphertext \(CT = \{C_0, C_1, C_2, C_3, C_4, J\}\).

**KeyGen** \((MSK, \vec{z}) = (z_1, \ldots, z_L) \in \Sigma_L\): The key generation algorithm chooses \(r_1, r'_1, r_2\) randomly in \(\mathbb{Z}_n\), and computes

\[
\begin{align*}
K_1 &= g^{r_1}, \ K_2 = g^{r_1'}, \ K_3 = g^{r_2}, \\
K_{4,0} &= w(\prod_{i=1}^{L} h_i^z v)^{r_1}(\prod_{i=1}^{L} (h_i')^{z_1} v')^{r_1}, \ K_{4,1} = (\prod_{i=1}^{L} h_i^z v)^{r_2}(\prod_{i=1}^{L} (h_i')^{z_1} v')^{r_2}, \\
&\quad \ldots \\
K_{4,N} &= (\prod_{i=1}^{L} h_i^z v)^{r_N}(\prod_{i=1}^{L} (h_i')^{z_1} v')^{r_N}.
\end{align*}
\]

The secret key is \(SK = \{K_1, K_2, K_3, K_{4,0}, \ldots, K_{4,N}\}\).

**Decrypt** \((CT, SK)\): The decryption algorithm first applies the Viète’s formu-
las to compute

\[ a_{\tau-k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq \tau} j_{i_1} j_{i_2} \cdots j_{i_k}, 0 \leq k \leq \tau \]

and then outputs

\[
M = e(K_1, C_3) \cdot e(K_2, C_4) \cdot e(K_3, C_2) \cdot e\left( \prod_{k=0}^\tau K_{4,k}^{a_k}, C_1 \right)
\]

**Correctness:**

\[
e(K_1, C_5) = e(g^{r_1}, (\prod_{i=1}^L VH_i^{\mu_i}) \prod_{k=1}^s (i-j_k) ^{\frac{a_k}{\mu_0}} \cdot Z_3)
\]

\[
\begin{align*}
&= \prod_{i=1}^L e(g, v^{\mu_i}) \cdot e(g, h_i^{\frac{a_k}{\mu_0}}) \cdot e(g, h_i^{\frac{a_k}{\mu_0}}) \\
&= \prod_{i=1}^L e(g, v^{\mu_i}) \cdot e(g, h_i^{\frac{a_k}{\mu_0}}) \\
&= e(g, v) \cdot e(g, f) \cdot e(g, h_i^{\frac{a_k}{\mu_0}})
\end{align*}
\]

\[
e(K_2, C_4) = e(g^{r_1'}, (\prod_{k=1}^L V'(H'_i) \prod (i-j_k) ^{\frac{a_k}{\mu_0}} \cdot Z_4)
\]

\[
\begin{align*}
&= \prod_{i=1}^L e(g, v'^{\mu_i}) \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= \prod_{i=1}^L e(g, v'^{\mu_i}) \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= e(g, v') \cdot e(g, f') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}})
\end{align*}
\]

\[
e(K_3, C_2) = e(g^{r_2}, F^a Z_2) = e(g, f) ^{r_2 a_0}.
\]

\[
e\left( \prod_{k=0}^\tau K_{4,k}^{a_k}, C_1 \right) = e(w^{a_0}, (\prod_{k=0}^\tau \prod_{i=1}^L v^k a_k h_i^{i} v^k a_k v^k a_k) \prod_{k=0}^\tau \prod_{i=1}^L (h_i^{\ast} v^k a_k v^k a_k) ^{r_i f r_2 a_0}, G^a Z_1)
\]

\[
= e(g, v^{\mu_0}) \cdot e(g, f) ^{a_0 a_0} \cdot \prod_{i=1}^L e(g, h_i^{\frac{a_k}{\mu_0}}) \cdot e(g, v^{\mu_0}) \cdot e(g, h_i^{\frac{a_k}{\mu_0}}) \\
\]

\[
\begin{align*}
&= e(g, v) ^{a_0 a_0} \cdot e(g, f) ^{a_0 a_0} \cdot \prod_{i=1}^L e(g, h_i^{\frac{a_k}{\mu_0}}) \\
&= e(g, v') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= \prod_{i=1}^L e(g, v') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= e(g, v') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= \prod_{i=1}^L e(g, v') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}) \\
&= \prod_{i=1}^L e(g, v') \cdot e(g, h'_i^{\frac{a_k}{\mu_0}}).
\end{align*}
\]
Then we have:

\[
\frac{e(K_1, C_3) \cdot e(K_2, C_4) \cdot e(K_3, C_2) \cdot e(K_4, C_1)}{e(\prod_{k=0}^{\tau} K_{4,k}^{a_k}, C_1)} = \prod_{i=1}^{L} e(g, v)^{a_0} \cdot e(g, h_i)^{a_0} \cdot e(g, v')^{a_0} \cdot e(g, h_i')^{a_0} \cdot e(g, f)^{a_2} \cdot M \cdot e(g, w)^{a_2} = M.
\]

### 6.2.2 Security Proof

**Theorem 10** Our CP-HVE Scheme 1 is secure if the Decisional $\mathbb{L} - cBDHE$ assumption, the $\mathbb{L} - cDDH$ assumption, and the BSD assumption hold.

We prove Theorem 1 by the following sequence of games.

\[\begin{align*}
Game_0 & : [C_0, C_1, C_2, C_3, C_4] \\
Game_1 & : [C_0 \cdot R_p, C_1, C_2, C_3, C_4] \\
Game_2 & : [R_0, C_1, C_2, C_3, C_4] \\
Game_3 & : [R_0, C_1, C_2, R_3, C_4] \\
Game_4 & : [R_0, C_1, C_2, R_3, R_4],
\end{align*}\]

where $R_p$ is a randomly chosen from $\mathbb{G}_{T,p}$, $R_0$ is uniformly distributed in $\mathbb{G}_T$, and $R_3, R_4$ are uniformly distributed in $\mathbb{G}$.

We will prove the following Lemmas. Notice that in $Game_4$ the challenge ciphertext is independent of the message and the encryption vector, which means the adversary has no advantage in winning the game over random guess.

**Lemma 2** Assume that the Decisional $\mathbb{L} - cBDHE$ assumption holds, then for any PPT adversary, the difference between the advantages in $Game_0$ and $Game_1$ is negligible.

**Lemma 3** Assume that the BSD assumption holds, then for any PPT adversary, the difference between the advantages in $Game_1$ and $Game_2$ is negligible.

**Lemma 4** Assume that the $\mathbb{L} - cDDH$ assumption holds, then for any PPT adversary, the difference between the advantages in $Game_2$ and $Game_3$ is negligible.
Lemma 5 Assume that the $L$–cDDH assumption holds, then for any PPT adversary, the difference between the advantages in $Game_3$ and $Game_4$ is negligible.

Proof of Lemma 1

Assume that the Decisional $L$–cBDHE assumption holds, then for any PPT adversary, the difference between the advantages in $Game_0$ and $Game_1$ is negligible.

We assume that adversary $A$’s advantage has a difference $\epsilon$ between $Game_0$ and $Game_1$. The simulator $B$ will use $A$ to solve the Decisional $L$–cBDHE problem. $B$ is given a challenge instance $Z, T'$ of the problem, where $Z = (g_p, g_q, h, g_p^\alpha, \ldots, g_p^{\alpha^L}, g_p^{\alpha^{L+2}}, \ldots, g_p^{\alpha^2L})$ and $T'$ is either $T = \epsilon(g_p, h)^{\alpha^{L+1}}$ or $R \in \mathbb{G}_{T, p}$.

In the rest of the proof, we denote $W(\overrightarrow{v}) = \{1 \leq i \leq L|v_i = *\}$ and $\overrightarrow{W}(\overrightarrow{v}) = \{1 \leq i \leq L|v_i \neq *\}$, and $W(\overrightarrow{v})|_j^k$ as $i \in W(\overrightarrow{v})|j \leq i \leq k$.

- **Init**: $A$ declares two challenge alphabet vectors $\overrightarrow{v}_0^* = (v_{0,1}^*, \ldots, v_{0,L}^*)$ and $\overrightarrow{v}_1^* = (v_{1,1}^*, \ldots, v_{1,L}^*)$ under the restriction that $W(\overrightarrow{v}_0^*) = W(\overrightarrow{v}_1^*)$.
- **Setup**: In this phase, $B$ generates:

\[
\begin{align*}
\gamma, y, y', \psi, \{u_i, u'_i\}_{i \in [L]} & \leftarrow \mathbb{Z}_n, \\
R_g, R_f, R_e, R_{\psi}, R_{h_1}, \ldots, R_{h_L}, R_{h'_1}, \ldots, R_{h'_L} & \leftarrow \mathbb{G}_q.
\end{align*}
\]

Then $B$ flips a coin $\mu \in \{0, 1\}$ and sets:

\[
\begin{align*}
G &= g_p R_g, \\
F &= g_p^y R_f, \\
E &= \epsilon(g_p^\alpha, g_p^{\alpha^L}, g_p^\gamma), \\
V &= g_p^y \prod_{i \in W(\overrightarrow{v}_0^*)} g_p^{\alpha^{L+1-i}v_{0,i}} R_{\psi}, \\
V' &= g_p^y R_{\psi}, \\
\{H_i = g_p^{u_{-i}} - \alpha^{L+1-i} R_{h_1}\}_{i \in W(\overrightarrow{v}_0^*)}, \\
\{H'_i = g_p^{u_{-i}} R_{h'_1}\}_{i \in W(\overrightarrow{v}_0^*)}, \\
\{H''_i = g_p^{u_{-i}} R_{h''_1}\}_{i \in W(\overrightarrow{v}_0^*)}.
\end{align*}
\]

Then the corresponding elements of the master secret key are:

\[
\begin{align*}
g &= g_p, \\
f &= g_p^\psi, \\
h_i &= g_p^{u_{-i}} - \alpha^{L+1-i}, \\
\{h_i = g_p^{u_{-i}}\}_{i \in W(\overrightarrow{v}_0^*)}, \\
\{h'_i = g_p^{u_{-i}}\}_{i \in W(\overrightarrow{v}_0^*)}, \\
\{h''_i = g_p^{u_{-i}}\}_{i \in W(\overrightarrow{v}_0^*)}, \\
v' &= g_p^y, \\
v &= g_p^y \prod_{i \in W(\overrightarrow{v}_0^*)} g_p^{\alpha^{L+1-i}v_{0,i}}
\end{align*}
\]

The master key component $w$ is $g_p^{\alpha^{L+1+\alpha\gamma}}$. Since $B$ does not have $g_p^{\alpha^{L+1}}$, $B$ cannot compute $w$ directly.
• **Query Phase 1:** A queries the user secret key for \( \sigma_u = (\sigma_1, \sigma_2, \ldots, \sigma_u) \) that does not match the challenge patterns. Let \( k \in W(v^*_\mu) \) be the smallest integer such that \( \sigma_k \neq v^*_\mu,k \).

B needs to simulate the user key generation process. We start from \( K_{4,i} \).

\[
K_{4,0} = w(L) \prod_{i=1}^L h^\sigma_i(v)^{r_1} (v') \prod_{i=1}^L (h'_i)^{\sigma_i} f^{r_2}
\]

\[
= g_p^{L+1+\alpha} \cdot \prod_{i \in W(v^*_\mu)_{k+1}} g_p^{u_i - \alpha L+1-i} \sigma_i \cdot \prod_{i \in W(v^*_\mu)_{k+1}} (g_p^{u_i})^{\sigma_i} \cdot g_p^{y+ \sum_{i \in W(v^*_\mu)} \alpha L+1-i \cdot \sigma_i} f^{r_2}
\]

\[
(v' \prod_{i=1}^L (h'_i)^{\sigma_i})^{r_1} f^{r_2},
\]

where

\[
X = \sum_{W(v^*_\mu)_{k+1}} \alpha^{L+1-i} v^*_{\mu,i} + y + \sum_{W(v^*_\mu)_{k+1}} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{W(v^*_\mu)_{k+1}} u_i \sigma_i.
\]

Since

\[
\sum_{W(v^*_\mu)_{k+1}} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{W(v^*_\mu)_{k+1}} u_i \sigma_i = \sum_{W(v^*_\mu)_{k+1}} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^k u_i \sigma_i
\]

and recall \( \sigma_i = v^*_\mu,i \) for \( i \in W(v^*_\mu)_{k+1} \) and \( \sigma_k \neq v^*_\mu,k \). Hence, we have:

\[
X = \alpha^{L+1-k} \Delta_k + \sum_{W(v^*_\mu)_{k+1}} \alpha^{L+1-i} v^*_{\mu,i} + \sum_{i=1}^k x_i \sigma_i + y,
\]

where \( \Delta_k = v^*_{\mu,k} - \sigma_k \). Then we choose \( r'_1, r'_1, r_2 \) randomly in \( \mathbb{Z}_n \), and set

\[
r_1 = -\frac{\alpha^L}{\Delta_k} + r_1.
\]

\( K_{4,0} \) can be represented as:

\[
K_{4,0}
\]

\[
= g_p^{\alpha L+1+\alpha} \cdot g_p^{-\alpha^{L+1}} \cdot \prod_{i \in W(v^*_\mu)_{k+1}} \cdot \frac{\sum_{i \in W(v^*_\mu)_{k+1}} -\alpha^{L+1-i-k} v^*_{\mu,i}}{\Delta_k} \cdot g_p^{k \cdot \frac{\sum_{i \in W(v^*_\mu)_{k+1}} -\alpha^{L+1-i-k} v^*_{\mu,i}}{\Delta_k}} \cdot (v' \prod_{i=1}^L h_i^{\sigma_i})^{r_1} f^{r_2}
\]

\[
= g_p^{\alpha L+1+\alpha} \cdot g_p^{-\alpha^{L+1}} \cdot \prod_{i \in W(v^*_\mu)_{k+1}} \cdot \frac{\sum_{i \in W(v^*_\mu)_{k+1}} -\alpha^{L+1-i-k} v^*_{\mu,i}}{\Delta_k} \cdot g_p^{k \cdot \frac{\sum_{i \in W(v^*_\mu)_{k+1}} -\alpha^{L+1-i-k} v^*_{\mu,i}}{\Delta_k}} \cdot (v' \prod_{i=1}^L h_i^{\sigma_i})^{r_1} f^{r_2}
\]

96
For \( \hat{k} = 1 \) to \( N \), we compute

\[
K_{4, \hat{k}} = \left( \prod_{W(v_i^* u_i)}^{L} (g_p^{u_i} - g_p^{\alpha L+1-i} \tau)^{y} \cdot \prod_{W(v_i^* u_i)}^{L} (g_p^{u_i}) \cdot g_p^{\sum_{i \in W(v_i^* u_i)}^{L} \alpha L+1-i \tau^*} \right)^{\hat{k}} \cdot \left( v' \prod_{i=1}^{L} (h_i^*)^{\alpha^i} \right)^{r_{\hat{k}}}. 
\]

Other elements in the key can also be simulated:

\[
K_1 = g_p^{r_1} = (g_p^{\alpha \hat{k}})^{-1} \cdot g_p^{r_1}, \quad K_2 = g_p^{r_1}, \quad K_3 = g_p^{r_2}.
\]

- **Challenge**: \( A \) sends two messages \( M_0, M_1 \) to \( B \). \( B \) generates \( Z_1, Z_2, Z_3, Z_4 \leftarrow \mathbb{G}_q \) and then sets using Viète’s formulas

\[
a_{\tau-k}(h) = (-1)^k \sum_{i_{1}<i_{2}<...<i_{k} \leq \tau} j_{i_{1}}j_{i_{2}}...j_{i_{k}}, \quad 0 \leq k \leq \tau.
\]

Let \( t = a_0 \). It creates ciphertext as:

\[
C_0 = M_b \cdot \gamma = C_1 \cdot C_2, \quad C_3 = (h)^{\psi} \cdot Z_3, \quad C_4 = ((h)^{\psi} \cdot Z_4).
\]

If \( T' = T = e(g_p, h)^{\alpha_{L+1}} \), where \( h = g_p^c \) for some unknown \( c \in \mathbb{Z}_p \), then

\[
C_0 = M_b \cdot (g_p^{c} \cdot g_p^{\alpha L+1}) \cdot e(g_p^{\alpha L+1} \cdot g_p^{\gamma}) = M_b \cdot e(g_p^{\alpha L+1}, g_p^{c}) = M_b \cdot E^c,
\]

\[
C_1 = (g_p^{c})^{1/t} \cdot Z_1 = G^c \cdot Z_1',
\]

\[
C_2 = (g_p^{c})^{\psi} \cdot Z_2 = F^c \cdot Z_2',
\]

\[
C_3 = ((h)^{\psi} \cdot Z_3)',
\]

\[
C_4 = ((h)^{\psi} \cdot Z_4)'.
\]

Hence, \( A \) is in \( Game_0 \). Otherwise, if \( T' = R_p = e(g_p, h)^{\alpha_{L+1}} \cdot R_p' \) for some random \( R_p' \in \mathbb{G}_{T,p} \), then \( A \) is in \( Game_1 \).

- **Query Phase 2**: Repeat Phase 1.
• **Guess:** A output $b' \in \{0, 1\}$. If $b' = b$ then $B$ outputs 1; otherwise $B$ outputs 0.

Let $Adv_B(k)$ be the advantage of $B$ to solve the $L$–wDBDHI problem, and $Adv_A^{Game_0}(k)$, $Adv_A^{Game_1}(k)$ be the advantages of $A$ in $Game_0$ and $Game_1$. Then we have

$$Adv_B(\lambda) = |\Pr[B \rightarrow 1|T' = T] - \Pr[B \rightarrow 1|T'' = R]|$$

$$= |\Pr[B \rightarrow 1|Game_0] - \Pr[B \rightarrow 1|Game_1]|$$

$$= |(\frac{1}{2} + Adv_A^{Game_0}(k)) - (\frac{1}{2} + Adv_A^{Game_1}(k))|$$

$$= \epsilon.$$

**Proof of Lemma 2**

Assume that the BSD assumption holds, then for any PPT adversary, the difference between the advantages in $Game_1$ and $Game_2$ is negligible.

We assume that adversary $A$’s advantage has a difference $\epsilon$ between $Game_1$ and $Game_2$. The simulator $B$ is given a challenge instance $Z, T'$ of the BSD problem, where $Z = (g_p, g_q)$ and $T'$ is either $T' \leftarrow \mathcal{G}_{T,p}$ or $R \leftarrow \mathcal{G}_T$. $B$ simulates the game for $A$ as follows.

• **Init:** $A$ declares two challenge alphabet vectors.

• **Setup:** $B$ follows the setup algorithm and creates the public key and master secret key using $g_p$ and $g_q$.

• **Query Phase 1:** $A$ queries the user secret key for $\tilde{\sigma}$. $B$ simulates the key generation algorithm honestly by using the master secret key.

• **Challenge:** $A$ sends to message $M_0, M_1$ to $B$. $B$ flips a coin $b \leftarrow \{0, 1\}$ and returns a normal ciphertext of $M_b$, with the exception that $C_0$ is multiplied by $T'$. If $T' = T \leftarrow \mathcal{G}_{T,p}$ then $A$ is in $Game_1$; otherwise, if $T' = R \leftarrow \mathcal{G}_T$, then $A$ is in $Game_2$.

• **Query Phase 2:** Repeat Phase 1.

• **Guess:** $A$ output $b' \in \{0, 1\}$. If $b' = b$ then $B$ outputs 1; otherwise, $B$ outputs 0.

Let $Adv_B(k)$ be the advantage of $B$ to solve the BSD problem, and $Adv_A^{Game_0}(k)$, $Adv_A^{Game_1}(k)$ be the advantages of $A$ in $Game_1$ and $Game_2$. Then we have

$$Adv_B(k) = |\Pr[B \rightarrow 1|T' = T] - \Pr[B \rightarrow 1|T'' = R]|$$

$$= |\Pr[B \rightarrow 1|Game_0] - \Pr[B \rightarrow 1|Game_1]|$$

$$= |(\frac{1}{2} + Adv_A^{Game_0}(k)) - (\frac{1}{2} + Adv_A^{Game_1}(k))|$$

$$= \epsilon.$$
Proof of Lemma 3

Assume that the $L$-cDDH assumption holds, then for any PPT adversary, the difference between the advantages in $Game_2$ and $Game_3$ is negligible.

We assume that the adversary $A$ has difference $\epsilon$ in the advantages between $Game_2$ and $Game_3$. We use $A$ to solve the $L$-cDDH problem. The simulator $B$ is given a challenge instance $Z, T'$ of the $L$-cDDH problem, where $Z = (g_p, q, h, g_p^\alpha, \ldots, g_p^{\alpha_{L+1}}, g_p^{\alpha_{L+1}} \cdot R_1, g_p^{\alpha_{L+1+b}} \cdot R_2)$ and $T'$ is either $T = g_p^b \cdot R_3$ or $R \leftarrow \mathbb{G}$. $B$ simulates the game for $A$ as follows:

- **Init:** $A$ declares two challenge alphabet vectors $\vec{v}_0 = (v_{0,1}, \ldots, v_{0,L})$ and $\vec{v}_1 = (v_{1,1}, \ldots, v_{1,L})$ under the restriction that $W(\vec{v}_0) = W(\vec{v}_1)$.

- **Setup:** In this phase, $B$ generates:

  $\gamma, y, y', \psi, \{u_i, u'_i\}_{i \in [L]} \leftarrow \mathbb{Z}_n,$
  
  $w \leftarrow \mathbb{G}_p,$
  
  $R_g, R_f, R_{v'}, R_{v}, R_{h_1}, \ldots, R_{h_L}, R_{h'_1}, \ldots, R_{h'_L} \in \mathbb{G}_q.$

Then $B$ flips $\mu \in \{0, 1\}$ and sets:

\[
G = g_p R_g, F = g_p^y R_f, E = e(g_p^\alpha, g_p^\gamma),
\]

\[
V = (g_p^{\alpha_{L+1}} R_1) \cdot g_p^y \prod_{i \in W(\vec{v}_0)} g_p^{\alpha_{L+1-i} \cdot v_i} R_0 = g_p^{\alpha_{L+1}} \cdot g_p^y \prod_{i \in W(\vec{v}_0)} g_p^{\alpha_{L+1-i} \cdot v_i} R_0,
\]

\[
V' = g_p^{y'} \prod_{i \in W(\vec{v}_1)} g_p^{\alpha_{L+1-i} \cdot v'_i} R_0,
\]

\[
\{H_i = g_p^{u_{i} - \alpha_{L+1-i}} R_{h_i}\}_{i \in W(\vec{v}_0)}, \{H_i = g_p^{u_{i} - \alpha_{L+1-i}} R_{h_i}\}_{i \in W(\vec{v}_0)},
\]

\[
\{H'_i = g_p^{u'_{i} - \alpha_{L+1-i}} R'_{h_i}\}_{i \in W(\vec{v}_1)}, \{H'_i = g_p^{u'_{i} - \alpha_{L+1-i}} R'_{h_i}\}_{i \in W(\vec{v}_1)}.
\]

The corresponding master secret key components are:

$g = g_p, f = g_p^y, \{h_i = g_p^{u_i - \alpha_{L+1-i}}\}_{i \in W(\vec{v}_0)}, \{h_i = g_p^{u_i - \alpha_{L+1-i}}\}_{i \in W(\vec{v}_0)}, \{h'_i = g_p^{u'_{i} - \alpha_{L+1-i}}\}_{i \in W(\vec{v}_1)}, \{h'_i = g_p^{u'_{i} - \alpha_{L+1-i}}\}_{i \in W(\vec{v}_1)}.$

\[
v = g_p^{\alpha_{L+1+y}} \prod_{i \in W(\vec{v}_0)} g_p^{\alpha_{L+1-i} \cdot v_i}, v' = g_p^{y'} \prod_{i \in W(\vec{v}_1)} g_p^{\alpha_{L+1-i} \cdot v'_i}. \text{ The master key component } v \text{ corresponding to the system parameters is } g_p^{\alpha_{L+1+y}} \prod_{i \in W(\vec{v}_0)} g_p^{\alpha_{L+1-i} \cdot v_i}.
\]

Since $B$ does not have $g_p^{\alpha_{L+1}}$, $B$ cannot compute $v$ directly.

- **Query Phase 1:** $A$ queries the user secret key for $\vec{v}_k = (\sigma_1, \sigma_2, \ldots, \sigma_u)$ that does not match the challenge patterns. Let $k \in W(\vec{v}_0)$ be the smallest integer such that $\sigma_k \neq v^*_k$.
B first simulates $K_{4,1}$ as follows.

$$K_{4,0} = w\left(\prod_{i=1}^{L} h_i^{\sigma_i} v_i\right)^{r_1} \left(\prod_{i=1}^{L} (f_i^{\sigma_i} v_i')\right)^{r_2}$$

$$= w\left(\sum_{i \in W(v_{\mu}^k)} g_p^{(u_i - \alpha^{L+1-i})\sigma_i} \cdot \sum_{W(v_{\mu}^k)} (g_p^{u_i'} \sigma_i) \cdot g_p^{y_i'+\sum_{i=1}^{k} \alpha^{L+1-i} v_{\mu,i}^*} \cdot )^{r_1} \right.$$

$$\cdot (\sum_{i \in W(v_{\mu}^k)} g_p^{(u_i - \alpha^{L+1-i})\sigma_i} \cdot \sum_{W(v_{\mu}^k)} (g_p^{u_i'} \sigma_i) \cdot g_p^{y_i'+\sum_{i=1}^{k} \alpha^{L+1-i} v_{\mu,i}^*} \cdot )^{r_2}$$

$$\stackrel{\text{def}}{=} w(g_p^X)^{r_1} (g_p^Y)^{r_2},$$

where

$$X = \alpha^{L+1} + y + \sum_{i \in W(v_{\mu}^k)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{W(v_{\mu}^k)} (u_i - \alpha^{L+1-i})\sigma_i + \sum_{W(v_{\mu}^k)} u_i\sigma_i$$

and

$$Y = y' + \sum_{i \in W(v_{\mu}^k)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{W(v_{\mu}^k)} (u_i' - \alpha^{L+1-i})\sigma_i + \sum_{W(v_{\mu}^k)} u_i'\sigma_i.$$

Since

$$\sum_{W(v_{\mu}^k)} (u_i - \alpha^{L+1-i})\sigma_i + \sum_{W(v_{\mu}^k)} u_i\sigma_i = \sum_{W(v_{\mu}^k)} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^{k} u_i\sigma_i,$$

and

$$\sum_{W(v_{\mu}^k)} (u_i' - \alpha^{L+1-i})\sigma_i + \sum_{W(v_{\mu}^k)} u_i'\sigma_i = \sum_{W(v_{\mu}^k)} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^{k} u_i'\sigma_i,$$

and recall $\sigma_i = v_{\mu,i}^*$ for $i \in W(v_{\mu}^k)|_{k-1}$ and $\sigma_k \neq v_{\mu,k}^*$. Hence,

$$X = \alpha^{L+1} + \alpha^{L+1-k} \Delta_k + \sum_{W(v_{\mu}^k)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{i=1}^{k} u_i\sigma_i + y,$$

$$Y = \alpha^{L+1-k} \Delta_k + \sum_{W(v_{\mu}^k)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{i=1}^{k} u_i'\sigma_i + y'.$$

where $\Delta_k = v_{\mu,k}^* - \sigma_k$. B then randomly chooses $r_1, r_1', r_2$ in $\mathbb{Z}_n$, sets $r_1' =$
\[ -\frac{a^k r_1}{\Delta_k} + r_1^\gamma \]. Hence, we have:

\[
K_{4,0} = w(g_p X_i^{r_1}) (g_p Y) - \frac{a^k r_1}{\Delta_k} + r_1^\gamma \cdot f^{r_2}
\]

\[
= w(g_p)^{X_i^{r_1}} (g_p Y) - \frac{a^k r_1}{\Delta_k} + r_1^\gamma \cdot f^{r_2}
\]

\[
= w(g_p) \alpha^{L_1 - k \Delta_k + \sum_{\nu_1} L_{k+1} \alpha^{L_1 - L_1 + \sum_{\nu_1} u_i \sigma_i + y}}
\]

\[
= w(g_p) \alpha^{L_1 - k \Delta_k + \sum_{\nu_1} L_{k+1} \alpha^{L_1 - L_1 + \sum_{\nu_1} u_i \sigma_i + y}}
\]

\[
\cdot (v' \prod \limits_{i=1}^L (h_i) \sigma_i)^{r_1} \cdot f^{r_2}.
\]

Also, for \( \hat{k} \) to 1, B sets

\[
K_{4,\hat{k}} = (g_p) \alpha^{L_1 - k \Delta_k + \sum_{\nu_1} L_{k+1} \alpha^{L_1 - L_1 + \sum_{\nu_1} u_i \sigma_i + y}}
\]

\[
\cdot (\prod \limits_{i=1}^L (h_i) \sigma_i)^{r_1 \hat{k}}.
\]

And other elements can also be simulated as follows:

\[
K_1 = g_p^{r_1}, K_2 = g_p^{r_1} = g_p^{-\frac{a^k r_1}{\Delta_k} + r_1^\gamma}, K_3 = g_p^{r_2}.
\]

- **Challenge**: A sends two messages \( M_0, M_1 \) to B. Then B generates \( R_0 \leftarrow \mathbb{G}_T, Z_1, Z_2, Z_3, Z_4 \leftarrow \mathbb{G}_q \) and sets using Viète’s formulas:

\[
a_{\tau-k} = (-1)^k \sum_{i_1 < i_2 < \ldots < i_k \leq \tau} j_{i_1} j_{i_2} \ldots j_{i_k}, 0 \leq k \leq \tau.
\]

Let \( t = a_0 \). B creates ciphertext as:

\[
C_0 = R_0, C_1 = T_{\tau_1} \cdot Z_1, C_2 = T_{\psi} \cdot Z_2,
\]

\[
C_3 = (((g_p^{a_{L+1} R_2} (T^y + \sum_{i=1}^L u_i \nu_i, \prod_{i=1}^L (i-j_k)^{1 \gamma}) \cdot Z_3,
\]

\[
C_4 = ((T^y + \sum_{i=1}^L u_i \nu_i, \prod_{i=1}^L (i-j_k)^{1 \gamma}) \cdot Z_4.
\]
If \( T' = g_p^b g_q^c \) some unknown \( c \in \mathbb{Z}_q \), then we have

\[
C_1 = (g_p^b g_q^c)^{1/t} \cdot Z_1 = G^{b/t} \cdot Z'_1,
\]

\[
C_2 = (g_p^b g_q^c) \psi \cdot Z_2 = F^b \cdot Z'_2,
\]

\[
C_3 = ((g_p^{a+1} R_2 (g_p^b g_q^c)^{y + \sum_{i=1}^{L} u_i v_{\mu, i}^*})^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_3,
\]

\[
= ((g_p^{a+1} + y + \sum_{i=1}^{L} u_i v_{\mu, i}^*)^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_3,
\]

\[
C_4 = ((g_p^{b'} g_q^{c'})^{y + \sum_{i=1}^{L} u_i v_{\mu, i}^*})^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_4,
\]

\[
= ((g_p^{b'} g_q^{c'})^{y + \sum_{i=1}^{L} u_i v_{\mu, i}^*)^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_4,
\]

and \( CT \) is in Game\(_2\). Otherwise, if \( T' = R = g_p^b g_q^c \) for some \( b' \in \mathbb{Z}_p, c' \in \mathbb{Z}_q \), then

\[
C_1 = (g_p^b g_q^c)^{1/t} \cdot Z_1 = G^{b/t} \cdot Z''_1,
\]

\[
C_2 = (g_p^b g_q^c) \psi \cdot Z_2 = F^{b'} \cdot Z''_2,
\]

\[
C_3 = ((g_p^{a+1} R_2 (g_p^b g_q^c)^{y + \sum_{i=1}^{L} u_i v_{\mu, i}^*})^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_3,
\]

\[
= ((g_p^{a+1} + y + \sum_{i=1}^{L} u_i v_{\mu, i}^*)^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_3,
\]

\[
C_4 = ((g_p^{b'} g_q^{c'})^{y + \sum_{i=1}^{L} u_i v_{\mu, i}^*})^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_4,
\]

\[
= ((g_p^{b'} g_q^{c'})^{y + \sum_{i=1}^{L} u_i v_{\mu, i}*)^{i(i-j)_k})^{\frac{1}{\tau}} \cdot Z_4,
\]

where \( \delta = b - b' \) is uniformly distributed in \( \mathbb{Z}_n \) for \( R \) chosen randomly from \( \mathbb{G} \), and hence \( CT \) is in Game\(_3\).
• **Query Phase 2:** Repeat Phase 1.

• **Guess:** $A$ outputs $b' \in \{0, 1\}$. If $b' = b$, then $B$ outputs 1; otherwise $B$ outputs 0.

Let $\text{Adv}_B(k)$ be the advantage of $B$ in solving the $L$-cDDH problem, and $\text{Adv}_A^{\text{Game}_2}(k)$, $\text{Adv}_A^{\text{Game}_3}(k)$ be the advantage of $A$ in $\text{Game}_2$ and $\text{Game}_3$, respectively. Then we have

$$
\text{Adv}_B(k) = |\Pr[B \rightarrow 1 | T' = T] - \Pr[B \rightarrow 1 | T' = R]| \\
= |\Pr[B \rightarrow 1 | \text{Game}_2] - \Pr[B \rightarrow 1 | \text{Game}_3]| \\
= |(\frac{1}{2} + \text{Adv}_A^{\text{Game}_2}(k)) - (\frac{1}{2} + \text{Adv}_A^{\text{Game}_3}(k))| \\
= \epsilon.
$$

**Proof of Lemma 4**

The proof for Lemma 4 is almost the same as that for Lemma 3, except that we generate $V'$ as the role of $V$ in Lemma 3.

Assume that the $L$-cDDH assumption holds, then for any PPT adversary, the difference between the advantages in $\text{Game}_3$ and $\text{Game}_4$ is negligible.

We assume that the adversary $A$ has difference $\epsilon$ in the advantages between $\text{Game}_2$ and $\text{Game}_3$. We use $A$ to solve the $L$-cDDH problem. The simulator $B$ is given a challenge instance $Z, T'$ of the $L$-cDDH problem, where $Z = (g_p, g_q, h, g_p^0, \ldots, g_p^{L+1} \cdot R_1, g_p^{L+1} \cdot R_2)$ and $T'$ is either $T = g_p^b \cdot R_3$ or $R \leftarrow \mathbb{G}$. $B$ simulates the game for $A$ as follows:

- **Init:** $A$ declares two challenge alphabet vectors $v_0^* = (v^*_0, \ldots, v^*_0, L)$ and $v_1^* = (v^*_1, \ldots, v^*_L, L)$ under the restriction that $W(v_0^*) = W(v_1^*)$.

- **Setup:** In this phase, $B$ generates:

  \[ \gamma, y, y', \psi, \{u_i, u'_i\}_{i \in [L]} \xleftarrow{\$} \mathbb{Z}_n, \]
  \[ w \xleftarrow{\$} \mathbb{G}_p, \]
  \[ R_q, R_f, R_\psi, R_{\psi'}, R_{h_1}, \ldots, R_{h_L}, R_{h'_1}, \ldots, R_{h'_L} \in \mathbb{G}_q. \]

Then $B$ flips $\mu \in \{0, 1\}$ and sets:

\[ G = g_p R_g, F = g_p^y R_f, E = e(g_p^u, g_p^\psi), \]
\[ V = g_p^y \prod_{v \in W(v_0^*)} g_p^{\ell+1-\mu} v^*_v R_v, \]
\[ V' = (g_p^{L+1} R_1) \cdot g_p^{y'} \prod_{v \in W(v_1^*)} g_p^{\ell+1-\mu} v^*_v R_{v'}, \]
\[ \{H_i = g_p^{u_i - \alpha^{L+1-i}} R_{h_i} \}_{i \in W(v_0^*)}, \{H_i = g_p^{u_i} R_{h_i} \}_{i \in W(v_0^*)}, \]
\[ \{H'_i = g_p^{u'_i - \alpha^{L+1-i}} R_{h'_i} \}_{i \in W(v_1^*)}, \{H'_i = g_p^{u'_i} R_{h'_i} \}_{i \in W(v_1^*)}. \]
The corresponding master secret key components are: \( g = g_p, f = g_p^v, \{ h_i = g_p^{u_i - \alpha^{L+1-i}} \}_{i \in \mathcal{W}(v_p^i)} \), \( \{ h_i = g_p^{u_i} \}_{i \in \mathcal{W}(v_p^i)} \), \( \{ h'_i = g_p^{u'_i - \alpha^{L+1-i}} \}_{i \in \mathcal{W}(v_p^i)} \), \( h'_i = g_p^{u'_i} \}_{i \in \mathcal{W}(v_p^i)} \), \( v = g_p^y \prod_{i \in \mathcal{W}(v_p^i)} g_p^{\alpha^{L+1-i} v_{\mu,i}} \), \( v' = g_p^{\alpha^{L+1} + y'} \prod_{i \in \mathcal{W}(v_p^i)} g_p^{\alpha^{L+1-i} v_{\mu,i}} \). The master key component \( v' \) corresponding to the system parameters is \( g_p^{\alpha^{L+1} + y'} \prod_{i \in \mathcal{W}(v_p^i)} g_p^{\alpha^{L+1-i} v_{\mu,i}} \).

Since \( B \) does not have \( g_p^{\alpha^{L+1}} \), \( B \) cannot compute \( v' \) directly.

- **Query Phase 1**: \( A \) queries the user secret key for \( \sigma^Y_u = (\sigma_1, \sigma_2, \ldots, \sigma_u) \) that does not match the challenge patterns. Let \( k \in \mathcal{W}(v_p^i) \) be the smallest integer such that \( \sigma_k \neq v_{\mu,k}^* \).

\( B \) first simulates \( K_{4,i} \) as follows.

\[
K_{4,0} = w(\prod_{i=1}^L h_i^{\sigma_i} v^{r_1} (\prod_{i=1}^L (h'_i)^{\sigma_i} v'^{r'_1}) f^{r_2})
\]

\[
= w((\prod_{\mathcal{W}(v_p^i)} g_p^{u_i - \alpha^{L+1-i} \sigma_i}) \cdot (\prod_{\mathcal{W}(v_p^i)} g_p^{\alpha^{L+1} + y + \sum_{i=1} \alpha^{L+1-i} v_{\mu,i}}))^{r_1}
\]

\[
\cdot ((\prod_{\mathcal{W}(v_p^i)} g_p^{u'_i - \alpha^{L+1-i} \sigma_i}) \cdot (\prod_{\mathcal{W}(v_p^i)} g_p^{\alpha^{L+1} + y' + \sum_{i=1} \alpha^{L+1-i} v_{\mu,i}}))^{r'_1} f^{r_2}
\]

\[
def w(g_p^X)^{r_1} (g_p^Y)^{r'_1} f^{r_2}
\]

where

\[
X = y + \sum_{i \in \mathcal{W}(v_p^i)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{\mathcal{W}(v_p^i)} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v_p^i)} u_i \sigma_i
\]

and

\[
Y = \alpha^{L+1} + y' + \sum_{i \in \mathcal{W}(v_p^i)} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{\mathcal{W}(v_p^i)} (u'_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v_p^i)} u'_i \sigma_i.
\]

Since

\[
\sum_{\mathcal{W}(v_p^i)} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v_p^i)} u_i \sigma_i = \sum_{\mathcal{W}(v_p^i)} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^k u_i \sigma_i,
\]

and

\[
\sum_{\mathcal{W}(v_p^i)} (u'_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v_p^i)} u'_i \sigma_i = \sum_{\mathcal{W}(v_p^i)} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^k u'_i \sigma_i,
\]
and recall $\sigma_i = v_{\mu,i}^*$ for $i \in \mathbb{W}(v_{\mu}^*)|_{k-1}$ and $\sigma_k \neq v_{\mu,k}^*$. Hence,

$$X = \alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y,$$

$$Y = \alpha^{L+1} + \alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y'.$$

where $\Delta_k = v_{\mu,k}^* - \sigma_k$. B then randomly chooses $r_1, r_1', r_2$ in $\mathbb{Z}_m$, sets $r_1 = -\frac{\alpha^k r_1'}{\Delta_k} + r_1$. Hence, we have:

$$K_{4,0} = \frac{w(g_p^X)^{-\frac{\alpha^{k} r_1}{\Delta_k} + r_1} (g_p^Y)^{r_1'} \cdot f^{r_2}}{w(g_p^L \prod_{i=1}^{L} (h_i)^{r_i})^{r_i}}$$

$$= \frac{w(g_p^X)^{r_1} (g_p^{Y + \frac{\alpha^{k} r_1}{\Delta_k}})^{r_1'} \cdot f^{r_2}}{w(g_p^L \prod_{i=1}^{L} (h_i)^{r_i})^{r_i}}$$

$$= \frac{w(v \prod_{i=1}^{L} (h_i)^{r_i})^{r_i}}{w(g_p^L \prod_{i=1}^{L} (h_i)^{r_i})^{r_i}} \cdot \frac{\alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y'}{\alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y'}$$

Also, for $k = 1$ to $N$, B sets

$$K_{4,k} = \frac{\alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y'}{\alpha^{L+1-k}\Delta_k + \sum_{i=1}^{k}\mathbb{W}(v_{\mu,i}^*)\alpha^{L+1-i}v_{\mu,i}^* + \sum_{i=1}^{k}u_i\sigma_i + y'}$$

$$\cdot (\prod_{i=1}^{L} (h_i)^{r_i})^{r_{1i}^{1k}}.$$

And other elements can also be simulated as follows:

$$K_1 = g_p^{r_1} \cdot K_2 = g_p^{r_1'} = g_p^{\frac{\alpha^{k} r_1 + r_1'}{\Delta_k}} \cdot K_3 = g_p^{r_2}.$$

- **Challenge:** A sends two messages $M_0, M_1$ to B. Then B generates $R_0 \overset{R}{\leftarrow} \mathbb{G}_T$, $Z_1, Z_2, Z_3, Z_4 \overset{R}{\leftarrow} \mathbb{G}_q$ and sets using Viète’s formulas:

$$a_{\tau-k} = (-1)^{k} \sum_{i \leq i_1 < i_2 < \ldots < i_k \leq \tau} j_{i_1}j_{i_2} \ldots j_{i_k}, 0 \leq k \leq \tau.$$

Let $t = a_0$. B creates ciphertext as:

$$C_0 = R_0, C_1 = T^{\frac{1}{t}} \cdot Z_1, C_2 = T^{\frac{\psi}{t}} \cdot Z_2,$$

$$C_3 = \left((\left(T^{\frac{\gamma + \sum_{i=1}^{k} u_i\sigma_i}}{t}\right) \prod_{k=1}^{k} (i-j_k) \right)^{1} \cdot Z_3,$$

$$C_4 = \left((g_p^{\alpha^{L+1-k}R_2}T^{\frac{\gamma + \sum_{i=1}^{k} u_i\sigma_i}}{t}\right) \prod_{k=1}^{k} (i-j_k) \right)^{1} \cdot Z_4.$$
If $T' = g_p b g_q c$ some unknown $c \in Z_q$, then we have

\[
C_1 = (g_p b g_q c)^{1/t} \cdot Z_1 = G^{b/t} \cdot Z_1',
\]

\[
C_2 = (g_p b g_q c)^{\psi} \cdot Z_2 = F^b \cdot Z_2',
\]

\[
C_3 = \left( \left( (g_p b g_q c)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( \left( (g_p)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot \left( (g_q)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( (V \prod_{i=1}^{L} (H_i)^{v_i}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3',
\]

\[
C_4 = \left( \left( (g_{p+L+1} g_q)^{c_{1}}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_4 \right.
\]

\[
= \left( \left( (g_p)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot \left( (g_q)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( (V \prod_{i=1}^{L} (H_i)^{v_i}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3',
\]

\[
= \left( (V' \prod_{i=1}^{L} (H'_i)^{v_i}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_4',
\]

and $CT$ is in $\text{Game}_3$. Otherwise, if $T' = R = g_p b g_q c'$ for some $b' \in Z_p, c' \in Z_q$, then

\[
C_1 = (g_p b g_q c')^{1/t} \cdot Z_1 = G^{b'/t} \cdot Z_1',
\]

\[
C_2 = (g_p b g_q c')^{\psi} \cdot Z_2 = F^{b'} \cdot Z_2',
\]

\[
C_3 = \left( \left( (g_p b g_q c')^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( \left( (g_p)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot \left( (g_q)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( (V \prod_{i=1}^{L} (H_i)^{v_i}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3',
\]

\[
C_4 = \left( \left( (g_{p+L+1} g_q)^{c_{1}}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_4 \right.
\]

\[
= \left( \left( (g_p)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot \left( (g_q)^{y + \sum_{i=1}^{L} u_i v_i^*}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_3 \right.
\]

\[
= \left( (V' \prod_{i=1}^{L} (H'_i)^{v_i}\prod_{k=1}^{r} (i-j_k) \right)^{1/t} \cdot Z_4',
\]

where $\delta = b - b'$ is uniformly distributed in $Z_n$ for $R$ chosen randomly from
$\mathbb{G}$, and hence $CT$ is in $Game_4$.

- **Query Phase 2**: Repeat Phase 1.

- **Guess**: $A$ outputs $b' \in \{0, 1\}$. If $b' = b$, then $B$ outputs 1; otherwise $B$ outputs 0.

Let $Adv_B(k)$ be the advantage of $B$ in solving the $L$-$\epsilon$DDH problem, and $Adv_A^{Game_3}(k)$, $Adv_A^{Game_4}(k)$ be the advantage of $A$ in $Game_3$ and $Game_4$, respectively. Then we have

$$Adv_B(k) = |\Pr[B \rightarrow 1|T' = T] - \Pr[B \rightarrow 1|T' = R]|$$

$$= |\Pr[B \rightarrow 1|Game_3] - \Pr[B \rightarrow 1|Game_4]|$$

$$= \left|\left(\frac{1}{2} + Adv_A^{Game_3}(k)\right) - \left(\frac{1}{2} + Adv_A^{Game_4}(k)\right)\right|$$

$$= \epsilon.$$

### 6.2.3 CP-HVE Scheme 2

One straightforward approach to obtain a new CP-HVE scheme under prime-order bilinear groups is to apply the conversion technique introduced by Lewko [Lew12].

In this section, we present a new prime-order CP-HVE scheme that is more efficient than the converted scheme.

**Setup** $(1^\lambda, \Sigma, L)$: The setup algorithm chooses $N << L$ to be the maximum number of wildcards that are allowed in an encryption vector. Then it generates other system parameters including

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

$L + 1$ random elements $V, H_1, \ldots, H_L \in_R \mathbb{G}$

Then chooses randomly generator $g, w, f \in \mathbb{G}$

$$Y = e(g, w).$$

The public key and master secret key are set as

$$PK = (Y, V, (H_1, \ldots, H_L), g, f, p, \mathbb{G}, \mathbb{G}_T, e),$$

$$MSK = w.$$

**Encrypt** $(PK, M, \tilde{v} = (v_1, \ldots, v_L) \in \Sigma_L^n)$: Assume that $\tilde{v} = (v_1, \ldots, v_L)$ contains $\tau \leq N$ wildcards which occur at positions $J = \{j_1, \ldots, j_\tau\}$. The encryption algorithm chooses $s \in_R \mathbb{Z}_p$, and computes using Viete’s formulas $t = a_0$. It then computes

$$C_0 = MY^s, C_1 = g^s, C_2 = f^s, C_3 = \left(\prod_{i=1}^{L} VH_i^{v_i}\right)^{\prod_{k=1}^{\tau}(i-j_k)s},$$
and set the ciphertext $CT = (C_0, C_1, C_2, C_3, J = \{j_1, j_2, \ldots, j_\tau\})$.

**Key Generation** ($MSK, z = (z_1, \ldots, z_L) \in \Sigma_L$): given a key vector $z = (z_1, \ldots, z_L)$, the key generation algorithm chooses $r, r_1 \in \mathbb{Z}_p$, then it creates secret key $SK$ as:

$$
K_1 = g^r, K_2 = g^{r_1}, \begin{pmatrix} K_{3,0} = w(L \prod_{i=1}^{L} (H_i^zV)^{r f_{k_0}}) \\ K_{3,1} = (L \prod_{i=1}^{L} H_i^zV)^{ir} \\ \vdots \\ K_{3,N} = (L \prod_{i=1}^{L} H_i^zV)^{i^{N_r}} \end{pmatrix}
$$

**Decrypt** ($CT, SK$): The decryption algorithm first applies the Viete formulas on $J = \{j_1, \ldots, j_\tau\}$ included in the ciphertext to compute

$$a_{\tau-k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq \tau} j_{i_1} j_{i_2} \cdots j_{i_k}, \text{ for } 0 \leq k \leq \tau$$

and then outputs

$$M = \frac{e(K_1, C_3) \cdot e(K_2, C_2)}{e(\prod_{k=0}^{\tau} K_{3,k}^{a_k}, C_1)} \cdot C_0.$$  

**Correctness**

$$e(K_1, C_3) = e(g^r, ((L \prod_{i=1}^{L} V H_i^{z_i})^{\prod_{k=1}^{\tau} (i-j_k) s_0})^{s_0} = L \prod_{i=1}^{L} e(g, V)^{s_0 \prod_{k=1}^{\tau} (i-j_k)}$$

$$e(K_2, C_2) = e(g^{r_1}, f^{s}) = e(g, f)^{r_1 s}$$

$$e(\prod_{k=0}^{\tau} K_{3,k}^{a_k}, C_1) = e(w^{s_0} (L \prod_{k=0}^{\tau} \prod_{i=1}^{L} H_i^{z_i i^k a_k} V^{i^k a_k})^{r f_{k_0}} g^{s_0})$$

$$= e(g, w)^{s_0} \cdot e(g, f)^{s r_1 s_0} \cdot L \prod_{i=1}^{L} e(g, V)^{s r \prod_{k=1}^{\tau} (i-j_k) s_0} e(g, H_i)^{s r \prod_{k=1}^{\tau} (i-j_k) z_i}$$

$$= e(g, w)^s \cdot e(g, f)^{s r_1} \cdot L \prod_{i=1}^{L} e(g, V)^{s r \prod_{k=1}^{\tau} (i-j_k) s_0} e(g, H_i)^{s r \prod_{k=1}^{\tau} (i-j_k) z_i}.$$
Then we have:
\[
e(k_1, c_3, k_2, c_2, c_0) = \frac{Me(g, w)^x \prod_{i=1}^L e(g, V)^x}{\prod_{i=1}^L e(g, H_i)} = M.
\]

### 6.2.4 Security Proof of CCP-HVE2

**Theorem 11** Assume decision L-BDHE assumption holds in \( \mathbb{G} \), then our CP-HVE Scheme 2 is secure.

Suppose that there exists an adversary \( A \) which can attack our scheme with non-negligible advantage \( \epsilon \), we construct another algorithm \( B \) which uses \( A \) to solve the decision L-BDHE problem. On input \( (g, h, \vec{y}, g_{\alpha}, L = (g_1, g_2, \ldots, g_L, g_{L+2}, \ldots, g_{2L}), T) \), where \( g_i = g^\alpha \) and for some unknown \( \alpha \in \mathbb{Z}_p^* \). The goal of \( B \) is to determine whether \( T = e(g_{L+1}, h) \) or not.

In the rest of the proof, we denote \( W(\vec{v}) = \{1 \leq i \leq L | v_i = */ \} \) and \( W(\vec{v}) = \{1 \leq i \leq L | v_i \neq * \} \), and \( W(\vec{v}) \mid j \) as \( \{i \in W(\vec{v}) | j \leq i \leq k \} \).

\( B \) simulates the game for \( A \) as follows:

- **Init**: \( A \) declares two challenge alphabet vectors \( \vec{v}_0^3 \in \Sigma^*_L \) and \( \vec{v}_1^3 \in \Sigma^*_L \) under the restriction that \( W(\vec{v}_0^3) = W(\vec{v}_1^3) \). \( B \) flips a coin \( \mu \in \{0, 1\} \). For simplicity we denote \( \vec{v}_\mu = (v_1^\mu, v_2^\mu, \ldots, v_L^\mu) \).

- **Setup**: \( B \) chooses \( N << L \), and random values \( \gamma, y, \psi, u_1, \ldots, u_L \in R \mathbb{Z}_p \) and sets
  \[
  Y = e(g^\alpha, g^\beta \cdot g^\gamma), f = g^\psi,
  V = g^y \prod_{i \in W(\vec{v}_0^\mu)} g^{\alpha^{L+1-i} \cdot v_{0,i}^\mu},
  \{H_i = g^{u_i} \mid i \in W(\vec{v}_0^\mu)\}, \{H_i = g^{u_i} \mid i \in W(\vec{v}_1^\mu)\}.
  \]

The master key component \( w \) is \( g^{\alpha^{L+1} + \alpha \gamma} \). Since \( B \) does not have \( g^{\alpha^{L+1}} \), \( B \) cannot compute \( w \) directly.

- **Query Phase 1**: \( A \) queries the user secret key for \( \vec{\sigma}_u = (\sigma_1, \sigma_2, \ldots, \sigma_u) \) that does not match the challenge patterns. Let \( k \in W(\vec{v}_0^\mu) \) be the smallest integer such that \( \sigma_k \neq v_{\mu,k}^\mu \).
Chapter 6. Hidden Vector Encryption with Constant Ciphertext Policy

$B$ needs to simulate the user key generation process. We start from $K_{3,0}$:

$$K_{3,0} = w^L \prod_{i=1}^{L} H^i V^r f^{r_1}$$

$$= g^{\alpha^{L+1} + \alpha \gamma} \left( \prod_{\mathcal{W}(v^*_i)^k} g^{u_i - \alpha^{L+1-i}} \cdot \prod_{\mathcal{W}(v^*_i)^k} (g^{u_i})^{\sigma_i} \cdot g^{y + \sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} v_{\mu,i}^*} \right)^r f^{r_1}.$$  

$$\text{def} = g^{\alpha^{L+1} + \alpha \gamma} (g^{X})^r f^{r_1}$$

where

$$X = \sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} v_{\mu,i}^* + y + \sum_{\mathcal{W}(v^*_i)^k} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v^*_i)^k} u_i \sigma_i.$$  

Since

$$\sum_{\mathcal{W}(v^*_i)^k} (u_i - \alpha^{L+1-i}) \sigma_i + \sum_{\mathcal{W}(v^*_i)^k} u_i \sigma_i = \sum_{\mathcal{W}(v^*_i)^k} (-\alpha^{L+1-i} \sigma_i) + \sum_{i=1}^{k} u_i \sigma_i$$

and recall $\sigma_i = v_{\mu,i}^*$ for $i \in \mathcal{W}(v^*_i)^k$ and $\sigma_k \neq v_{\mu,k}^*$. Hence, we have

$$X = \alpha^{L+1-k} \Delta_k + \sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} v_{\mu,i}^* + \sum_{i=1}^{k} x_i \sigma_i + y$$

where $\Delta_k = v_{\mu,k}^* - \sigma_k$. Then we choose $\hat{r}$, $r_1$ randomly in $\mathbb{Z}_n$, and set $r = -\alpha^k + \hat{r}$. $K_{3,0}$ can be represented as

$$K_{3,0} \quad = \quad g^{\alpha^{L+1} + \alpha \gamma} \cdot g^{\Delta_k} \cdot g^{\sum_{\mathcal{W}(v^*_i)^k} \frac{-\alpha^{L+1-i} x_i \sigma_i}{\Delta_k}} \cdot g^{\sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} x_i v_{\mu,i}^*} \cdot g^{y + \sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} \sigma_i} \cdot (V \prod_{i=1}^{k} H_i^{\sigma_i})^r f^{r_1}$$

$$= \quad g^{\alpha^{L+1} + \alpha \gamma} \cdot g^{\sum_{\mathcal{W}(v^*_i)^k} \frac{-\alpha^{L+1-i} x_i \sigma_i}{\Delta_k}} \cdot g^{y + \sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} \sigma_i} \cdot (V \prod_{i=1}^{k} H_i^{\sigma_i})^r f^{r_1}.$$  

For $k = 1$ to $N$, we compute

$$K_{3,k} = \left( g^{\sum_{\mathcal{W}(v^*_i)^k} \alpha^{L+1-i} v_{\mu,i}^*} \cdot \left( \prod_{\mathcal{W}(v^*_i)^k} g^{u_i - \alpha^{L+1-i}} \cdot \prod_{\mathcal{W}(v^*_i)^k} (g^{u_i})^{\sigma_i} \cdot \frac{-\alpha^{L+1-i} v_{\mu,i}^*}{\Delta_k} \right) f^{r_1} + \hat{r}^k \right).$$

Other elements in the key can also be simulated:

$$K_1 = g^r = (g^{\alpha^k})^{-1/\Delta_k} \cdot g^r, K_2 = g^{r_1}.$$  

110
• **Challenge**: $A$ sends message $M_0, M_1$ to $B$, then sets using Viete formulas

$$a_{\tau-k} = (-1)^k \sum_{i_1 < i_2 < \ldots < i_k \leq \tau} j_{i_1} j_{i_2} \ldots j_{i_k}, \ 0 \leq k \leq \tau.$$ 

Let $t = a_0$. It creates ciphertext as:

$$C_0 = M_b \cdot T \cdot e(g^\alpha, h)^\gamma, C_1 = h^{1/t}, \ C_2 = h^\psi,$$

$$C_3 = \left( (h^{y + \sum_{i=1}^{L} u_i v_{\mu,i}^{*}})^{\prod (i-j_k)}) \right)^{\frac{1}{t}}$$

If $T = e(g, h)^{\alpha^{L+1}}$, the challenge ciphertext is a valid encryption of $M_b$. On the other hand, when $T$ is uniformly distributed in $G_T$, the challenge ciphertext is independent of $b$.

• **Query Phase 2**: Same Phase 1.

• **Guess**: $A$ output $b' \in \{0, 1\}$. If $b' = b$ then $B$ outputs 1, otherwise outputs 0.

If $b' = 0$, then the simulation is the same as in the real game. Hence, $A$ will have the probability $\frac{1}{2} + \epsilon$ to guess $b$ correctly. If $b' = 1$, then $T$ is random in $G$, then $A$ will have probability $\frac{1}{2}$ to guess $b$ correctly. Therefore, $B$ can solve the decision $L$-BDHE assumption also with advantage $\epsilon$.

### 6.3 Comparison

We give a detailed comparison among all the HVE schemes in Table 6.1. The schemes are compared in terms of the order of the underlying group, ciphertext size, decryption cost, and security assumption. In the table, $p$ denotes the pairing operation, $L$ the length of the vector, and $N$ denotes the maximum number of wildcards.
### Table 6.1: Performance Comparison

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Group Order</th>
<th>Ciphertext Size</th>
<th>Decryption Cost</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Katz et al. [KSW08a]</td>
<td>pqr</td>
<td>$(2L + 1)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Shi–Waters [SW08]</td>
<td>pqr</td>
<td>$(L + 3)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Iovino–Persiano [IP08]</td>
<td>$p$</td>
<td>$(2L + 1)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Sedghi et al. [SLN10]</td>
<td>$p$</td>
<td>$(N + 3)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Lee–Dong [LL11]</td>
<td>pqr</td>
<td>$(L + 2)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Park [Par11]</td>
<td>$p$</td>
<td>$(2L + 3)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>Hattori et al. [HHI11]</td>
<td>pq</td>
<td>$(2L + 3)</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$L – cDBDH$</td>
</tr>
<tr>
<td>CP-HVE1</td>
<td>pq</td>
<td>$4</td>
<td>G</td>
<td>+ 1</td>
</tr>
<tr>
<td>CP-HVE2</td>
<td>$p$</td>
<td>$3</td>
<td>G</td>
<td>+ 1</td>
</tr>
</tbody>
</table>

### 6.4 Summary

We proposed two efficient ciphertext policy Hidden Vector Encryption schemes in this chapter. Both of our encryption schemes can achieve constant ciphertext size, which forms the major contribution of this work. We proved the security of our schemes in a selective security model which captures both plaintext and attribute hiding properties.
Chapter 7

Edit Distance Based Encryption

Measuring the similarity between two strings is an important task in many applications such as natural language processing, bio-informatics, and data mining. One of the common similarity metrics that has been widely used in the above applications is the Edit Distance (a.k.a. Levenshtein distance), which counts the minimum number of operations (namely, insertion, deletion, and substitution) required to transform one string into the other. In this chapter, we investigate a challenging problem of building fuzzy public key encryption schemes based on edit distance.

Our work is motivated by an open problem raised by Sahai and Waters in [SW05b], where the notion of Fuzzy Identity-Based Encryption (IBE) was proposed. The Fuzzy IBE scheme introduced in [SW05b] can be regarded as the first Attribute-Based Encryption (ABE) scheme with a threshold access policy. To be more precise, it allows to use a private key corresponding to an identity string $I'$ to decrypt a ciphertext encrypted with another identity string $I$ if and only if the “set overlap” between $I$ and $I'$ (i.e., $|I \cap I'|$) is larger than a pre-defined threshold. One of the open problems raised in [SW05b] is to construct fuzzy encryption schemes based on other similarity metrics.

We should note that edit distance is very different from the “set overlap” distance used in Fuzzy IBE. For example, consider the biometric identity application of Fuzzy IBE described in [SW05b], given two strings $I = \text{"ATCG"}$ and $I' = \text{"GACT"}$, we have $|I \cap I'| = 4$ (i.e., the distance is 0). However, the edit distance between $I$ and $I'$ is 3. It is easy to see that the order of the alphabets in those strings will affect the edit distance, but not the set overlap distance. This simple example shows that to a certain extent edit distance provides better accuracy than the set overlap distance in measuring the similarity of two strings. As another example, given an encryption string $I = \text{"admirer"}$ and a threshold distance $d = 1$, for edit distance, we can allow a decryption key associated with $I' = \text{"admirers"}$ to decrypt the mes-
sage; while for set overlap distance, we can have some totally unrelated anagrams of \( I \), such as \( I' = \text{"married"} \), whose corresponding secret key can also decrypt the message. Due to the difference between the two distances (or similarity metrics), we cannot easily extend the technique used in [SW05b] to construct a fuzzy encryption scheme for edit distance. Also, in order to distinguish our fuzzy encryption scheme based on edit distance from the Fuzzy IBE proposed in [SW05b], we name our new encryption scheme Edit Distance based Encryption (or EDE, for short).

**Related Work**

Since the seminal work of Sahai and Waters [SW05b], many Attribute Based Encryption (ABE) schemes with the threshold access structure have been proposed (e.g., [GPSW06, BSW07, HLR10, GZC12]). In [GPSW06], Goyal et al. extended the work of Sahai and Waters to construct more expressive Key-Policy (KP) ABE where the access structure is defined via a tree of threshold gates. Bethencourt et al. [BSW07] proposed the first Ciphertext-Policy (CP) ABE using the same access structure. Under the motivation of reducing the ciphertext size, which is linear in the size of the encryption attribute set in most of the existing ABE schemes, Herranz et al. [HLR10] proposed a constant-size ABE scheme for the threshold access structure, which is essentially the same as the set overlap distance metric used in Fuzzy IBE [SW05b]. In [GZC12], Ge et al. proposed another constant-size ABE scheme with the same threshold access structure but under a relatively weaker assumption.

Another type of fuzzy identity-based encryption is the Wildcarded IBE (or WIBE for short) proposed by Abdalla et al. [ACD06, ABC11, ACP12]. A WIBE allows wildcard symbols to appear in an identity string used in the encryption process, and the wildcard positions will be ignored when measuring the equality of two identity strings. Another notion that is similar to WIBE is the Hidden Vector Encryption (HVE) [KSW08b, IP08, SLN10, Par11, PYS14], which also allows wildcards to appear in either the encryption string or the key generation string. However, both WIBE and HVE are based on the fuzzy equality test between two strings, which is different from the problem we aim to solve in this chapter.

There are also a few works on the privacy-preserving edit distance evaluation between two strings [AKD03, JKS08, RS10, CKL15, WHZ15]. These works mainly focused on finding the edit distance of two (perhaps encrypted) strings in a privacy-preserving manner, and hence is completely different from this work.

**Contributions**

In this chapter, we introduce the notion of Edit Distance based Encryption (EDE), formalize its security, and propose a practical scheme in the standard model.
Edit distance can be measured in polynomial time using different techniques, such as dynamic programming or recursion. However, in an EDE scheme, the two strings $I$ and $I'$ are embedded in the ciphertext $CT$ and the user secret key $SK$, respectively. Hence, the problem becomes how to measure the distance of $I$ and $I'$ using $CT$ and $SK$. We observe that the most important operation in the edit distance algorithms is the equality test between two alphabets $I[x]$ and $I'[y]$. Based on this observation, our proposed EDE scheme uses bilinear map [BF01a] to solve this issue. We illustrate our idea using the following example.

Suppose we have two strings $I = \text{"ATTGA"}$ and $I' = \text{"AGTA"}$. We first encode each alphabet as a group element. Then in the encryption process, we create a randomized vector $\vec{I} = (A^s, T^s, T^s, G^s, A^s)$ using the same random number $s$. Similarly, we create another randomized vector $\vec{I}' = (A'^r, G'^r, T'^r, A'^r)$ in the key generation process. Then we apply bilinear map to conduct equality test between $I$ and $I'$ using the two vectors $\vec{I}$ and $\vec{I}'$ which are included in the ciphertext and the secret key respectively. The crux of the idea is illustrated in Figure 7.1. In order to deal with the threshold problem, we apply the technique of Viète’s formulas [SLN+10] to solve the problem. In the encryption process, we create a vector $\vec{d} = (1, 2, \ldots, d, 0, \ldots, 0)$ for the threshold distance $d$ and embed the vector $\vec{d}$ in the ciphertext. Also, based on the edit distance $d'$ between $I$ and $I'$, we create another vector $\vec{d}' = (1, 2, \ldots, d', *, \ldots, *)$ where $*$ denotes the wildcard (i.e., don’t care) symbol. Then based on $\vec{d}$ and $\vec{d}'$, we ensure that the decryption can be successful if and only if $d' \leq d$. Also, we overcome the issue of malleability by using the composite order group in constructing the EDE scheme. We prove that our proposed scheme is selectively secure under the $L$-composite Decisional Diffie-Hellman ($L$-cDDH) assumption.

We also show an interesting application of our EDE scheme named Fuzzy Broadcast Encryption (FBE), which is very useful in broadcasting networks. An FBE scheme allows the encryptor (i.e., message sender) to specify a set of receiver identities during the encryption process, and a user can decrypt the message if and only
if the *minimum* edit distance between his/her identity and all the identities chosen by the encryptor is below a threshold that is also specified by the encryptor during the encryption process.

## 7.1 Definition

### 7.1.1 Edit Distance Based Encryption Definition

An Edit Distance Based Encryption (EDE) scheme consists of the following four probabilistic polynomial-time algorithms:

- **Setup**$(1^n, \Sigma)$: on input a security parameter $1^n$, an alphabet $\Sigma$, the algorithm outputs a public key $PK$ and a master secret key $MSK$.

- **Encrypt**$(PK, \vec{v}, M, d)$: on input a public key $PK$, a message $M$, a vector $\vec{v} \in \Sigma^n$ and a distance $d$, the algorithm outputs a ciphertext $CT$.

- **KeyGen**$(MSK, \vec{x})$: on input a master secret key $MSK$, a vector $\vec{x} \in \Sigma^m$, the algorithm outputs a decryption key $SK$.

- **Decrypt**$(CT, SK)$: on input a ciphertext $CT$ and a secret key $SK$, the algorithm outputs either a message $M$ if $EditDistance(\vec{v}, \vec{x}) \leq d$, or a special symbol $\bot$.

### 7.1.2 Security Model of EDE scheme

The security model for an EDE scheme is defined via the following game between an adversary $A$ and a challenger $B$.

- **Setup**: The challenger $B$ run **Setup**$(1^n, \Sigma)$ to generate the $PK$ and $MSK$. $PK$ is then passed to $A$.

- **Query Phase 1**: The challenger answers all private key queries for a vector $\vec{\sigma}$ by returning : $sk_\sigma \leftarrow KeyGen(MSK, \vec{\sigma})$.

- **Challenge**: $A$ submits two equal-length messages $M_0$ and $M_1$, a target vector $\vec{v}^* \in \Sigma^n$ and threshold $\tau$ such that $EditDistance(\vec{v}^*, \vec{x}) > \tau$ for any vector $\vec{x}$ that has been queried in Phase 1. The challenger then flips a coin $\beta \leftarrow \{0,1\}$ and computes the challenge ciphertext $C^* \leftarrow Encrypt(PK, \vec{v}^*, M_\beta, \tau)$, which is given to $A$.

- **Query Phase 2**: same as Query Phase 1 except that $EditDistance(\vec{v}^*, \vec{x}) > \tau$ for any vector $\sigma$ queried in this phase.
Chapter 7. Edit Distance Based Encryption

7.2 Construction

7.2.1 Edit Distance based Encryption

In this section, we introduce our EDE scheme, which is based on the Dynamic Programming [Gus97] algorithm for calculating edit distance.

**Setup**($1^λ, Σ$): The setup algorithm first chooses $L = poly(n)$ as the maximum number of length of a word that would appear in the encryption and key generation. It then picks large primes $p, q$, generates bilinear groups $G, G_T$ of composite order $n = pq$, and selects generators $g_p ∈ G_p, g_q ∈ G_q$. After that, generate:

\[
\begin{align*}
&v_0, v'_0, b_0, g, f, ω, h_1, \ldots, h_L, u_1, \ldots, u_L ∈_R G_p, x_1, \ldots, x_L, x'_1, \ldots, x'_L ∈_R \mathbb{Z}_n, \\
v_1 = v'_0^{x_1}, \ldots, v_L = v'_0^{x_L}, v'_1 = (v'_0)^{x_1}, \ldots, v'_L = (v'_0)^{x_L}, b_1 = b'_0^{x_1}, \ldots, b_L = b'_0^{x_L}, \\
R_g, R_f, R_{x_0}, \ldots, R_{x_L}, R_{x'_0}, \ldots, R_{x'_L}, \\
R_{b_0}, \ldots, R_{b_L}, R_{h_1}, \ldots, R_{h_L}, R_{u_1}, \ldots, R_{u_L} ∈ G_q, \\
G = g R_g, F = f R_f, Y = e(g, ω), \\
V_0 = v_0 R_{x_0}, \ldots, V_L = v_L R_{x_L}, V'_0 = v'_0 R_{x'_0}, \ldots, V'_L = v'_L R_{x'_L}, B_0 = b_0 R_{b_0}, \ldots, \\
B_L = b_L R_{b_L}, H_1 = h_1 R_{h_1}, \ldots, H_L = h_L R_{h_L}, U_1 = u_1 R_{u_1}, \ldots, U_L = u_L R_{u_L},
\end{align*}
\]

and set the public key and secret key as:

\[
PK = \{Y, G, F, (V_0, \ldots, V_L), (V'_0, \ldots, V'_L), (B_0, \ldots, B_L), (H_1, \ldots, H_L), (U_1, \ldots, U_L)\},
\]

\[
MSK = \{g, f, ω, (v_0, \ldots, v_L), (v'_0, \ldots, v'_L), (b_0, \ldots, b_L), (h_1, \ldots, h_L), (u_1, \ldots, u_L)\}.
\]

**Encrypt**($PK, \overrightarrow{v} = (v_1, \ldots, v_{n_1}) ∈ Σ^{n_1}, M, d$): On input the public key $PK$, a vector $\overrightarrow{v} = (v_1, \ldots, v_{n_1})$ with $n_1 ≤ L$, it first generates for each alphabet $v_i$ a vector $\overrightarrow{x_i} = (v_i, 1, \ldots, 1_L)$, and expands $\overrightarrow{v}$ to $\overrightarrow{v} = (v_1, v_2, \ldots, v_{n_1}, \ldots, 1_L)$ and sets $\overrightarrow{d} = (1, \ldots, d, 0_{d+1}, \ldots, 0_L)$. Then choose $s ∈ R \mathbb{Z}_n$, and $Z_1, Z_2, Z_3, Z_4$, 

• **Output**: $A$ outputs a bit $β'$ as her guess for $β$.

Define the advantage of $A$ as $Adv_{A}^{EDE}(k) = |Pr[β' = β] - 1/2|$. 

**Selective Security.** In the selective security model, the adversary $A$ is required to submit the target vector $\overrightarrow{v^*} ∈ Σ^n$ and threshold $τ$ before the game setup, and $A$ is only allowed to make private key queries for any vector $\overrightarrow{v}$ that satisfies $EditDistance(\overrightarrow{v^*}, \overrightarrow{v}) > τ$ throughout the game.
Chapter 7. Edit Distance Based Encryption

\[ Z_5 \in_R \mathbb{G}_q, \] and compute:

\[ C_0 = MY^s, C_1 = G^s Z_1, C_2 = F^s Z_2, C_{3,i} = (V_i \prod_{j=1}^L H_i^{x_{ij}})^s Z_3, \]

\[ C_4 = (V'_i \prod_{i=1}^L H_i^{y_{ij}}) \cdot Z_4, C_{5,k,t} = (V'_i (B_k \prod_{i=1}^L (U_i)^{d_{ik}})(\prod_{j=1}^L (H_j)^{y_{ij}}))^s \cdot Z_5. \]

Set the ciphertext as: \( CT = (n_1, C_0, C_1, C_2, \{C_{3,i}\}_{i=1}^n, C_4, \{C_{5,k,t}\}_{k=0}^L)_{t=0}. \)

**KeyGen** \((MSK, \bar{\mathcal{Z}} = (z_1, \ldots, z_m) \in \Sigma^m)\): Given a key vector \( \bar{\mathcal{Z}} = (z_1, \ldots, z_m) \), it generates \( \bar{y}_i = (z_i, 1, 1, \ldots, 1_L) \) for each alphabet \( z_i \), and creates \( \mathcal{Z} = (1, 2, \ldots, L) \) and expands \( \bar{\mathcal{Z}} \) to \( \mathcal{Z} = (z_1, z_2, \ldots, z_m, 1_L) \). Then choose \( r_1, r_2 \in_R Z_n \), and compute:

\[
K_1 = g^{r_1}, K_2 = g^{r_2}, K_{3,i} = (v_i \prod_{j=1}^L h_i^{y_{ij}})^{r_2},
\]

\[
\begin{pmatrix}
K_{4,0,0} = \omega(v'_0 \prod_{i=1}^L h_i^{z_{i0}})^{r_2} \\
(b_0 \prod_{i=1}^L (u_i^{\sigma_i}))(v'_0 \prod_{j=1}^L h_j^{z_{j0}})^{r_1} \cdot f^{r_1}
\end{pmatrix} = \begin{pmatrix}
K_{4,0,t} = (b_0 \prod_{i=1}^L (u_i^{\sigma_i}))(v'_t \prod_{j=1}^L h_j^{z_{jt}})^{r_1} \\
(b_1 \prod_{i=1}^L (u_i^{\sigma_i}))(v'_0 \prod_{j=1}^L h_j^{z_{j0}})^{r_1} \\
\vdots \\
(b_L \prod_{i=1}^L (u_i^{\sigma_i}))(v'_t \prod_{j=1}^L h_j^{z_{jt}})^{r_1}
\end{pmatrix},
\]

\( (t = 1, \ldots, L). \)

Then set the user secret key as \( SK = (m, K_1, K_2, \{K_{3,i}\}_{i=1}^m, \{K_{4,k,t}\}_{k=0}^L)_{t=0}. \)

**Decrypt** \((CT, SK)\): The decryption algorithm first executes the dynamic programming algorithm for edit distance by following **Algorithm 1** which returns a distance \( d' = cost(len_{\mathcal{X}} - 1) \), the matching indices array \( pos[0][] \) for \( \bar{\mathcal{Z}} \) and \( pos[1][] \) for \( \mathcal{Z} \). It sets \( \tau = L - d' \), and applies the Viète’s formulas to compute

- for the index set \( \Omega_x = \{L \setminus \{pos[0][0], \ldots, pos[0][d' - 1]\}\} = \{\omega_1, \ldots, \omega_{L - d'}\} \), then \( a_{\tau - k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq \tau} \omega_{i_1} \omega_{i_2} \ldots \omega_{i_k} \quad (0 \leq k \leq \tau), \)
- for the index set \( \Omega_z = \{L \setminus \{pos[1][0], \ldots, pos[1][d' - 1]\}\} = \{\bar{\omega}_1, \ldots, \bar{\omega}_{L - d'}\} \), then \( \bar{a}_{\tau - k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq \tau} \bar{\omega}_{i_1} \bar{\omega}_{i_2} \ldots \bar{\omega}_{i_k} \quad (0 \leq k \leq \tau), \)
- for the threshold index set \( J = \{j_1, \ldots, j_r\} \) with \( j_1 = d' + 1, \ldots, j_r = L \), then

118
\[ \hat{a}_{\tau-k} = (-1)^k \sum_{1 \leq i_1 < i_2 < \ldots < i_k \leq \tau} j_{i_1} j_{i_2} \ldots j_{i_k} \quad (0 \leq k \leq \tau). \]

**Algorithm 1:** Edit distance evaluation via dynamic programming

**input:** $CT, SK$

**output:** $d', pos$

\[
\text{len}_v = n + 1; \text{len}_z = m + 1; \\
\text{Creat cost}[\text{len}_v]; \text{Creat newcost}[\text{len}_v]; \text{Creat pos[2][2]}; \\
// setup two arrays to store the position matching pos[0][i] for vector v, pos[1][j] for vector z
\]

for $i \leftarrow 0$ to $\text{len}_v$ do
  \[\text{cost}[i] = i;\]
end

$k = 0;$

for $j \leftarrow 1$ to $\text{len}_z$ do
  \[\text{newcost}[0] = j;\]
  for $i \leftarrow 1$ to $\text{len}_v$ do
    // matching current letters in both strings
    \[\text{match} = (e(K_2, C_{3,i-1}) == e(C_1, K_{3,j-1})); 0 : 1;\]
    // store the $i$ match in array pos[0], $j$ match in array pos[1]
    if $i \notin \text{pos}[0], j \notin \text{pos}[1]$ then
      \[\text{pos}[0][k + +] = i, \text{pos}[1][k + +] = j;\]
    end
    // computing cost for each transformation
    \[\text{replace} = \text{cost}[i - 1] + \text{match}; \text{insert} = \text{cost}[i] + 1; \text{delete} = \text{newcost}[i - 1] + 1;\]
    \[\text{keep minimum cost}\]
    \[\text{newcost}[i] = \text{Math.min} (\text{Math.min} (\text{cost} \ldots \text{insert}, \text{cost} \ldots \text{delete}); \text{cost} \ldots \text{replace});\]
  end
  // swap cost-newcost arrays
  \[\text{swap}[] = \text{cost}; \text{cost} = \text{newcost}; \text{newcost} = \text{swap};\]
end
// return the cost for transforming all letters in both strings and array list
\[
\text{pos including pos[0], pos[1]}
\]
\[\text{return cost}[\text{len}_v - 1], \text{pos};\]

Then recover $M$ as:

\[
M = \frac{e(K_2, C_4) \cdot e(K_1, C_2) \prod_{k=0}^r e(K_{1 \slash a_0}^{1+a_0}, \prod_{t=0}^r C_{5,k,t}^{a_t})^\hat{a}_k}{\prod_{k=0}^r e(\prod_{t=0}^r K_{4,k,t}^{a_t}, C_{1 \slash a_0}^{1+a_0})^\hat{a}_k} \cdot C_0.
\]

**Correctness.**

In Algorithm 1:
We then illustrate an example:

**Input:** “AAGTA”, “AAAGG”

**Output:**
- \( d' = 2 \)
- \( pos = < pos[0] >, pos[1] > \), with \( pos[0] = \{1, 2\} \), \( pos[1] = \{1, 2\} \)

In message recovery:

\[
e(C_0) = M \cdot e(g, \omega)^s
\]

\[
e(K_2, C_4) = e(g^{r_2}, (V_i^t \prod_{i=1}^L H_i^{x_{ij}})^s \cdot Z_4)
\]

\[
e(g, v_i)^{sr_2} e(g, \prod_{i=1}^L (h_i^{v_{ij}})^{sr_2})
\]

\[
e(K_1, C_2) = e(g^{r_1}, F^s Z_2) = e(g, f)^{sr_1}
\]

\[
\prod_{k=0}^r e(K_1^{-\tau \sum_{i=0} a_{i_{k}}}, \prod_{t=0}^\tau C_{5,t,k,l}^{a_{i_{k}}})^{\hat{a}_{i_{k}}}
\]

\[
= e(g, v_0^{\tau}) \prod_{t=0}^\tau \prod_{k=0}^r e(g, b_0) \prod_{i=1}^L e(g, h_j) e(v_{j}^{r_2} \prod_{t=0}^\tau \prod_{k=0}^r \hat{a}_{i_{k}})
\]

\[
\prod_{k=0}^r e(K_1^{a_{i_{k}}}, C_1^{-\tau \sum_{i=0} a_{i_{k}}})^{\hat{a}_{i_{k}}}
\]

\[
e(\omega^{a_{i_{0}} a_{i_{0}}}, (V_i^t \prod_{i=1}^L (h_i^{z_{ij}}))^r_2 a_{i_{0}} \prod_{k=1}^r \prod_{i=1}^L ((v_i^t b_k \prod_{i=1}^L (u_i)^i)^k)
\]

\[
\prod_{j=1}^L (h_j^{z_{ij}})^{r_1 a_{i_{0}}} f^{r_1 a_{i_{0}}}, G^{a_{i_{0}} Z_4}
\]
Asymptotic Security of Z,T

Theorem 12 Assume that the Decisional L−cBDHE assumption holds, then for any PPT adversary, our EDE scheme is selectively secure.

Let B denote the algorithm to solve the Decisional L−cBDHE problem. B is given a challenge instance Z,T' of the problem, where Z = (g_p, g_q, h, g_p^a, ..., g_p^L, g_p^{α+2}, ..., g_p^{α+L}) and T' is either T = e(g_p, h)^{α+1} or R ∈ R \mathbb{G}_{T,p}. B simulates the game for A as follows:

- **Init**: A submits a target vector \( \vec{v}^* \in \Sigma^n \), and target threshold \( τ \). Let \( \vec{d} = (1, \ldots, τ, τ + 1, \ldots, L) \) denote a vector of length \( L \). We denote \( \text{ind}(\vec{d}) = \{1 \leq i \leq L | d_i = 0\} \) and \( \text{ind}(\vec{d})^0 = \{1 \leq i \leq L | d_i \neq 0\} \), and \( \text{ind}(\vec{d})\}_i^0 \) as \( \{i \in \text{ind}(\vec{d}) | 0 \leq i \leq φ\} \).

- **Setup**: In this phase, B generates:

\[
\gamma, \psi, v_0, v_0', b_0, g, f, h, u_1', \ldots, u_L', v_1, \ldots, v_L, v_1', \ldots, v_L' = (v_1')^x, \ldots, v_L' = (v_L')^x, b_1 = v_0^x, \ldots, b_L = v_L^x, R_g, R_f, R_{u_0}, \ldots, R_{u_L}, R_{v_0}, \ldots, R_{v_L}, R_{b_0}, \ldots, R_{b_L}, R_{u_1'}, \ldots, R_{u_L'}, R_{v_1'}, \ldots, R_{v_L'}, \in R \mathbb{G}_q,
\]

\[
G = g_p R_y, F = g_p^\psi R_f, Y = e(g_p^a, g_p^{α} g_p^\gamma),
\]

\[
V_t = g_p^{\psi x_t} R_{v_t}, V'_t = g_p^{\psi x_t} R_{v'_t}, \text{ with } t = 1, \ldots, L,
\]

\[
B_k = g_p^{\psi x_k} \prod_{i \in \text{ind}(\vec{d})} g_p^{α+1-i d_i} R_{b_k}, \text{ with } k = 1, \ldots, L,
\]

\[
H_i = g_p^{k_i} R_{h_i}, \{U_i = g_p^{u_i-α+1-i} R_{u_i}\}_{i \in \text{ind}(\vec{d})}, \{U_i = g_p^{u'_i} R_{u'_i}\}_{i \in \text{ind}(\vec{d})}
\]

The corresponding master secret key components are: \( g = g_p, f = g_p^\psi, h_i = g_p^{k_i}, \{u_i = g_p^{u_i-α+1-i}\}_{i \in \text{ind}(\vec{d})}, \{u_i = g_p^{u'_i}\}_{i \in \text{ind}(\vec{d})}, v_t = v_0^x, v_t' = v_0', \) with \( t = 1, \ldots, L, b_k = b_0^x \prod_{i \in \text{ind}(\vec{d})} g_p^{α+1-i d_i}, \text{ with } k = 1, \ldots, L. \) Notice that the master key component \( ω \) is \( g_p^{αL+1+α}\gamma . \) Since B does not have \( g_p^{αL+1} , B \)
cannot compute \( \omega \) directly.

- **Query Phase 1:** A queries the user secret key for a string \( \overrightarrow{z} = (z_1, z_2, \ldots, z_m) \) under the constraint that \( \text{EditDistance}(\overrightarrow{v}^*, \overrightarrow{z}) > \tau \). Assume \( \text{EditDistance}(\overrightarrow{v}^*, \overrightarrow{z}) = \sigma \) and denote \( \overrightarrow{d} = (1, 2, \ldots, \sigma, 0, \ldots, 0) \) and \( \overrightarrow{d} = (1, 2, \ldots, \tau, 0, \ldots, 0) \).

  Note that since \( \sigma > \tau \), there exists at least one position \( i \) such that \( d_i = 0 \) and \( \sigma_i \neq 0 \). Let \( \phi \in \text{ind}(\overrightarrow{d}) \) be the smallest integer such that \( \sigma_\phi \neq d_\phi \).

  B simulates the user key generation process as follows:

  \[
  K_{4,0,0} = \omega(v'_0 \prod_{i=1}^{L} h_{i}^{(\overrightarrow{w}_i)}) r_2 (b_0 \prod_{i=1}^{L} (u'_i)^{\sigma_i}) r_1 (v'_0 \prod_{i=1}^{L} h_{i}^{(\overrightarrow{w}_i)}) r_1 f r_1 \\
  = g_p^{\alpha L+1+\alpha\gamma}(v'_0 \prod_{i=1}^{L} h_{i}^{(\overrightarrow{w}_i)}) r_2 (g_p^{\phi} \prod_{i \in \text{ind}(\overrightarrow{d})} g_p^{\alpha L+1-i d_i} \prod_{i \in \text{ind}(\overrightarrow{d})} (g_p^{u'_i-\alpha L+1-i}) \sigma_i \\
  \cdot \prod_{i \in \text{ind}(\overrightarrow{d})} (g_p^{u'_i}) r_1 (v'_0 \prod_{j=1}^{L} h_j^{(\overrightarrow{w}_j)}) r_1 f r_1 \\
  = g_p^{\alpha L+1+\alpha\gamma}(v'_0 \prod_{i=1}^{L} u_i^{(\overrightarrow{w}_i)}) r_2 (g_p^{X} r_1 (v'_0 \prod_{j=1}^{L} h_j^{(\overrightarrow{w}_j)}) r_1 f r_1
  \]

where \( X = \sum_{\text{ind}(\overrightarrow{d})} \alpha^{\alpha L+1-i d_i} + b_0 + \sum_{\text{ind}(\overrightarrow{d})} (u'_i-\alpha^{L+1-i}) \sigma_i + \sum_{\text{ind}(\overrightarrow{d})} u'_i \sigma_i \).

  Since \( \sum_{\text{ind}(\overrightarrow{d})} (u'_i-\alpha^{L+1-i}) \sigma_i + \sum_{\text{ind}(\overrightarrow{d})} u'_i \sigma_i = -\sum_{\text{ind}(\overrightarrow{d})} \alpha^{L+1-i} \sigma_i + \sum_{i=1}^{L} u'_i \sigma_i \), and recall \( \sigma_i = d_i \) for \( i \in \text{ind}(\overrightarrow{d}) \) and \( \sigma_\phi \neq d_\phi \). Hence, we have:

  \[
  X = \sum_{\text{ind}(\overrightarrow{d})} \alpha^{L+1-i} (d_i - \sigma_i) + \sum_{i=1}^{L} u'_i \sigma_i + b_0 = \alpha^{L+1-\phi} \Delta_\phi + \sum_{i=1}^{L} u'_i \sigma_i + y
  \]

where \( \Delta_\phi = (d_\phi - \sigma_\phi) \). Then we choose \( r_1, r_2 \) randomly in \( \mathbb{Z}_n \), and set \( r_1 = \frac{-\alpha^{\phi}}{\alpha^{\phi}} + r_1', r_2 = r_2' \).

Then \( K_{4,0,0} \) can be represented as:

\[
K_{4,0,0} = g_p^{\alpha L+1+\alpha\gamma}(v'_0 \prod_{i=1}^{L} (h_i)^{(\overrightarrow{w}_i)}) r_2 \cdot (g_p^{\alpha^{L+1-\phi} \Delta_\phi + \sum_{i=1}^{L} u'_i \sigma_i}) \frac{-\alpha^{\phi}}{\alpha^{\phi}} + r_1' \\
= g_p^{\alpha L+1+\alpha\gamma}(v'_0 \prod_{i=1}^{L} h_i^{(\overrightarrow{w}_i)}) r_2 \cdot g_p^{\alpha^{L+1-\phi} \Delta_\phi} (g_p^{\phi} \sum_{i=1}^{L} u'_i \sigma_i) \frac{-\alpha^{\phi}}{\alpha^{\phi}} + r_1' \\
= (v'_0 \prod_{j=1}^{L} h_j^{(\overrightarrow{w}_j)}) \frac{-\alpha^{\phi}}{\alpha^{\phi}} + r_1' f \frac{-\alpha^{\phi}}{\alpha^{\phi}} + r_1'
\]
Then we simulate $T_{4,k,t}$ with $k, t \neq 0$ as:

$$T_{4,k,t} = ((b_k \prod_{i=1}^{L} (w_i^t \sigma_i ))(v_i' \prod_{j=1}^{L} h_j^d))_{r_1}$$

$$= (g_p^{b_k} \prod_{\phi \in \text{ind}(d)} g_p^{t \phi d_{\phi}} \prod_{\phi \in \text{ind}(\sigma)} (g^{u_{\phi}^t} g^d_{\phi}^{u_{\phi}^t a^{t+1}})^{\sigma_{\phi}} \prod_{\phi \in \text{ind}(\sigma)} (g^{u_{\phi}^t} g^d_{\phi}^{u_{\phi}^t a^{t+1}})^{\sigma_{\phi}} + r_1)$$

$$\cdot (v_i' \prod_{j=1}^{L} h_j^d)_{r_1} \cdot (v_i' \prod_{j=1}^{L} h_j^d)_{r_1}$$

Next, it generates for each alphabet in $\vec{z}$:

$$\begin{cases}
\vec{y}_1 = (z_1, 1, \ldots, 1_L) \\
\vec{y}_m = (z_m, 1, \ldots, 1_L)
\end{cases}$$

computes $K_{3,i} = (v_i \prod_{j=1}^{L} h_j^d)^{r_2}$. Other elements in the key can also be simulated: $K_1 = g^{r_1} = g^{\frac{\alpha g}{\phi} + r_1}, K_2 = g^{r_2}$.

- **Challenge**: $A$ sends two messages $M_0, M_1$ to $B$. The challenger then flips a coin $\beta \leftarrow \{0, 1\}$.

First, $B$ generates for each alphabet in $\vec{v}^{*}$:

$$\begin{cases}
\vec{x}_1^{*} = (v_1, 1, \ldots, 1_L) \\
\vec{x}_m^{*} = (v_m, 1, \ldots, 1_L)
\end{cases}$$

generates $Z_1, Z_2, Z_3, Z_4, Z_5 \xleftarrow{\$} \mathbb{G}_q$ and sets:

$$C_0 = M_b \cdot T' \cdot e(g_0^c, h)^\gamma, C_1 = hZ_1, C_2 = h^vZ_2, C_{3,i} = h^{v_{x_i} + \sum_{j=1}^{L} h_j^d z_{i,j}^*} Z_3,$$

$$C_4 = h^{v_{i} + \sum_{i=1}^{L} h_i^v z_{i}^*} Z_4, C_{5,k,t} = h^{b_k \sum_{i=1}^{L} u_i^d h_i^d + \sum_{j=1}^{L} v_j^d h_j^d} Z_5$$

where $h = g_p^c$ for some unknown $c \in \mathbb{Z}_p$. $B$ returns the challenge ciphertext

$$CT^{*} = (n_1, C_1, C_2, \{C_{3,i}\}_{i=1}^{n_1}, C_4, \{\{C_{5,k,t}\}_{k=0}^{L} \}_{l=0}^{L})$$
to $A$. If $T' = T = e(g_p, h)^{\alpha + 1}$, then:

$$C_0 = M_b \cdot e(g_p, g_p^{\alpha + 1}) \cdot e(g_p^\alpha, g_p^\gamma) = M_b \cdot e(g_p, g_p^{\alpha + 1})^c \cdot e(g_p^\alpha, g_p^\gamma)^c = M_b \cdot Y^c$$

$$C_1 = (g_p^\gamma) \cdot Z_1 = G^c \cdot Z_1^c, C_2 = (g_p^\gamma)^\psi \cdot Z_2 = F^c \cdot Z_2^c,$$

$$C_{3,i} = (g_p^\gamma)^{v_0 x_i + \sum_{j=1}^L h'_i x_{i,j}^*} \cdot Z_3 = (g_p^{v_0 x_i + \sum_{j=1}^L h'_i x_{i,j}^*})^c \cdot Z_3 = (V_i \prod_{j=1}^L \bar{H}_i^{x_{i,j}})^c \cdot Z_3^c,$$

$$C_4 = (g_p^{v_0^* + \sum_{j=1}^L h'_i v_{i,j}^*}) \cdot Z_4 = ((g_p^{v_0^* + \sum_{j=1}^L h'_i v_{i,j}^*})^c \cdot Z_4 = (V_0' \prod_{i=1}^L \bar{H}_i^{v_{i,j}^*})^c \cdot Z_4^c.$$

$$C_{5,k,t} = (g_p^b) \cdot Z_5 = (g_p^b)^{\sum_{i=1}^L (U_i)^{d_i k} + \sum_{j=1}^L v_{i,j}^* h'_i j^t} \cdot Z_5 = (V_t' \prod_{i=1}^L \bar{H}_j^{v_{i,j}^*})^c \cdot Z_5^c.$$

The challenge ciphertext is a valid encryption of $M_b$. On the other hand, when $T'$ is uniformly distributed in $\mathbb{G}_{T,p}$, the challenge ciphertext is independent of $b$.

- **Query Phase 2**: Same as Phase 1.

- **Guess**: $A$ output $b' \in \{0, 1\}$. If $b' = b$ then $B$ outputs 1; otherwise outputs 0.

If $b' = 0$, then the simulation is the same as in the real game. Hence, $A$ will have the probability $\frac{1}{2} + \epsilon$ to guess $b$ correctly. If $b' = 1$, then $T'$ is random in $\mathbb{G}_{T,p}$, then $A$ will have probability $\frac{1}{2}$ to guess $b$ correctly. Therefore, $B$ can solve the Decisional $L-c$BDHE assumption also with advantage $\epsilon$. \hfill \Box

### 7.4 EDE Application-Fuzzy Broadcast Encryption

We demonstrate an extension of the proposed EDE scheme to achieve Fuzzy Broadcast Encryption. To illustrate how the scheme works, let’s consider the following example. Suppose we encrypt a message under a keyword vector $w = \{\text{Labour Party, Defence Unit}\}$ and a threshold distance $d = 2$. Subsequently, people who have the attributes related to the keyword $w = \text{Labour Party}$ or $w' = \text{Defence Unit}$ can decrypt the message since the minimum edit distance between $w$ ($w'$, respectively) and all the keywords in $W$ is 1, which is less than the threshold $d = 2$. 

124
7.4.1 Definition

A Fuzzy Broadcast Encryption (FBE) scheme consists of the following four probabilistic polynomial-time algorithms:

- **Setup**($1^\lambda, \Sigma$): on input a security parameter $1^n$, an alphabet $\Sigma$, the algorithm outputs a public key $PK$ and the corresponding master secret key $MSK$.

- **Encrypt**($PK, M, W = (w_{1,l_1}, w_{2,l_2}, \ldots, w_{k,l_k}) \in \Sigma^{n_1}, d$): on input a public key $PK$, a list of $k$ keywords $W = (w_{1,l_1}, w_{2,l_2}, \ldots, w_{k,l_k})$ in which each keyword $w_{i,l_i}$ has $l_i$ characters, and a threshold distance $d$, the algorithm outputs a ciphertext $CT$.

- **Key Gen**($MSK, w \in \Sigma^m$): on input the master secret key $MSK$ and a keyword $w$ of length $m$, the algorithm outputs a secret key $SK_w$.

- **Decrypt**($CT, SK_w$): on input a ciphertext $CT$ with keywords $W = (w_{1,l_1}, w_{2,l_2}, \ldots, w_{k,l_k})$ and a secret key $SK_w$ with keyword $w$, the algorithm outputs $M$ if $\operatorname{Min}\{\text{EditDistance}(w_{i,l_i}, w)\}^{k}_{i=1} \leq d$, or $\bot$ otherwise.

7.4.2 Construction

Below we present a FBE scheme based on our EDE scheme.

**Setup**($1^\lambda, \Sigma$): The setup algorithm is generated similar to the original EDE scheme.

**Encrypt**($PK, W = (w_{1,l_1}, w_{2,l_2}, \ldots, w_{k',l_{k'}}), M, d$): On input the public key $PK$, a list of $k$ keywords $W = (w_{1,l_1}, w_{2,l_2}, \ldots, w_{k',l_{k'}})$ in which each keyword $w_{i,l_i}$ has $l_i$ alphabets, it first generates for each alphabet $w_{i,j}$ in keyword $w_{i,l_i}$ a vector

\[
\begin{align*}
x_{11}^i &= (w_{11,1}, \ldots, 1_L), \ldots, x_{1l_i}^i = (w_{1l_i,1}, \ldots, 1_L), \\
\vdots \\
x_{k1}^{k'} &= (w_{k'1,1}, \ldots, 1_L), \ldots, x_{k'l_{k'}}^{k'} = (w_{k'l_{k'},1}, \ldots, 1_L).
\end{align*}
\]

Define

\[
\begin{align*}
\vec{w}_1 &= (w_{11}, w_{1,2}, \ldots, w_{1,l_i}, \ldots, 1_L), \\
\vdots \\
\vec{w}_{k'} &= (w_{k'1}, w_{k',2}, \ldots, w_{k',l_{k'}}, \ldots, 1_L),
\end{align*}
\]

and $\vec{d} = (1, \ldots, d, 0_{d+1}, \ldots, 0_L)$. Then choose $s \in \mathbb{Z}_m$, and $Z_1, Z_2, Z_3, Z_4$, 

125
Chapter 7. Edit Distance Based Encryption

\[Z_5 \in R \mathbb{G}_q, \text{ and compute:}\]

\[C_0 = MY^a, C_1 = G^z Z_1, C_2 = F^z Z_2, C_{3,\delta,i} = (V_i \prod_{j=1}^L H_i^{w_{ij}})^s Z_3,\]

\[C_{4,\delta} = (V_0' \prod_{i=1}^L H_i^{w_{ij}})^s \cdot Z_4, C_{5,k,\delta,i} = (V'_i (B_k \prod_{i=1}^L (U_i)^{d_{ik}} \prod_{j=1}^L (H_j)^{w_{ij} r^j}))^s \cdot Z_5.\]

Set the ciphertext as: \(CT = \{(l_\delta)^{k'}_{\delta=1}, C_0, C_1, C_2, \{C_{3,\delta,i}^{k'}\}_{\delta=1}^{l_i}, \{C_{4,\delta}^{k'}\},\{\{C_{5,k,\delta,i}\}_{l=0}^L\}_{\delta=1}^{k'}\).\)

**KeyGen** \((MSK, \bar{w} = (\bar{w}_1, \ldots, \bar{w}_m) \in \Sigma^m)\): given a keyword \(\bar{w}\) of length \(m\), it generates \(\bar{y}_i = (\bar{w}_i, 1, \ldots, L)\) for each alphabet \(\bar{w}_i\), and creates \(\bar{\sigma} = (1, 2, \ldots, L)\) and expands \(\bar{w}\) to \(\bar{w} = (\bar{w}_{1z}, \bar{w}_{2z}, \ldots, \bar{w}_{mz}, \ldots, 1_L)\). Then choose \(r_1, r_2 \in R Z_n\), and compute:

\[K_1 = g^{r_1}, K_2 = g^{r_2}, K_{3,i} = (v_i \prod_{j=1}^L h_i^{w_{ij}})^{r_2},\]

\[
\begin{align*}
K_{4,0,0} &= \omega(v_0' \prod_{i=1}^L h_i^{w_{ij}})^{r_2} \\
K_{4,1,0} &= (b_0 \prod_{i=1}^L (u_i^{r_1}))(v_0' \prod_{j=1}^L h_j^{w_{ij}})^{r_1} f^{r_1} \\
K_{4,i,0} &= (b_1 \prod_{i=1}^L (u_i^{r_1}))(v_0' \prod_{j=1}^L h_j^{w_{ij}})^{r_1} \\
\cdots \\
K_{4,L,0} &= (b_L \prod_{i=1}^L (u_i^{r_1}))(v_0' \prod_{j=1}^L h_j^{w_{ij}})^{r_1}
\end{align*}
\]

, for \(t = 1, \ldots, L\). Then set the secret key as \(SK = (m, K_1, K_2, \{K_{3,i}\}_{i=1}^m, \{\{K_{4,k,i}\}_{k=0}^L\}_{i=0}^L)\).

**Decrypt** \((CT, SK)\): The decryption algorithm first executes the dynamic programming algorithm for edit distance by following **Algorithm 2** which returns a minimum distance \(d'\), the index \(pos_w\) of the corresponding keyword \(w\) in \(W\), the matching indices array \(pos[0][]\) for \(w\) and \(pos[1][]\) for \(\bar{w}\). It sets \(\tau = L - d'\), and applies the Viète’s formulas to compute

We then set \(a_{r-k}, \bar{a}_{r-k}, \bar{a}_{r-k}\) similar to (3), (4), (5).

Then recover \(M\) as:

\[M = \frac{e(K_2, C_{4,pos_w}) \cdot e(K_1, C_2) e(K_{1}^{\frac{1}{a_{t}}}, \prod_{t=0}^{i} \prod_{C_{5,k,pos_w,t}}^{a_{t}})}{e(\prod_{t=0}^{i} K_{4,k,t}^{\frac{1}{a_{t}}}, C_{1}^{\frac{1}{a_{t}}})} \cdot C_0.\]
Algorithm 2: Multi-keyword Edit Distance Evaluation via Dynamic Programming

**input**: CT, SK

**output**: distance $d'$, index $pos_w$, array $pos[2][]$

Create Array[len(W)]; Create pos[2][]; Create Array <$pos > aPos;

for $\theta \leftarrow 1$ to len(W) do

\[ len_v = n_\theta + 1; len_z = m + 1; \]

Create cost[len_v]; Create newcost[len_v];

for $i \leftarrow 0$ to len_v do

\[ cost[i] = i; \]

end

$k = 0$;

for $j \leftarrow 1$ to len_z do

\[ newcost[0] = j; \]

for $i \leftarrow 1$ to len_v do

match = ($e(K_1, C_{3,\theta,i-1}) == e(C_1, K_{3,j-1})) \? 0 : 1$

if $i \notin pos[0], j \notin pos[1]$ then

\[ pos[0][k++] = i, pos[1][k++] = j; \]

end

aPos.add(pos);

\[ cost - replace = cost[i - 1] + match; cost - insert = cost[i] + 1; cost - delete = newcost[i - 1] + 1; \]

\[ newcost[i] = Math.min(Math.min(cost - insert, cost - delete), cost - replace); \]

end

\[ swap[] = cost; cost = newcost; newcost = swap; \]

end

Array[t ++] = cost[len_v - 1]; Refresh pos;

end

return Min(Array[]), $pos_w = index[Array[i] == Min(Array[])], pos = aPos[pos_w];$

**Theorem 13** Assume that the Decisional L–cBDHE assumption holds, then for any PPT adversary, our FBE scheme is selectively secure.

We give the security definition in the full version since the limited space. The security proof follows that of **Theorem 1**.

### 7.5 Summary

We introduced a new type of fuzzy public key encryption in this chapter. Our new encryption scheme, called Edit Distance-based Encryption (EDE), allows a user associated with an identity or attribute string to decrypt a ciphertext encrypted under another string if and only if the edit distance between the two strings are
within a threshold specified by the encrypter. We provide the formal definition, security model, and a concrete EDE scheme in the standard model. We also showed an extension of our EDE scheme for fuzzy broadcast encryption.
Chapter 8

Concluding Remark

8.1 Summary of our contributions

We briefly summarize our main contributions as each following subsections.

8.1.1 Ciphertext Policy Attribute Based Encryption

We propose two new Ciphertext Policy Attribute Based Encryption (CP-ABE) schemes where the access policy is defined by AND-Gate with wildcard. In the first scheme, we present a new technique which uses only one group element to represent an attribute, while the existing ABE schemes of the same type need to use three different group elements to represent an attribute for the three possible values (namely positive, negative, and wildcard). Our new technique leads to a new CP-ABE scheme with constant ciphertext size, which however cannot hide the access policy used for encryption. The second scheme of this work is to propose a new CP-ABE scheme with the property of hidden access policy named Hidden Ciphertext Policy-ABE by extending the technique we used in the construction of our first scheme.

8.1.2 Attribute Based Broadcast Encryption

We proposed two ABBE schemes including KP-ABBE and CP-ABBE schemes. We achieve short ciphertext size for CP-ABBE and short key size for KP-ABBE, which forms the major contribution of this work.

8.1.3 Anonymous Attribute Based Broadcast Encryption

We presented two new constructions of Attribute Based Broadcast Encryption for the OR/AND Gate and AND Gate with wildcard access policies. We solved the
interesting problem of providing the anonymity in ABBE scheme.

8.1.4 Efficient Ciphertext Hidden Vector Encryption

We introduce two constant size ciphertext policy hidden vector encryption (CP-HVE) schemes. Our first scheme is constructed on composite order bilinear groups, while the second one is built on bilinear groups with prime order. Both schemes are proven secure in a selective security model which captures plaintext (or payload) and attribute hiding.

8.1.5 Edit Distance Based Encryption

We introduce a new type of fuzzy public key encryption called Edit Distance-based Encryption (EDE). In EDE, the encryptor can specify an alphabet string and a threshold when encrypting a message, and a decryptor can obtain a decryption key generated from another alphabet string, and the decryption will be successful if and only if the edit distance between the two strings is within the pre-defined threshold. We provide a formal definition and security model for EDE, and propose an EDE scheme that can securely evaluate the edit distance between two strings embedded in the ciphertext and the secret key. We also show an interesting application of our EDE scheme named Fuzzy Broadcast Encryption which is very useful in a broadcasting network.

8.2 Future Work

- For our proposed Hidden Ciphertext Policy-ABE, one shortcoming is that its ciphertext size is no longer constant. It is an interesting future work to construct a hidden ciphertext policy-ABE with constant ciphertext.

- The ABBE scheme which achieves the short ciphertext and decryption key can be extended to achieve adaptive security.

- Another interesting problem is the construction of an anonymous EDE scheme, which implies a Fuzzy Public-key Encryption with Keyword Search scheme to preserve the privacy of the keyword.


[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Proceedings of the 4th conference on Theory


[DPP07] Cécile Deferabréllé, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size cipher-


