Predicate Encryption with Various Properties

by

Dongdong Sun

Thesis submitted in accordance with the regulations for the
Degree of Doctor of Philosophy

Information Security Institute
Faculty of Science and Engineering
Queensland University of Technology

December 14, 2012
Predicate encryption (PE) is a new primitive which supports flexible control over access to encrypted data. In PE schemes, users’ decryption keys are associated with predicates $f$ and ciphertexts encode attributes $a$ that are specified during the encryption procedure. A user can successfully decrypt if and only if $f(a) = 1$. In this thesis, we will investigate several properties that are crucial to PE. We focus on expressiveness of PE, Revocable PE and Hierarchical PE (HPE) with forward security. For all proposed systems, we provide a security model and analysis using the widely accepted computational complexity approach.

Our first contribution is to explore the expressiveness of PE. Existing PE supports a wide class of predicates such as conjunctions of equality, comparison and subset queries, disjunctions of equality queries, and more generally, arbitrary combinations of conjunctive and disjunctive equality queries. We advance PE to evaluate more expressive predicates, e.g., disjunctive comparison or disjunctive subset queries. Such expressiveness is achieved at the cost of computational and space overhead. To improve the performance, we appropriately revise the PE to reduce the computational and space cost. Furthermore, we propose a heuristic method to reduce disjunctions in the predicates. Our schemes are proved in the standard model.

We then introduce the concept of Revocable Predicate Encryption (RPE), which extends the previous PE setting with revocation support: private keys can be used to decrypt an RPE ciphertext only if they match the decryption policy (defined via attributes encoded into the ciphertext and predicates associated with private keys) and were not revoked by the time the ciphertext was created. We propose two RPE schemes. Our first scheme, termed Attribute-Hiding RPE (AH-RPE), offers attribute-hiding, which is the standard PE property. Our second scheme, termed Full-Hiding RPE (FH-RPE), offers even stronger privacy guarantees, i.e., apart from possessing the Attribute-Hiding property, the scheme also ensures that no information about revoked users is leaked from a given ciphertext. The proposed schemes are also proved to be secure under well established assumptions in the standard model.

Secrecy of decryption keys is an important pre-requisite for security of (H)PE and compromised private keys must be immediately replaced. The notion of Forward Security (FS) reduces damage from compromised keys by guaranteeing confidentiality of messages that were encrypted prior to the compromise event. We present the first Forward-Secure Hierarchical Predicate Encryption (FS-HPE) that is proved secure in the standard model. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates (to support dynamic behavior for delegation of decrypting rights to new users), local update for users’ private keys (i.e., no master authority needs to be contacted), forward security, and the scheme’s
encryption process does not require knowledge of predicates at any level including when those predicates join the hierarchy.
## Contents

**Front Matter**
- Abstract ................................................................................. i
- Table of Contents ................................................................. iii
- Declaration ............................................................................. vii
- Previously Published Material ............................................... ix
- Acknowledgements ............................................................... xi

**1 Introduction**
- 1.1 Background ........................................................................ 1
- 1.2 Research Objectives .......................................................... 4
- 1.3 Contributions ..................................................................... 4
- 1.4 Overview of Chapters ......................................................... 5

**2 Cryptographic Background**
- 2.1 Cryptographic Preliminaries ............................................... 7
  - 2.1.1 Number Theory .......................................................... 7
  - 2.1.2 Bilinear Map ............................................................. 8
  - 2.1.3 Dual Pairing Vector Spaces ........................................ 9
  - 2.1.4 Computational Assumptions ....................................... 10
- 2.2 Predicate Encryption ......................................................... 11
  - 2.2.1 Syntax ..................................................................... 11
  - 2.2.2 Security Definition ................................................... 11
- 2.3 Hierarchical Predicate Encryption .................................... 12
  - 2.3.1 Syntax ..................................................................... 12
  - 2.3.2 Security Definition ................................................... 13
- 2.4 Inner Product Encryption and its Applications .................. 14
  - 2.4.1 Anonymous Identity-Based Encryption ....................... 14
  - 2.4.2 Hidden-Vector Encryption ......................................... 14
  - 2.4.3 PE Supporting Polynomial Evaluation ....................... 15
  - 2.4.4 Evaluating Disjunctions and Conjunctions .................. 16
- 2.5 Provable Security ............................................................. 16
  - 2.5.1 Proof Techniques ...................................................... 17
3 Predicate Encryption for Multi Inner Product

3.1 Introduction ................................................................. 19
  3.1.1 Our Results .......................................................... 20
3.2 Complexity Assumptions ................................................. 21
3.3 Security Model ............................................................. 22
3.4 The Intuition of Our Design .............................................. 24
3.5 Scheme 1 ................................................................. 25
  3.5.1 Security Proof of Scheme 1 .......................................... 26
3.6 Scheme 2 ................................................................. 27
  3.6.1 Security Proof of Scheme 2 .......................................... 28
3.7 Applications of Our Schemes ........................................... 35
  3.7.1 The System Evaluating Disjunctive Comparison Predicates .......... 35
  3.7.2 The System Evaluating Disjunctive Subset Predicates ................. 37
3.8 Arbitrary Combinations of Conjunctive and Disjunctive Predicates ....... 38
3.9 Achieving Adaptive and CCA security .................................. 40
3.10 Conclusion ............................................................... 40

4 Revocable Predicate Encryption with Attribute Hiding

4.1 Introduction ............................................................... 41
  4.1.1 Contributions of this chapter .................................... 43
4.2 Security Model ............................................................ 43
  4.2.1 Syntax ............................................................... 44
  4.2.2 Security Definition ................................................ 44
4.3 Our Scheme .............................................................. 45
4.4 Proof of Security ....................................................... 47
4.5 Conclusion ............................................................... 67

5 Revocable Predicate Encryption with Full Hiding

5.1 Introduction ............................................................... 69
  5.1.1 Contributions of this chapter .................................... 70
5.2 Security Model ............................................................ 71
  5.2.1 Security Definition ................................................ 71
5.3 Our Scheme .............................................................. 71
5.4 Proof of Security ....................................................... 74
5.5 Conclusion ............................................................... 93

6 A Forward-Secure Hierarchical Predicate Encryption Scheme

6.1 Introduction ............................................................... 95
  6.1.1 Contributions of this chapter .................................... 97
  6.1.2 Initial Attempts, or Why FS-HPE is Challenging? ................. 98
6.2 Security Model ............................................................ 99
  6.2.1 Notation ............................................................. 100
  6.2.2 Syntax ............................................................... 100
  6.2.3 Security Definition ................................................ 101
Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: ..................................................  Date: ..............................
Publication


The papers 2, 3 and 4 contain material based on the content of this thesis.
I would like to thank my principal supervisor Dr. Juan Manuel González Nieto and associate supervisor Prof. Colin Boyd, for their constant support and encouragement. With their thoughtful guidance, I was able to enter into this new challenging research field. Special thanks go to Prof. Mark Manulis, for his help in improving a few drafts of my papers. I am also grateful to my parents, for their great love and spiritual supports. Last, but not least, I would like to express my deepest thanks to my wife Na Meng and my son Daniel Sun. Without their love, this thesis would not have been possible.
Chapter 1

Introduction

An intelligence agency may want to send a classified message to some agents. However, the privacy is a major concern, i.e., the agency only wants the intended agent to read the message. A possible solution is to use public key encryption. An agent generates a public key/private key pair, and publishes the public key $PK$. The agency can encrypt a message with $PK$ and send it to the agent. The agent can recover the message with the private key.

Now the agency may instead want to encode some attributes in the encrypted message. For example, a message might be associated with attributes (level 3, military). Each agent has a key for some policy. The key can be used to decrypt the ciphertext if the attributes in the ciphertext satisfy the policy in the key. Alice has a key associated with the policy (Classification $> 2$) and (Area $\in \{\text{economy, military}\}$), which means that Alice can decrypt the ciphertext with classification level greater than 2 and in the area of economy or military. Similarly, Bob may have a key for policy (Classification $= 1$) and (Area $\in \{\text{economy, politics}\}$). It can be seen that Alice can decrypt the ciphertext encrypted with attributes (level 3, military), but Bob cannot decrypt it. Moreover, not only the message but also the attributes are highly classified. Exposure of the attributes in the ciphertext may result in potential threats. Both of them should be encrypted in the ciphertext.

This problem can be addressed with a newly emerged encryption called predicate encryption (PE). Using PE, the agency can create a ciphertext by encrypting a message as well as attributes. An agent can decrypt the ciphertext if the attributes in the ciphertext satisfy the policy in the key. In this thesis, we will focus on PE and explore several new properties of this primitive.

1.1 Background

Public-Key Encryption and Identity-Based Encryption: In a public-key encryption, each user has a public key $e$ and a corresponding private key $d$, where $d$ is computationally infeasible to find given $e$. To create a ciphertext, the message is encrypted using $e$. On the
other hand, \( d \) can be used to decrypt the ciphertext. This cryptosystem is realized by public key infrastructure (PKI), where there is a certificate authority (CA) which generates a digital certificate to bind an entity with its public key. Despite PKI’s merits, the system has not been used as quickly as desired. The major reason is that PKI contains complex certificate construction, which involves heavy computation and management costs.

To cope with the situation, identity-based cryptography was introduced \[12, 53\]. In identity-based systems, a public key can be derived from a widely known identity, such as an email address or phone number. User’s private key can be generated by a trusted authority called private key generator (PKG) with the help of the master secret key. There is no need to bind a digital certificate and a user’s public key, since the public key is widely known. This mechanism reduces the costs of key management and certificate verification as much as possible. To create a ciphertext, the message is encrypted with a user’s identity \( \text{ID} \). Meanwhile, the key for \( \text{ID}' \) can be used to decrypt the ciphertext if \( \text{ID}' = \text{ID} \). Because of the efficiency of the system, ID-based cryptography is now flourishing in the research community.

**Attribute-Based Encryption**: Identity-based encryption (IBE) eliminates the certificate overheads by making the recipient’s public key derivable from their related identities, such as email addresses or phone numbers. This is sufficient for point to point communication. However, ID-based systems may not be applicable in all applications, where a ciphertext may be accessed by many users or one user may decrypt many ciphertexts. Attribute-based encryption (ABE) provides a way to address this issue. In 2005, Sahai et al. \[50\] proposed the first attribute-based encryption scheme. Generally speaking, in an ABE system, a user’s private keys are associated with some access structure and ciphertexts are labeled with sets of descriptive attributes. A user can decrypt a particular ciphertext only if the labeled attributes satisfy the access structure. We refer to this system as key-policy ABE. The seminal work of Sahai et al. \[50\] partly addresses the problem of fine-grained access control over encrypted data. The system allows for decryption when at least \( k \) attributes overlap between a ciphertext and a private key (also called Fuzzy IBE). The system possesses the property of collusion resistance and the security proof is under the Selective-ID model. However, the system only supports threshold access policies, which limits its applicability to larger systems due to the lack of expressibility. Goyal et al. \[28\] proposed another attribute-based encryption which supports a much richer type of access policies. The construction is based on a kind of tree-access structure. The authors also presented a key delegation capability for the system, which is similar to that in the hierarchical identity-based encryption \[16\]. One of the disadvantages is that the system is only constrained in monotonic access structure. The issue was later addressed by Ostrovsky et al. \[47\]. They proposed a key-policy ABE scheme which supports any access formula over attributes, including non-monotonic ones. To realize the desired property, a linear secret-sharing scheme is applied as an underlying tool. This work can be regarded as a significant improvement in ABE systems. Later on, Chase \[20\] gave an efficient construction for multi-authority attribute based encryption based on the work by Sahai et al. \[50\]. In this construction, a sender specifies for each authority a set of attributes produced by that authority and a number \( d \). The ciphertext can be decrypted by a user who has at least \( d \) attributes from each authority. As an extension, a large universe access control structure ABE scheme \[28\] in
multi-authority settings was also presented.

In ciphertext-policy ABE, the private key is associated with attributes and the ciphertext is associated with the access structure. In 2007, Bethencourt et al. [6] proposed the first ciphertext-policy attribute-based encryption (ABE) system. Their work is based on Goyal’s scheme [28]. They gave a novel construction which prevents collusion attacks. The tree-access structure is applied in the scheme, which achieves more expressive access control. They applied the similar technique used by Sahai et al. [50] to achieve the collusion-resistant property. They also gave a security proof under generic group model. However, it is desirable to find a scheme which is provably secure in a more solid model. This issue is addressed by Waters [60]. Waters presented a new ciphertext-policy ABE scheme using a general set of access structures in the standard model under concrete and non-interactive assumptions. He gives three schemes that allow different tradeoffs between the system’s efficiency and the complexity of the assumptions used. The schemes support more expressive access structures than the work by Bethencourt et al. [6] and Goyal et al. [28]. Cheung [21] proposed another ciphertext-policy ABE scheme. The scheme achieves AND gate on positive and negative attributes as an access policy on the ciphertext, which is the conjunctive normal form policy. A variant with smaller ciphertexts and faster encryption/decryption was also proposed.

Predicate Encryption: In this thesis we focus on the notion of Predicate Encryption (PE), formalized by Katz, Sahai, and Waters [34], and further studied in [34, 35, 42, 43, 54–56]. PE can be viewed as a generalization of ID-based and Attribute-based encryption. In PE schemes the private keys of users are associated with predicates \( f \) and ciphertexts are bound to attributes \( a \). The decryption procedure is successful if and only if \( f(a) = 1 \). If this relation is not satisfied then no information about the plaintext is leaked. In contrast to Attribute-Based Encryption, which also states this requirement on the security of the decryption procedure, PE schemes offer privacy of attributes that legitimate recipients of PE ciphertexts must possess, that is PE ciphertexts ensure attribute hiding in that they do not leak any information about \( a \) for which the condition \( f(a) = 1 \) would be satisfied. Concrete constructions of PE schemes typically focus on the realization of certain types of predicates \( f \). In their seminal work, Katz, Sahai, and Waters [34] introduced PE schemes supporting Inner-Product Encryption (IPE), i.e. vector \( \vec{y} \) represents attributes and vector \( \vec{x} \) determines the predicate \( f_{\vec{x}} \) such that \( f_{\vec{x}}(\vec{y}) = 1 \) iff \( \vec{x} \cdot \vec{y} = 0 \) (\( \vec{x} \cdot \vec{y} \) denotes the inner product of vectors \( \vec{x} \) and \( \vec{y} \) over a field or ring). It has been shown that IPE can be leveraged to evaluate a wide class of predicates such as conjunctions or disjunctions of equality tests, conjunctions of comparison or subset tests, and more generally, arbitrary conjunctive normal form (CNF) or disjunctive normal form (DNF) formulae. The original scheme of Katz, Sahai, and Waters [34] has been proved selectively secure under less standard assumptions (in the generic group model). Recent result of Lewko et al. [35] provided more sophisticated PE constructions achieving (stronger) adaptive security under non-standard assumptions. Furthermore, Okamoto and Takashima [43] investigated Functional Encryption that is adaptively secure under standard assumptions. In [35, 42] the authors also explored constructions of Hierarchical PE (HPE) schemes providing their users with the ability to delegate their decryption keys down the hierarchy by restricting predicates associated to the delegated keys and by this restricting the abilities of lower-level users to
decrypt.

1.2 Research Objectives

This thesis focuses on exploring different properties of (H)PE schemes. We aim to achieve the following objectives:

1. As mentioned in the previous section, the existing PE supports a wide class of predicates\cite{34}. However, there are still some predicates that cannot be evaluated by previous PE. For example, disjunctive comparison or disjunctive subset queries are not supported by the existing schemes. Since there is practical interest for those predicates, we believe that it is worthwhile to investigate PE for such predicates. The first aim of this thesis is to design a PE that can be used to evaluate not only the existing supported predicates but also the aforementioned predicates, i.e., disjunctive comparison or disjunctive subset predicates. We also aim to reduce the overhead incurred from the new system and prove the security under some security model.

2. Efficient revocation is a key requirement for many cryptographic systems. There is no exception for PE. We note that current revocation techniques in IBE\cite{12, 53} or ABE\cite{6, 20, 28, 50} systems, aside from their scalability issues, are only partially applicable to PE schemes due to the distinguished attribute-hiding property of the latter. To our knowledge, revocation in PE schemes has not been investigated so far and it is not clear whether revocation introduces further privacy challenges, in addition to the challenge of preserving their basic attribute-hiding property. We aim to propose the first PE with revocation. Our new scheme should preserve the attribute hiding. We will also consider the efficiency of the system, i.e., the new scheme should be practical.

3. Forward Security (FS) offers meaningful protection in cryptographic applications with long-term private keys in the unfortunate case when these keys become compromised. A message encrypted with an (H)PE scheme can potentially be decrypted by many users. Compromise of some users’ private keys in these schemes will cause damage since the adversary could obtain messages that were encrypted for those users. Adding forward security to (H)PE schemes is thus desirable to mitigate this issue. We note that there exists no forward secure HPE scheme (FS-HPE) in the literature. We aim to formalize and design the first FS-HPE that also naturally implies an FS-PE (1-level FS-HPE). We also want to provide security proof for the proposed system.

1.3 Contributions

The goal of this thesis is twofold: (1) constructing PEs with various properties, and (2) providing security models and analyzing the security of the proposed schemes.

The main contributions of this thesis are as follows:
1. We propose PE schemes supporting multi inner product, which in turn can be used to evaluate a wide class of predicates, including disjunctive comparison or disjunctive subset queries.

2. We present two revocable PE schemes. The first scheme is a revocable PE with attribute hiding while the second one is a revocable PE possessing full hiding property.

3. We propose the first forward secure (H)PE scheme which also implies a FS-PE scheme.

4. We provide security models for our schemes and prove the schemes to be secure in the standard model.

1.4 Overview of Chapters

This thesis is organized in the following way:

In Chapter 2, we review the cryptographic preliminaries on which the rest of the thesis is based. Firstly, we introduce some preliminaries, including primitives in number theory, bilinear map, dual pairing vector spaces and computational assumptions. We also provide the security models of PE and HPE. Next, we show the applicability of PE for inner product predicate. Finally, we review the concept of provable security.

Chapter 3 is devoted to the study of PE for multi inner product (PEM). We define the multi inner product predicate. We then provide the syntax of PEM and its security model. We also present two PEM schemes that can be used to evaluate multi inner product predicate and prove the schemes to be secure in the standard model. We show how to leverage the proposed PEM to realize the system supporting disjunctions of comparison and disjunctions of subset queries. Furthermore, we extend our technique for disjunctive queries to achieve arbitrary combinations of conjunctive and disjunctive equality, comparison and subset predicate evaluations. To improve the system performance, we provide a heuristic method to reduce disjunctions in the predicates.

In Chapter 4, we introduce the concept of Revocable Predicate Encryption (RPE), which extends the previous PE setting with revocation support: private keys can be used to decrypt an RPE ciphertext only if they match the decryption policy (defined via attributes encoded into the ciphertext and predicates associated with private keys) and were not revoked by the time the ciphertext was created. We formalize the notion of attribute hiding in the presence of revocation and propose our first RPE construction, called AH-RPE, which possesses attribute hiding property in the ciphertext. The proposed scheme is proved to be secure in the standard model.

In Chapter 5, we further investigate RPE and present another scheme, termed RPE with full hiding (FH-RPE), which achieves stronger security, i.e., apart from the attribute hiding, the scheme ensures that no information about revoked users is leaked from a given ciphertext. Security model and proof are also provided.

In Chapter 6, we formalize and design the first forward secure hierarchical predicate encryption (FS-HPE) scheme. We first present a new syntax and security definitions that are specific to FS-HPE; in particular, the definition of attribute hiding has to be extended in order to
account for FS, in a more complex way than in FS-HIBE definitions. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates (to support dynamic behavior for delegation of decrypting rights to new users), local update for users’ private keys (i.e., no master authority needs to be contacted), forward security, and the scheme’s encryption process does not require knowledge of predicates at any level including when those predicates join the hierarchy. We then analyze the security of proposed FS-HPE in the standard model.

In Chapter 7, we provide a summary of this thesis. We also present some open problems and discuss future research directions.
Chapter 2

Cryptographic Background

In this chapter, we will provide a brief review of cryptographic background, which is necessary for understanding the rest of the thesis. We first introduce some preliminaries, including number theory, bilinear map, dual pairing vector spaces and computational assumptions. Security models of PE and HPE will be discussed. We also show how the PE for inner product can be leveraged to achieve a wide class of predicates. Finally, we provide the concept of provable security. We aim to provide a self-contained cryptographic background in this chapter so that readers with little knowledge in this field will be able to understand the ideas and arguments presented in the thesis.

2.1 Cryptographic Preliminaries

2.1.1 Number Theory

Firstly, we review some definitions from number theory which forms the foundation of modern cryptography.

Definition 2.1. A non-negative integer \( d \) is the greatest common divisor of integers \( a \) and \( b \), denoted \( d = \gcd(a, b) \), if

(1) \( d \) is a common divisor of \( a \) and \( b \); and

(2) whenever \( c \mid a \) and \( c \mid b \), then \( c \mid d \).

Definition 2.2. Two integers \( a \) and \( b \) are said to be relatively prime or coprime if \( \gcd(a, b) = 1 \).

Definition 2.3. For \( n \geq 1 \), let \( \phi(n) \) denote the number of integers in the interval \([1, n]\) which are relatively prime to \( n \). The function is called the Euler phi function (or the Euler totient function).

Definition 2.4. If \( a \) and \( b \) are integers, then \( a \) is said to be congruent to \( b \) modulo \( n \), written \( a \equiv b \mod n \), if \( n \) divides \( a - b \). The integer \( n \) is called the modulus of the congruence.
Definition 2.5. The integers modulo $n$, denoted $\mathbb{Z}_n$, is the set of (equivalence classes of) integers $\{0, 1, 2, \ldots, n - 1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo $n$.

Definition 2.6. Let $a \in \mathbb{Z}_n$. The multiplicative inverse of $a$ modulo $n$ is an integer $x \in \mathbb{Z}_n$ such that $ax \equiv 1 \mod n$. If such an $x$ exists, then it is unique, and $a$ is said to be invertible, or a unit; the inverse of $a$ is denoted by $a^{-1}$.

Definition 2.7. Let $a, b \in \mathbb{Z}_n$. Division of $a$ by $b$ modulo $n$ is the product of $a$ and $b^{-1}$ modulo $n$, and is only defined if $b$ is invertible modulo $n$.

Definition 2.8. The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n|\gcd(a,n) = 1\}$. In particular, if $n$ is a prime, then $\mathbb{Z}_n^* = \{a|1 \leq a \leq n - 1\}$.

Definition 2.9. The order of $\mathbb{Z}_n^*$ is defined to be the number of elements in $\mathbb{Z}_n^*$, written $|\mathbb{Z}_n^*|$.

Definition 2.10. Let $a \in \mathbb{Z}_n^*$. The order of $a$, denoted $\text{ord}(a)$, is the least positive integer $t$ such that $a^t \equiv 1 \mod n$.

Definition 2.11. Let $\alpha \in \mathbb{Z}_n^*$. If the order of $\alpha$ is $\phi(n)$, then $\alpha$ is said to be a generator or a primitive element of $\mathbb{Z}_n^*$. If $\mathbb{Z}_n^*$ has a generator, then $\mathbb{Z}_n^*$ is said to be cyclic.

More inquisitive audience can refer the book by Menezes et al. [39] for further details.

2.1.2 Bilinear Map

A bilinear map (or, the so-called pairing) is a cryptographic tool which has been extensively investigated in many applications [11, 13, 16, 26, 50, 59]. In 2001, Boneh and Franklin [12] proposed the first practical ID-based encryption with security proof. Their scheme solved the open problem proposed by Shamir [53].

We formally define the concept and properties of bilinear pairing as follows:

Definition 2.12. Let $\lambda$ be a security parameter and $q$ be a $\lambda$-bit prime number. Let $G$ and $G_T$ be groups of the same prime order $q$. A bilinear map $e : G \times G \rightarrow G_T$ has the following properties:

1. **Bilinear**: for all $P, Q \in G$ and $s, t \in \mathbb{Z}_q^*$, $e(sP, tQ) = e(P, Q)^{st}$.

2. **Non-degeneracy**: for any generator $G \in G$, $e(G, G) \neq 1$.

3. **Computability**: there is an efficient algorithm to compute $e(P, Q)$, for $P, Q \in G$.

The bilinear maps also possess additional properties as follows:

1. for all $P_1, P_2, Q \in G$, $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$.

2. for all $P, Q \in G$, $e(P, Q) = e(Q, P)$. 

The group $\mathbb{G}$ is written additively. Typically $\mathbb{G}$ is a subgroup of elements on an elliptic curve. Non-degeneracy implies that if $\mathbb{G}$ generates $\mathbb{G}$, $e(\mathbb{G}, \mathbb{G})$ generates $\mathbb{G}_T$. The map $e$ is called an admissible bilinear map. We can obtain such non-degenerate admissible maps over cyclic groups from the Weil or the Tate pairing over supersingular elliptic curves \cite{12} or abelian varieties \cite{49}. For simplicity, we will use the term bilinear map to refer to the admissible bilinear map denoted above in the rest of this thesis.

Let $\mathcal{G}_{\text{bpg}}$ be an algorithm that takes as input a security parameter $\lambda$ and outputs a description of the bilinear pairing group setting $(q, \mathbb{G}, \mathbb{G}_T, G, e)$ where $q$ is a prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $q$, $G$ is the generator of $\mathbb{G}$, $e$ is a non-degenerate bilinear map defined in Definition \ref{def:bilinear_map}. The generator $\mathcal{G}_{\text{bpg}}$ will be used in the rest of the thesis.

### 2.1.3 Dual Pairing Vector Spaces

#### Vector space $\mathbb{V}$

Let $\mathbb{V} = \mathbb{G} \times \cdots \times \mathbb{G}$ and each element in $\mathbb{V}$ is expressed by $N$-dimensional vector. $x = (x_1G, \ldots, x_NG)$ ($x_i \in \mathbb{F}_q$ for $i = 1, \ldots, N$).

#### Canonical basis $\mathcal{A}$

Let $\mathcal{A} = (a_1, \ldots, a_N)$ of $\mathbb{V}$, where $a_1 = (G, 0, \ldots, 0)$, $a_2 = (0, G, 0, \ldots, 0)$, 

\ldots, $a_N = (0, \ldots, 0, G)$.

#### Pairing operation: $e(x, y) = \prod_{i=1}^N e(x_iG, y_iG) = e(G, G)^{\sum_{i=1}^N x_iy_i} = g_T^{\overrightarrow{v}} \in \mathbb{G}_T$, where $x = (x_1G, \ldots, x_NG) = x_1a_1 + \cdots + x_Na_N \in \mathbb{V}$, $y = (y_1G, \ldots, y_NG) = y_1a_1 + \cdots + y_Na_N \in \mathbb{V}$, $\overrightarrow{v} = (x_1, \ldots, x_N)$ and $\overrightarrow{y} = (y_1, \ldots, y_N)$.

#### Basis change:

Let basis $\mathbb{B} = (b_1, \ldots, b_N)$ of $\mathbb{V}$, which is obtained from canonical basis $\mathcal{A}$ using a uniformly at random chosen linear transformation, $\Lambda = (\lambda_{i,j}) \overset{\mathcal{U}}{\leftarrow} \text{GL}(N, \mathbb{F}_q)$, such that $b_i = \sum_{j=1}^N \lambda_{i,j}a_j$, for $i = 1, \ldots, N$. Similarly, $\mathbb{B}^* = (b_1^*, \ldots, b_N^*)$ of $\mathbb{V}$ is also obtained from $\mathcal{A}$, such that $\mu_{i,j} = (\Lambda^T)^{-1}$. $b_i^* = \sum_{j=1}^N \mu_{i,j}a_j$, for $i = 1, \ldots, N$. It can be shown that $e(b_i, b_j^*) = g_T^{\delta_{i,j}}$, where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$. $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthonormal bases of $\mathbb{V}$.

#### Hard problem and trapdoor:

The hard problem due to this approach is the decisional subspace problem (DSP) \cite{11}. The DSP assumption is that it is hard to distinguish $v = v_{N_2}b_{N_2+1} + \cdots + v_Nb_N$ from $u = v_1b_1 + \cdots + v_{N_1}b_{N_1}$, where $(v_1, \ldots, v_{N_1}) \overset{\mathcal{U}}{\leftarrow} \mathbb{F}_q^{N_1}$ and $N_2 + 1 < N_1$. Moreover, there exists a trapdoor $t^* \in \text{span}(b_1^*, \ldots, b_{N_2}^*)$ which can be used to solve the DSP problem. $\text{span}(b_1^*, \ldots, b_{N_2}^*)$ denotes the subspace generated by $b_1^*, \ldots, b_{N_2}^*$. Given $v = v_{N_2+1}b_{N_2+1} + \cdots + v_Nb_N$ or $u = v_1b_1 + \cdots + v_{N_1}b_{N_1}$, we can tell $v$ from $u$ using $t^*$ because $e(v, t^*) = 1$ and $e(u, t^*) \neq 1$ with high probability. The hard problem is not directly used in the proof of our schemes, but it gives intuition to propose the high level assumptions in the construction of our schemes. We will show those assumptions later.

### Advantage of using dual orthonormal bases $\mathbb{B}$ and $\mathbb{B}^*$ of $\mathbb{V}$

In a typical vector treatment, the pairing for two vectors $R = (x_1G, \ldots, x_NG)$ and $S = (y_1G, \ldots, y_NG)$
is computed as $e(R, S) = \prod_{i=1}^{N} e(x_iG, y_iG)$. In the approach using canonical basis $A$, $R$ and $S$ can be rephrased as $R = x_1a_1 + \cdots + x_Na_N$ and $S = y_1a_1 + \cdots + y_Na_N$ respectively. However, the drawback of this approach is that it is trivial to decompose $x_ia_i = (0, \ldots, 0, x_iG, 0, \ldots, 0)$ from $R = x_1a_1 + \cdots + x_Na_N = (x_1G, \ldots, x_NG)$, which means the decisional subspace problem can be easily solved. In contrast, the current approach employs basis $B$ which is linearly transformed from $A$ using a secret random matrix $\Lambda \in \mathbb{F}_q^{N \times N}$. It seems hard to decompose $x_ib_i$ from $R' = x_1b_1 + \cdots + x_Nb_N$. Hence, the decisional subspace problem is hard to solve. Moreover, we can view the secret matrix $\Lambda$ and the dual orthonormal basis $B^*$ of $V$ as the trapdoors to the decisional subspace problem. DSP can be solved using the pairing operation over $B$ and $B^*$ as mentioned above.

**Definition 2.13 (Dual Pairing Vector Space (DPVS) [12])**. Let $(q, G, G_T, G, e)$ be a symmetric pairing group. A Dual Pairing Vector Space $(q, V, G_T, A, e)$, generated by an algorithm denoted $G_{dpvs}$, is a tuple of a prime $q$, $N$-dimensional vector space $V$ over $\mathbb{F}_q$, a cyclic group $G_T$ of order $q$, canonical basis $A = (a_1, \ldots, a_N)$ of $V$, and pairing $e : G \times G \to G_T$ that satisfy the following conditions:

1. **Non-degenerate bilinear pairing**: There exists a polynomial-time computable non-degenerate bilinear pairing $e(x, y) = \prod_{i=1}^{N} e(G_i, H_i)$ where $x = (G_1, \ldots, G_N) \in V$ and $y = (H_1, \ldots, H_N) \in V$. This is non-degenerate bilinear i.e., $e(sx, ty) = e(x, y)^{st}$ and if $e(x, y) \neq 1$ for all $y \in V$, then $x = 0$.

2. **Dual orthonormal bases**: $A$ and $e$ satisfy that $e(a_i, a_j) = g_T^{\delta_{i,j}}$ for all $i$ and $j$, where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise, and $g_T \neq 1 \in G_T$.

3. **Distortion maps**: Linear transformations $\phi_{i,j}$ on $V$ s.t. $\phi_{i,j}(a_j) = a_i$, and $\phi_{i,j}(a_k) = 0$ if $k \neq j$ are polynomial-time computable. We call $\phi_{i,j}$ distortion maps.

### 2.1.4 Computational Assumptions

In this section, we will introduce the computational assumption that will be used to prove the security of our proposed schemes.

**Definition 2.14 (Decisional Linear Assumption (DLIN) [10])**. The DLIN problem is to decide on bit $\beta \in \{0, 1\}$, given $(\text{param}_G, G, aG, bG, acG, bdG, Y_\beta) \in G_{\beta}^{\text{DLIN}}(1^\lambda)$, where the algorithm $G_{\beta}^{\text{DLIN}}(1^\lambda)$ is as follows:

$$\text{param}_G = (q, G, G_T, G, e) \xleftarrow{\text{R}} G_{\text{bug}}(1^\lambda), a, b, c, d \xleftarrow{\text{R}} \mathbb{F}_q, Y_0 = (c+d)G, Y_1 \xleftarrow{\text{R}} G, \beta \xleftarrow{\text{R}} \{0, 1\}.$$


The advantage of a probabilistic polynomial-time DLIN solver $D$ is defined as follows:

$$\text{Adv}_D^{\text{DLIN}}(\lambda) = \left| \Pr[D(1^\lambda, \varnothing) \rightarrow 1 \mid \varnothing \xleftarrow{\text{R}} G_0^{\text{DLIN}}(1^\lambda)] - \Pr[D(1^\lambda, \varnothing) \rightarrow 1 \mid \varnothing \xleftarrow{\text{R}} G_1^{\text{DLIN}}(1^\lambda)] \right|.$$

The DLIN assumption states that for any probabilistic polynomial-time solver $D$, the advantage $\text{Adv}_D^{\text{DLIN}}(\lambda)$ is negligible in $\lambda$. 

As explained by Boneh et al. [10], it can be shown that an algorithm for solving Decision Linear in $G$ provides an algorithm for solving DDH in $G$. The converse is believed to be false, i.e., Decision Linear is a hard problem even in bilinear groups where DDH is easy.

**Remark 2.1.** The schemes in Chapter 4, 5 and 6 are proven under the DLIN problem, while the security of the scheme in Chapter 3 relies on some non-standard assumptions, we put those assumptions in Chapter 3 for clarity.

### 2.2 Predicate Encryption

#### 2.2.1 Syntax

**Definition 2.15.** A Predicate Encryption (PE) consists of four algorithms (Setup, GenKey, Encrypt, Decrypt) and has associated attribute space $\mathcal{A}$ and predicate space $\mathcal{P}$.

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

- **GenKey($MSK, f$)** The GenKey algorithm takes as input a master secret key $MSK$ and a predicate $f \in \mathcal{P}$. It outputs a secret key $SK_f$.

- **Encrypt($PK, I, M$)** The Encrypt algorithm takes as input a public key $PK$, an attribute $I \in \mathcal{A}$, and a message $M$ in some associated message space. It outputs a ciphertext $C$.

- **Decrypt($C, SK_f$)** The Decrypt algorithm takes as input a ciphertext $C$ and a secret key $SK_f$. It outputs either a message $M$ or the distinguished symbol $\perp$.

**Correctness.** The correctness property of the schemes says that for all $\lambda$, all $(PK, MSK)$ output by Setup($1^\lambda$) algorithm, all predicate $f \in \mathcal{P}$, any key $SK_f$ generated by GenKey($MSK, f$) algorithm, and all attributes $I \in \mathcal{A}$,

- If $f(I) = 1$ then $\text{Decrypt}(\text{Encrypt}(PK, I, M), SK_f) = M$.

- If $f(I) = 0$ then $\text{Decrypt}(\text{Encrypt}(PK, I, M), SK_f) = \perp$ with all but negligible probability.

#### 2.2.2 Security Definition

**Definition 2.16.** A Predicate Encryption Scheme is adaptively attribute hiding against chosen plaintext attacks if for all PPT adversaries $A$, the advantage $\text{Adv}_{A}^{\text{PE}}$ of $A$ in the following game is a negligible function of the security parameter $\lambda$:

- **Setup.** A challenger $C$ runs the Setup algorithm to generate public key $PK$ and master secret key $MSK$, and $PK$ is given to $A$.

- **Query phase 1.** $A$ adaptively makes a polynomial number of secret key queries for predicates $f$. In response, $C$ computes the secret key $SK_f$ and reveals it to $A$. 
Chapter 2. Cryptographic Background

Challenge. A outputs challenge attributes \(I_0, I_1 \in A\) and challenge plaintexts \(M_0, M_1\), subject to the restriction that \(f(I_0) = f(I_1)\), and if \(f(I_0) = f(I_1) = 1\), then it is required that \(M_0 = M_1\), for all queried keys \(SK_f\).

C flips a random coin \(b\). If \(b = 0\) then \(A\) is given \(C = \text{Encrypt}(PK, I_0, M_0)\) and if \(b = 1\) then \(A\) is given \(C = \text{Encrypt}(PK, I_1, M_1)\).

Query phase 2. Repeat the Query phase 1 subject to the restrictions as in the challenge phase.

Guess. A outputs a guess \(b'\) of \(b\), and succeeds if \(b' = b\).

The advantage of \(A\) is defined to be \(\text{Adv}_{A}^{PE}(\lambda) = |\Pr[b = b'] - 1/2|\).

2.3 Hierarchical Predicate Encryption

In this section, we define hierarchical predicate encryption (HPE) and its security. In a delegation system, it is required that a user who has a capability can delegate to another user a more restrictive capability.

Let \(\Delta\) be a format of hierarchy that includes the depth \(d\) of the hierarchy and size of attribute/predicate on each level. With \(A_l, l = 1, \ldots, d\) we denote attribute sets. A hierarchical attribute space is defined as \(A = \bigcup_{l=1}^{d} (A_1 \times \cdots \times A_l)\) and a hierarchical predicate space is defined as \(P = \bigcup_{l=1}^{d} (P_1 \times \cdots \times P_l)\). For \(f_i \in P_i\), a hierarchical attribute \(Y = (I_1, \ldots, I_h) \in A\) is said to satisfy a hierarchical predicate \(F = (f_1, \ldots, f_l)\) iff \(l \leq h\) and \(f_i(I_i) = 1\) for \(1 \leq i \leq l\), which we denote as \(F(Y) = 1\). We call \(h\) (resp. \(l\)) the level of \((I_1, \ldots, I_h)\) (resp. \((f_1, \ldots, f_l)\)).

For hierarchical predicates \(F\) and \(F'\) in \(P\), we denote \(F' \leq F\) if the predicate for \(F\) is a prefix of that for \(F'\).

2.3.1 Syntax

Definition 2.17. A Hierarchical Predicate Encryption Scheme (HPE) is a tuple of five algorithms \((\text{Setup}, \text{GenKey}, \text{Encrypt}, \text{Decrypt}, \text{Delegate})\) described in the following:

Setup\((1^\lambda, \Delta)\) This algorithm takes as input a security parameter \(1^\lambda\) and the format of hierarchy \(\Delta\). It outputs public parameters of the system, including a (master) public key \(PK\) and a (master) secret key \(MSK\).

GenKey\((MSK, (f_1, \ldots, f_l))\) This algorithm takes as input a master secret key \(MSK\) and hierarchical predicate \((f_1, \ldots, f_l)\). It outputs corresponding secret key \(SK_{f_1, \ldots, f_l}\).

Encrypt\((PK, (I_1, \ldots, I_h), M)\) This algorithm takes as input the public key \(PK\), hierarchical attribute \((I_1, \ldots, I_h)\), and a message \(M\) from the associated message space. It outputs a ciphertext \(C\).

Decrypt\((C, SK_{f_1, \ldots, f_l})\) This algorithm takes as input a ciphertext \(C\) and a secret key \(SK_{f_1, \ldots, f_l}\). It outputs either a message \(M\) or the distinguished symbol \(\perp\).
Delegate\((SK_1, \ldots, f_i, f_{l+1})\) This algorithm takes as input an \(l\)-th level secret key \(SK_1, \ldots, f_i\) and \((l+1)\)-th level predicate \(f_{l+1}\). It outputs an \((l+1)\)-th level secret key \(SK_1, \ldots, f_{l+1}\).

Correctness. The correctness property of the schemes says that for all \(\lambda, \Delta\), all \((PK, MSK)\) output by \text{Setup}\((\lambda, \Delta)\) algorithm, all hierarchical predicate \((f_1, \ldots, f_l)\), any key \(SK_1, \ldots, f_i\) generated by \text{GenKey}\((MSK, (f_1, \ldots, f_l))\) algorithm, and all hierarchical attributes \((I_1, \ldots, I_h)\) \in \mathcal{A},

- If \(\forall i \in \{1, \ldots, l\} : f_i(I_i) = 1\), then \(\text{Decrypt}(\text{Encrypt}(PK, (I_1, \ldots, I_h), M), SK_1, \ldots, f_i) = M\).

- If \(\exists i \in \{1, \ldots, l\} : f_i(I_i) = 0\), then \(\text{Decrypt}(\text{Encrypt}(PK, (I_1, \ldots, I_h), M), SK_1, \ldots, f_i) = \bot\) with all but negligible probability.

2.3.2 Security Definition

Definition 2.18. A Hierarchical Predicate Encryption Scheme is adaptively attribute hiding against chosen plaintext attacks if for all PPT adversaries \(\mathcal{A}\), the advantage \(\text{Adv}_{\mathcal{A}}^{\text{HPE}}\) of \(\mathcal{A}\) in the following game is a negligible function of the security parameter \(\lambda\):

\textbf{Setup.} A challenger \(C\) runs the \text{Setup} algorithm to generate public key \(PK\) and master secret key \(MSK\), and \(PK\) is given to \(A\).

\textbf{Query phase 1.} \(A\) may adaptively make a polynomial number of queries of the following type:

- \textbf{Create key:} \(A\) asks the challenger to create a secret key for a hierarchical predicate \(F\). The challenger creates a key for \(F\) without giving it to \(A\).

- \textbf{Create delegated key:} \(A\) specifies a key for hierarchical predicate \(F\) that has already been created, and asks the challenger to perform a delegation operation to create a child key for \(F' \leq F\). The challenger computes the child key without giving it to the adversary.

- \textbf{Reveal key:} \(A\) asks the challenger to reveal an already created key for predicate \(F\).

Note that when key creation requests are made, \(A\) does not automatically see the created key. \(A\) sees a key only when it makes a reveal key query.

\textbf{Challenge.} \(A\) outputs its challenge, containing two attributes \((Y^{(0)}, Y^{(1)})\) = \((I_1^{(0)}, \ldots, I_{h_0}^{(0)}), (I_1^{(1)}, \ldots, I_{h_1}^{(1)})\) and two plaintexts \((M^{(0)}, M^{(1)})\) subject to the restriction that \(F(Y^{(0)}) = F(Y^{(1)})\), if \(F(Y^{(0)}) = F(Y^{(1)})\) = 1, then it is required that \(M^{(0)} = M^{(1)}\), for all the reveal key queried predicate \(F\).

\(C\) then flips a random coin \(b\). If \(b = 0\) then \(A\) is given \(C = \text{Encrypt}(PK, Y^{(0)}, M^{(0)})\) and if \(b = 1\) then \(A\) is given \(C = \text{Encrypt}(PK, Y^{(1)}, M^{(1)})\).  

\textbf{Query phase 2.} Repeat the \textbf{Query phase 1} subject to the restrictions as in the challenge phase. In addition, when the adversary make a key query for \(F'\) such that \(F(Y^{(0)}) = F(Y^{(1)})\) = 1, we require that the adversary cannot delegate a child key \(F'\) from \(F\) such that \(F'(Y^{(0)}) \neq F'(Y^{(1)})\).
Guess. \( A \) outputs a bit \( b' \), and succeeds if \( b' = b \).

We define the advantage of \( A \) as the quantity \( \text{Adv}_{A}^{\text{HPE}}(\lambda) = |\Pr[b = b'] - 1/2| \).

### 2.4 Inner Product Encryption and its Applications

Katz, Sahai, and Waters [34] introduced PE supporting Inner-Product Encryption (IPE), i.e. vector \( \vec{y} \) represents attributes and vector \( \vec{x} \) determines the predicate \( f_{\vec{x}} \) such that \( f_{\vec{x}}(\vec{y}) = 1 \) iff \( \vec{x} \cdot \vec{y} = 0 \) (\( \vec{x} \cdot \vec{y} \) denotes the inner product of vectors \( \vec{x} \) and \( \vec{y} \) over a field or ring). We review how to leverage inner product encryption (IPE) to achieve a wide class of predicates [34].

#### 2.4.1 Anonymous Identity-Based Encryption

In Anonymous Identity-Based Encryption [12, 16], an identity is encrypted with a message in the ciphertext, i.e., the identity is hidden in the ciphertext. Without the key for the identity, it is hard to tell which identity is encoded in the ciphertext. Since the identity can be viewed as an attribute, we show that anonymous identity-based encryption can be recovered by IPE. Let \( (\text{Setup}_{IPE}, \text{GenKey}_{IPE}, \text{Encrypt}_{IPE}, \text{Decrypt}_{IPE}) \) be a secure IPE.

- **Setup** is the same as \( \text{Setup}_{IPE} \).

- **GenKey\((MSK, ID)\)**, where \( ID \in \mathbb{F}_q \) is an identity. Set \( \vec{I} = (1, ID) \) and output \( SK_{ID} \leftarrow \text{GenKey}_{IPE}(MSK, \vec{I}) \).

- **Encrypt\((PK, M, ID')\)**, where \( ID' \) is an identity. To encrypt a message \( M \) for identity \( ID' \), set \( \vec{I}' = (-ID', 1) \) and output \( C \leftarrow \text{Encrypt}_{IPE}(PK, \vec{I}', M) \).

- **Decrypt\((C, SK_{ID})\)**. Output \( \text{Decrypt}_{IPE}(C, SK_{ID}) \).

To verify the correctness of the system, we have:

\[
\vec{I} \cdot \vec{I}' = 0 \quad \text{iff} \quad ID = ID',
\]

correctness and security follow.

#### 2.4.2 Hidden-Vector Encryption

In hidden vector encryption (HVE) [15], secret keys and ciphertexts are associated with vectors. A ciphertext can be decrypted with a key, if the elements from key vector match the elements of vector in the ciphertext. Let \( \Sigma = \Sigma \cup \{*\} \), where \( \Sigma \) is a set. HVE corresponds to a predicate encryption scheme for the class of predicates \( \Psi_{HVE}^{l} = \{\phi_{HVE}^{l}(a_{1}, \ldots, a_{l}) | a_{1}, \ldots, a_{l} \in \Sigma\} \), where

\[
\phi_{HVE}^{l}(x_{1}, \ldots, x_{l}) = \begin{cases} 
1, & \text{if either } a_{i} = x_{i} \text{ or } a_{i} = *, \text{ for } i = 1, \ldots, l; \\
0, & \text{otherwise.}
\end{cases}
\]

HVE can be realized by IPE \( (\text{Setup}_{IPE}, \text{GenKey}_{IPE}, \text{Encrypt}_{IPE}, \text{Decrypt}_{IPE}) \) of dimension \( 2l \):
2.4. Inner Product Encryption and its Applications

- Setup is the same as Setup_{IPE}.

- GenKey(\(MSK, \psi_{HVE}^{(a_1, \ldots, a_i)}\)). To generate a secret key corresponding to the predicate \(\psi_{HVE}^{(a_1, \ldots, a_i)}\), first construct a vector \(\hat{A} = (A_1, \ldots, A_2l)\) as follows:
  
  \[
  \begin{align*}
  \text{if } a_i \neq \ast & : A_{2i-1} = 1, \ A_{2i} = a_i \\
  \text{if } a_i = \ast & : A_{2i-1} = 0, \ A_{2i} = 0
  \end{align*}
  \]

  then output the key by running \(SK_{\psi} \leftarrow \text{GenKey}_{IPE}(MSK, \hat{A})\).

- Encrypt(\(PK, M, \vec{x}\)), where \(\vec{x} = (x_1, \ldots, x_l)\). To encrypt a message \(M\) for the attribute \(\vec{x} = (x_1, \ldots, x_l)\), first construct a vector \(\vec{X} = (X_1, \ldots, X_{2l})\) as follows:
  
  \[
  X_{2i-1} = -r_i \cdot x_i, \ X_{2i} = r_i,
  \]

  where \(r_1, \ldots, r_l \leftarrow \mathbb{F}_q\). Finally, output the ciphertext \(C \leftarrow \text{Encrypt}_{IPE}(PK, \vec{X}, M)\).

- Decrypt(\(C, SK_{\psi}\)). Output Decrypt_{IPE}(C, SK_{\psi}).

To verify the correctness of the system, we have:

\[\psi_{HVE}^{(a_1, \ldots, a_i)}(x_1, \ldots, x_l) = 1 \Rightarrow \vec{X} \cdot \vec{A} = 0,\]

correctness and security follow.

2.4.3 PE Supporting Polynomial Evaluation

We can also construct PE for the predicates corresponding to polynomial evaluation \([34]\). Let \(\Psi^{\text{poly}}_{\leq d} = \{f_p \mid p \in \mathbb{F}_q[x], \deg(p) \leq d\}\), where

\[
f_p(x) = \begin{cases} 
1, & \text{if } p(x) = 0; \\
0, & \text{otherwise.}
\end{cases}
\]

for \(x \in \mathbb{F}_q\). A predicate encryption for \(\Psi^{\text{poly}}_{\leq d}\) can be realized by IPE (Setup_{IPE}, GenKey_{IPE}, Encrypt_{IPE}, Decrypt_{IPE}) of dimension \(d + 1\):

- Setup is the same as Setup_{IPE}.

- GenKey(\(MSK, p = a_dx^d + \cdots + a_0x^0\)). To generate a secret key corresponding to the polynomial \(p = a_dx^d + \cdots + a_0x^0\), set \(\vec{p} = (a_d, \ldots, a_0)\), then output the key by running \(SK_{\vec{p}} \leftarrow \text{GenKey}_{IPE}(MSK, \vec{p})\).

- Encrypt(\(PK, M, w \in \mathbb{F}_q\)). To encrypt a message \(M\) for the attribute \(w\), set \(\vec{w} = (w^d, \ldots, w^0)\), output the ciphertext \(C \leftarrow \text{Encrypt}_{IPE}(PK, \vec{w}, M)\).

- Decrypt(\(C, SK_{\vec{p}}\)). Output Decrypt_{IPE}(C, SK_{\vec{p}}).
To verify the correctness of the system, we have:

\[ p(w) = 0 \Rightarrow \overline{p} \cdot \overline{w} = 0, \]

correctness and security follow.

### 2.4.4 Evaluating Disjunctions and Conjunctions

Based on the polynomial-based constructions of the previous section, we can build predicate encryption for disjunctions of equality tests, e.g., the predicate OR_{I_1, I_2}, where OR_{I_1, I_2}(x) = 1 iff either \( x = I_1 \) or \( x = I_2 \), can be encoded as the univariate polynomial

\[ p(x) = (x - I_1) \cdot (x - I_2), \]

which evaluates to 0 iff the relevant predicate evaluates to 1. Similarly, the predicate OR_{a_1, a_2}, where OR_{a_1, a_2}(x_1, x_2) = 1 iff either \( x_1 = a_1 \) or \( x_2 = a_2 \), can be encoded as the bivariate polynomial

\[ p'(x_1, x_2) = (x_1 - a_1) \cdot (x_2 - a_2). \]

Conjunctions can be evaluated in a similar way. For example, the predicate AND_{I_1, I_2}, where AND_{I_1, I_2}(x_1, x_2) = 1 iff both \( x_1 = I_1 \) and \( x_2 = I_2 \), can be constructed as follows. We decide the relevant secret key by choosing a random \( r \in \mathbb{F}_q \) and letting the secret key correspond to the polynomial

\[ p''(x_1, x_2) = r(x_1 - I_1) + (x_2 - I_2). \]

Note that if AND_{I_1, I_2}(x_1, x_2) = 1 then \( p''(x_1, x_2) = 0 \), whereas if AND_{I_1, I_2}(x_1, x_2) = 0 then, with all but negligible probability over choice of \( r \), it will hold that \( p''(x_1, x_2) \neq 0 \).

### 2.5 Provable Security

Provable security is an important technique in cryptography. When a new cryptographic scheme is constructed, we need to make sure that it is secure in some specific security model, since the scheme may be vulnerable to some potential attacks. For example, Rabin’s scheme was believed to be equivalent in security to its underlying computational problem, but it was broken when the attacker model was changed. Informally, in this approach, security of a scheme is proved by reductionist style, which reduces security of a defined notion to an underlying hard problem. Indeed, the idea of reduction is from the theory of computation; more inquisitive readers can refer to for further details. The concept of provable security was proposed by Goldwasser and Micali in 1984. The outline of the method is listed as follows:

1. Define a security goal (e.g. unforgeability of signature or indistinguishability of encryption).

2. Construct a security model and set up an experiment between the potential attacker and the simulated real world environment.
2.5. Provable Security

3. Decide what it means for the cryptographic scheme to be secure.


5. Use the reductionist style to show that the only way to break the scheme is to solve the underlying computationally hard problem.

2.5.1 Proof Techniques

We say a scheme is secure in the standard model if that scheme can be proved secure using only complexity assumptions. In the standard model, an adversary is only limited by the amount of time and computational power available. All the proposed schemes in this thesis are proved to be secure in the standard model.

The main technique used in our proof is sequence-of-games approach (or game hopping) \[5, 22, 57\]. By applying this technique, the security proof is organized as sequences of games, which may appear more complicated otherwise. In the proof employing game hopping, we start with original security game that is played between an adversary and a challenger that simulates the attack environment. We then change the games step by step until we can evaluate the success probability of the adversary against the scheme. We will count on all the upper bound on the increase in the success probability of the adversary when changes from one game to another. To obtain a bound on the success probability of the adversary in the original game, we will consider all the bounds computed from the rest of games.

Let \( \Pr[S_i] \) be the probability that an event \( S_i \) occurs. Transitions between games can be classified as one of the following types:

**Transitions based on indistinguishability.** Assume distributions \( P_1 \) and \( P_2 \) are computationally indistinguishable for distinguisher \( D \). In the transition, we change from Game \( i \) to Game \( i + 1 \) by replacing the value received by adversary from distribution \( P_1 \) to that of \( P_2 \). Since indistinguishability assumption holds, i.e., the probability of distinguishing \( P_1 \) and \( P_2 \) is negligible, success probability of the adversary will be increased by a negligible amount. We establish that \( |\Pr[S_i] - \Pr[S_i+1]| \) is negligible.

**Transitions based on failure events.** In this type of transition, one argues that Games \( i \) and Game \( i + 1 \) are exactly the same except that some “failure event” \( F \) occurs. The events \( S_i \land \neg F \) and \( S_{i+1} \land \neg F \) are the same if both games are defined over the same probability space. We then have \( |\Pr[S_i] - \Pr[S_{i+1}]| \leq \Pr[F] \) due to the difference lemma \[57\]. Thus, to prove that \( \Pr[S_i] \) is negligibly close to \( \Pr[S_{i+1}] \), we only need to prove that \( \Pr[F] \) is negligible.

**Bridging steps.** Bridging step is introduced in this transition, which is a way of restating how certain quantities can be computed in an equivalent way. Hopping from Game \( i \) to Game \( i + 1 \), we obtain \( \Pr[S_i] = \Pr[S_{i+1}] \). Bridging step, although seems not necessary, prepares the ground for a transition of one of the above two types.
Chapter 3

Predicate Encryption for Multi Inner Product

In this chapter, we propose a PE system that can be used to evaluate a wide class of predicates, including disjunctive comparison and disjunctive subset queries. Furthermore, our system can be extended to achieve arbitrary conjunctive and disjunctive queries. We provide proof of the proposed schemes and show some method to improve the efficiency of the system.

3.1 Introduction

Prior work on predicate encryption [15,34,54,56] has focused on expanding the expressiveness of predicates. The most expressive system proposed by Katz, Sahai and Waters [34] can be employed to evaluate a wide class of predicates such as conjunctions of equality, comparison and subset queries, disjunctions of equality queries, and more generally, arbitrary combinations of conjunctive and disjunctive equality queries. For example, in a banking system, consider a bank manager that checks a number of encrypted transactions. The manager may request a key for a predicate $P$ from an authority. The authority checks the eligibility of the manager and gives the key to the manager, then the manager can use the secret key to identify all transactions satisfying the predicate $P$. A manager may obtain a key for a predicate such as $(\text{TransactionTime} > 4\text{PM}) \land (\text{TransactionType} = \text{Debit})$, or even more complicated predicates like $(\text{TransactionTime} = 4\text{PM}) \land ( (\text{TransactionType} = \text{Debit}) \lor (\text{TransactionMonth} = \text{January}) )$. The above examples can be implemented using existing schemes. However, the following example cannot be implemented using existing schemes. Let’s consider the predicate $(\text{TransactionTime} > 4\text{PM}) \land ( (\text{TransactionType} = \text{Debit}) \lor (\text{TransactionMonth} \in \{\text{January, March, July}\}) )$. Unfortunately, the current attribute-hiding (attribute is hidden in the ciphertext) systems [15,34,42] cannot handle this predicate, since the existing schemes don’t support arbitrary conjunctions and disjunctions of comparison queries, and arbitrary conjunctions and disjunctions of subset queries.
Chapter 3. Predicate Encryption for Multi Inner Product

3.1.1 Our Results

The aim of this work is to improve the expressiveness of predicate encryption. Firstly, we construct systems that support disjunctive comparison and disjunctive subset queries. As a main result of this work, we then show how to employ our technique to achieve arbitrary combinations of conjunctive and disjunctive equality, comparison and subset predicate evaluations. Finally, we discuss methods to reduce the sizes of ciphertext and secret key in our system. In summary, our contributions are as follows:

- We propose two predicate encryption schemes evaluating multi inner-product predicate (PEM). The proposed PEMs can be used to evaluate disjunctive comparison and disjunctive subset queries as well as all predicates supported by the existing PE \cite{15,34,42}. The first scheme is constructed from \cite{42} in a direct way, i.e., we elaborately construct the attribute/predicate vectors that can be used to evaluate multi inner-product predicate, and encode the attribute and predicate vectors in the ciphertext and key respectively. In our first scheme, the size of public parameter increases quadratically when the number of predicates in multi inner-product predicate increases. To alleviate this problem, we propose our second scheme, where the sizes of public keys are significantly reduced. The sizes of ciphertext and key are still comparable to the first scheme. It is non trivial to construct such system for two reasons. First, the PE \cite{42} on which our scheme is based uses matrix construction over prime order group to compute the public key. It is not known how to reduce public parameters under such technique. Moreover, the ciphertext and key should be consistent with the reduced public keys which is not straightforward. Secondly, when we try to improve the performance of the system, we should make sure that attribute hiding property is not compromised. To achieve such efficiency we split the public key into multiple sub public keys in a way that the size of the latter is much smaller than that of the former. The ciphertexts and keys are also divided into multiple components and re-combined in some way. A kind of secret sharing is used to obtain the property of collusion resistance, where a key must depend on all predicates in the multi inner-product predicate. The attribute hiding property seems a natural extension of the PE, but the resulting ciphertext consists of multiple components with which different attributes are encoded. This makes the security analysis more complicated than that of our first scheme.

- A security analysis for the proposed schemes is presented. The first scheme is selectively attribute hiding against chosen plaintext attacks (CPA) in the standard model under two assumptions: the RDSP and IDSP assumptions \cite{42}. The second scheme is more efficient than the first one in terms of the public key size, and the security of the second scheme is based on multi-instance version of RDSP and IDSP assumptions: the $l$-RDSP and $l$-IDSP assumptions.

- We show how to leverage the proposed PEM to realize the system supporting disjunctions of comparison and disjunctions of subset queries. To achieve this goal, we combine some techniques from the hidden vector encryption (HVE) \cite{15} and predicate encryption for inner product (IPE) \cite{34}. The intuition for the design can be found in Section 3.4.
3.2 Complexity Assumptions

As the main result of this chapter, we extend our technique for disjunctive queries to achieve arbitrary combinations of conjunctive and disjunctive equality, comparison and subset predicate evaluations. The sizes of the ciphertext and secret key depend on the number of disjunctive operators in the predicates in some specific forms. For example, to evaluate a predicate \( \bigwedge_{i=1}^{n}(V_{j=1}^{m_{i}} s_{i,j}) \), where \( s_{i,j} \) is one of the equality, comparison and subset query, our system achieves both ciphertexts and user private keys of size \( O(Nl_{\max}) \), where \( N \) is the length of the maximum inner-product vector for the predicate \( s_{i,j} \), and \( l_{\max} = \max\{m_{1}, \ldots, m_{n}\} \). To optimize the performance of the system, we discuss some related issues such as CNF and DNF, and propose a heuristic method to reduce disjunctions in the predicates. Finally, we show that it is possible to achieve adaptive security with the existing techniques.

3.2 Complexity Assumptions

In this section, we define two assumptions, the RDSP and IDSP assumptions \([42]\). Firstly, we construct a DPVS generator \( G_{dpvs} \) defined in definition 2.13. Then, a random orthonormal basis generator \( G_{ob} \) is defined as follows:

\[
G_{ob}(1^\lambda, N) : \text{param} = (q, V, G_T, A, e) \xleftarrow{\$} G_{dpvs}(1^\lambda, N)
\]

\[
\Lambda = (\lambda_{i,j}) \xleftarrow{\$} GL(N, \mathbb{F}_q), \quad \mu_{i,j} = (\Lambda^T)^{-1},
\]

\[
b_i = \sum_{j=1}^{N} \lambda_{i,j} a_j, \quad \mathbb{B} = (b_1, \ldots, b_N), \quad b_i^* = \sum_{j=1}^{N} \mu_{i,j} a_j, \quad \mathbb{B}^* = (b_1^*, \ldots, b_N^*),
\]

return \((\text{param}, \mathbb{B}, \mathbb{B}^*)\).

We define the RDSP instance generator \( G_{RDSP}^{\beta} \):

\[
G_{RDSP}^{\beta}(1^\lambda, n) : (\text{param}, \mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} G_{ob}(1^\lambda, n + 3), \quad \overline{y} = (y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{F}_q \setminus \{0\},
\]

\[
\delta_1, \delta_2, \zeta_1, \zeta_2 \xleftarrow{\$} \mathbb{F}_q, \quad d_{n+1} = b_{n+1} + b_{n+2}, \quad \mathbb{B} = (b_1, \ldots, b_n, d_{n+1}, b_{n+3}),
\]

\[
(\omega^{(k)}_1, \gamma_1^{(k)}, \gamma_2^{(k)})_{k=1,2,3} \xleftarrow{\$} GL(\mathbb{F}_q, 3),
\]

For \( i = 1, \ldots, n; \; k = 1, 2, 3; \)

\[
h_i^{(k)} = \omega^{(k)} b_i^* + \gamma_1^{(k)} y_i b_{n+1}^{*} + \gamma_2^{(k)} y_i b_{n+2}^{*}, \quad \tau_i^{(k)} = (\gamma_1^{(k)} + \gamma_2^{(k)}) y_i,
\]

\[
e_0 = \delta_1 \left( \sum_{i=1}^{n} y_i b_i \right) + \delta_2 b_{n+3},
\]

\[
e_1 = \delta_1 \left( \sum_{i=1}^{n} y_i b_i \right) + \zeta_1 b_{n+1} + \zeta_2 b_{n+2} + \delta_2 b_{n+3},
\]

return \((\text{param}, \mathbb{B}, \{h_i^{(k)}, \tau_i^{(k)}\}_{i=1,\ldots,n; \; k=1,2,3}, \overline{y}, e_\beta)\).

The IDSP instance generator \( G_{IDSP}^{\beta} \) is shown below:

\[
G_{IDSP}^{\beta}(1^\lambda, n) : (\text{param}, \mathbb{B}, \mathbb{B}^*) \xleftarrow{\$} G_{ob}(1^\lambda, n + 3),
\]

\[
\overline{y} = (y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}, \quad \overline{u} = (u_1, \ldots, u_n) \xleftarrow{\$} \mathbb{F}_q \setminus \{0\},
\]
Definition 3.1 (Decisional Subspace Problem with Relevant Dual Vector Tuples (RDSP) [42]).
For all security parameter \( \lambda \in \mathbb{N} \), we define RDSP advantage of a probabilistic machine \( \mathcal{B} \) as follows:

\[
\text{Adv}^\text{RDSP}_{\mathcal{B}}(\lambda) = | \Pr[\mathcal{B}(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{\text{R}} \mathbb{F}_q^{\text{RDSP}}(1^\lambda, n)] - \Pr[\mathcal{B}(1^\lambda, \rho) \rightarrow 1 \mid \rho \xleftarrow{\text{R}} \mathbb{F}_q^{\text{IDSP}}(1^\lambda, n)] |
\]

The RDSP assumption holds, if \( \text{Adv}^\text{RDSP}_{\mathcal{B}}(\lambda) \) is negligible in \( \lambda \) for any probabilistic polynomial-time adversary \( \mathcal{B} \).

Definition 3.2 (Decisional Subspace Problem with Irrelevant Dual Vector Tuple (IDSP) [42]).
The IDSP advantage of \( \mathcal{B} \), \( \text{Adv}^\text{IDSP}_{\mathcal{B}}(\lambda) \), and the IDSP assumption are defined similarly as in Definition 3.1.

Remark 3.1. Here we give the intuition of RDSP assumption by using a simplified one. The simplified RDSP assumption is that given \( (\mathbb{B}, \mathbb{y}, \{\mathbf{h}_i\}_{i=1}^n, \mathbf{e}_\beta) \), where \( \mathbb{B} = (b_1, \ldots, b_{n+2}), \mathbb{y} = (y_1, \ldots, y_n), \mathbf{h}_i = \omega b_i^* + y_i b_{n+1}^* (i = 1, \ldots, n; \omega \xleftarrow{\text{U}} \mathbb{F}_q) \), and \( \mathbf{e}_\beta = \delta_1 (\sum_{i=1}^n y_i b_i) + \beta \mathbf{b}_{n+1} + \delta_2 \mathbf{b}_{n+2} (\beta \xleftarrow{\text{U}} \{0, 1\}, \delta_1, \delta_2, \zeta \xleftarrow{\text{U}} \mathbb{F}_q) \), it is hard for an adversary \( \mathcal{A} \) to correctly guess \( \beta \). \( \{\mathbf{h}_i\}_{i=1}^n \) is used to simulate the key generation in the security proof: for predicate \( \overrightarrow{x} = (x_1, \ldots, x_n) \) with \( \overrightarrow{x} \cdot \mathbb{y} \neq 0 \), a secret key \( \mathbf{k}^* \) for \( \overrightarrow{x} \) can be computed as \( \mathbf{k}^* = \frac{1}{\overrightarrow{x} \cdot \mathbb{y}} \sum_{i=1}^n x_i \mathbf{h}_i^* = \frac{1}{\overrightarrow{x} \cdot \mathbb{y}} \sum_{i=1}^n x_i b_i^* + b_{n+1}^* = \omega' \sum_{i=1}^n x_i b_i^* + b_{n+1}^* \) where \( \omega' = \frac{1}{\overrightarrow{x} \cdot \mathbb{y}} \). If \( \overrightarrow{x} \cdot \mathbb{y} = 0 \), \( \frac{1}{\overrightarrow{x} \cdot \mathbb{y}} \) cannot be computed. \( \{\mathbf{h}_i^*\}_{i=1}^n \) does not seem helpful to break the RDSP assumption, since a secret key \( \mathbf{k}^* \) for \( \overrightarrow{x} \) with \( \overrightarrow{x} \cdot \mathbb{y} = 0 \) can be used to identify \( \beta \) by checking whether \( c(\mathbf{k}^*, \mathbf{e}_\beta) = 1 \) or not.

With the similar argument, the IDSP assumption also seems hold.

3.3 Security Model

We define predicate encryption for the class of multi inner-product predicates (PEM) and its security. A vector symbol denotes a vector representation over \( \mathbb{F}_q \). For example, \( \overrightarrow{x} \) denotes \( (x_1, \ldots, x_n) \in \mathbb{F}_q^n \). \( \overrightarrow{x} \cdot \mathbb{y} \) denotes the inner product \( \sum_{i=1}^n x_i y_i \) of two vectors \( \overrightarrow{x} = (x_1, \ldots, x_n) \) and \( \mathbb{y} = (y_1, \ldots, y_n) \). We take a tuple \( (n, l) \), where \( n \) is the dimension of each vector and \( l \) represents the number of vectors. (The dimension of each vector may be different. However, for simplicity, we assume that all vectors have the same dimension.) Let \( \Sigma_j = \mathbb{F}_q^n \setminus \{\mathbb{0}\} \) be the sets of attributes for \( j = 1, \ldots, l \). Let the multi attributes \( \Sigma = \Sigma_1 \times \ldots \times \Sigma_l \). We take our
class of predicates to be $F = \{ f(x_1, \ldots, x_i) \mid x_i \in \mathbb{F}_p \} \setminus \{ 0 \}$ where $f(x_1, \ldots, x_i)(y_1, \ldots, y_i) = 1$ if $\prod_{j=1}^{l} x_j y_j = 0$. ($\prod_{j=1}^{l} x_j y_j$ represents the product of all the inner products.)

**Definition 3.3.** Let $n$ be the dimension of a vector and $l$ be the number of vectors. A Predicate Encryption Scheme for Multi Inner Products comprises of the following algorithms:

- **Setup($1^\lambda, n, l$)** The Setup algorithm takes as input a security parameter $1^\lambda$, $n$ and $l$, and outputs a public key $PK$ and a master secret key $MSK$.

- **GenKey($MSK, (\vec{x}_1, \ldots, \vec{x}_l)$)** The GenKey algorithm takes as input a master secret key $MSK$ and predicate vectors $(\vec{x}_1, \ldots, \vec{x}_l)$. It outputs an associated secret key $SK(\vec{x}_1, \ldots, \vec{x}_l)$.

- **Encrypt($PK, (\vec{y}_1, \ldots, \vec{y}_l), M$)** The Encrypt algorithm takes as input a public key $PK$, attribute vectors $(\vec{y}_1, \ldots, \vec{y}_l)$, and a message $M$ in some associated message space. It outputs a ciphertext $C$.

- **Decrypt($SK(\vec{x}_1, \ldots, \vec{x}_l), C$)** The Decrypt algorithm takes as input a secret key $SK(\vec{x}_1, \ldots, \vec{x}_l)$ and a ciphertext $C$. It outputs either a message $M$ or the distinguished symbol $\perp$.

**Correctness.** We require the following correctness property. For all $\lambda$, $n$ and $l$, let $(PK, MSK)$ $\xleftarrow{\$}$ Setup($1^\lambda, n, l$), $C \xleftarrow{\$}$ Encrypt($PK, (\vec{y}_1, \ldots, \vec{y}_l), M$), and $SK(\vec{x}_1, \ldots, \vec{x}_l)$ $\xleftarrow{\$}$ GenKey($MSK, (\vec{x}_1, \ldots, \vec{x}_l)$).

- If $f(\vec{x}_1, \ldots, \vec{x}_l)(\vec{y}_1, \ldots, \vec{y}_l) = 1$ then $\text{Decrypt}(SK(\vec{x}_1, \ldots, \vec{x}_l), C) = M$.
- If $f(\vec{x}_1, \ldots, \vec{x}_l)(\vec{y}_1, \ldots, \vec{y}_l) = 0$ then $\text{Decrypt}(SK(\vec{x}_1, \ldots, \vec{x}_l), C) = \perp$ with all but negligible probability.

**Definition 3.4.** A Predicate Encryption Scheme with respect to $F$ and $\Sigma$ is selectively attribute hiding against chosen plaintext attacks if for all PPT adversaries $A$, the advantage $\text{Adv}_A$ of $A$ in the following game is a negligible function of the security parameter:

- **Init.** $A$ outputs challenge vectors $Y^{0} = (\vec{y}_1^{0}, \ldots, \vec{y}_l^{0})$, $Y^{1} = (\vec{y}_1^{1}, \ldots, \vec{y}_l^{1})$.

- **Setup.** A challenger $C$ runs the Setup algorithm to generate public key $PK$ and master secret key $MSK$, and $PK$ is given to $A$.

- **Query phase 1.** $A$ adaptively makes a polynomial number of secret key queries for any predicates $f_1, \ldots, f_m \in F$. The keys revealed to $A$ are subject to the restriction such that $f_i(Y^{0}) = f_i(Y^{1}) = 0$ for all $i = 1, \ldots, m$.

- **Challenge.** $A$ outputs two equal-length messages $M^{0}$ and $M^{1}$. $C$ flips a random coin $b$. If $b = 0$ then $A$ is given $C$ = Encrypt($PK, Y^{0}, M^{0}$) and if $b = 1$ then $A$ is given $C$ = Encrypt($PK, Y^{1}, M^{1}$).

- **Query phase 2.** Repeat the Query phase 1 subject to the restrictions as before.

- **Guess.** $A$ outputs a guess $b'$ of $b$, and succeeds if $b' = b$.

The advantage of $A$ is defined to be $\text{Adv}_A(\lambda) = |\text{Pr}[b = b'] - 1/2|$.
Remark: In the security definition of [34], an adversary is allowed to request keys for $(\vec{v}_1, \ldots, \vec{v}_d)$ such that $f_i(\vec{v}_1, \ldots, \vec{v}_d)(Y^{(0)}) = f_i(\vec{v}_1, \ldots, \vec{v}_d)(Y^{(1)}) = 1$ provided that $M^{(0)} = M^{(1)}$. However, those queries are not allowed in Definition [34]. This restriction is introduced to prove the security of the proposed scheme only under the RDSP and IDSP assumptions. Okamoto and Takashima [42] showed that we can relax the restriction with some variants of those assumptions.

### 3.4 The Intuition of Our Design

Before presenting our schemes, we explain the key idea behind our constructions. The hidden vector encryption (HVE) [15] can be used to achieve equality, comparison or subset predicate evaluations. Katz, Sahai and Waters [34] showed that any predicate encryption for inner product can be transformed into the corresponding vector in inner-product encryption, which allows us to apply inner-product system to evaluate all predicates supported by HVE. On the other hand, it has been shown that disjunctive equality, or more complex combinations of disjunctive and conjunctive equality queries can be achieved using inner-product encryption via polynomial evaluations [34]. It was noted that the complexity of the resulting scheme depends polynomially on $d^t$, where $t$ is the number of variables and $d$ is the maximum degree in each variable. However, it seems that this technique cannot be employed to achieve disjunctive comparison or subset queries.

We show that those queries can be achieved on the inner-product layer. It is easiest to explain with an example. Consider the predicate $P$: $P_1 \lor P_2$. $P_1$, $P_2$ can be one of the equality, comparison or subset queries. In the inner-product system, we let $\vec{x}_1$ associate with the predicate $P_1$ and $\vec{x}_2$ associate with $P_2$. Assume that $\vec{y}_1$ and $\vec{y}_2$ are the corresponding attributes in a ciphertext. To evaluate the predicate $P$, we need to construct a system so that the Decrypt algorithm will output $g_r^{(D_1, D_2)}$, where $r$ is randomly selected, $D_1 = \vec{x}_1 \cdot \vec{y}_1$ and $D_2 = \vec{x}_2 \cdot \vec{y}_2$. The reason is that $D_1 = 0$ or $D_2 = 0$ if $D_1 \cdot D_2 = 0$. Meanwhile, the predicate $P_1 = 1$ if $D_1 = \vec{x}_1 \cdot \vec{y}_1 = 0$ and $P_2 = 1$ if $D_2 = \vec{x}_2 \cdot \vec{y}_2 = 0$. We establish that the predicate $P = 1$ if $(\vec{x}_1 \vec{y}_1) \cdot (\vec{x}_2 \vec{y}_2) = 0$.

Since $(\vec{x}_1 \vec{y}_1) \cdot (\vec{x}_2 \vec{y}_2)$ contains the vector elements from both the ciphertext and secret key, we need to rephrase the formulae and distill the attribute vector and predicate vector for the ciphertext and secret key respectively. For simplicity, we assume that $\vec{x}_1$, $\vec{x}_2$, $\vec{y}_1$ and $\vec{y}_2$ are two dimensional vectors, i.e., each with two elements.

\[
(\vec{x}_1 \vec{y}_1) \cdot (\vec{x}_2 \vec{y}_2) = (x_1^{(1)} y_1^{(1)} + x_2^{(1)} y_2^{(1)}) \cdot (x_1^{(2)} y_1^{(2)} + x_2^{(2)} y_2^{(2)})
\]

\[
= x_1^{(1)} y_1^{(1)} x_1^{(2)} y_1^{(2)} + x_1^{(1)} y_1^{(1)} x_2^{(2)} y_2^{(2)} + x_2^{(1)} y_2^{(2)} x_1^{(2)} y_1^{(1)} + x_2^{(2)} y_2^{(2)} x_2^{(1)} y_1^{(1)}
\]

\[
= x_1^{(1)} x_1^{(2)} y_1^{(1)} + x_1^{(1)} x_2^{(2)} y_2^{(1)} + x_2^{(1)} x_1^{(2)} y_1^{(2)} + x_2^{(2)} x_2^{(1)} y_1^{(2)}
\]

Now we have the predicate vector $(x_1^{(1)} x_1^{(2)}, x_1^{(1)} x_2^{(2)}, x_2^{(1)} x_1^{(2)}, x_2^{(1)} x_2^{(2)})$ and the attribute vec-
3.5 Scheme 1

We now describe our first predicate encryption for multi inner products. As we have mentioned in Section 3.3 for simplicity, it is assumed that all vectors have the same dimension \( n \).

**Setup** \((1^3, n, l)\): On input \((1^3, n^l + 3)\), \(\mathcal{G}_{\text{ob}}\) outputs \((\mathbb{B}, \mathbb{B}^*, \text{param})\). Let \(d_{n^l+1} = b_{n^l+1} + b_{n^l+2}\).

The public key \(PK\) is \((\text{param}, b_1, \ldots, b_{n^l}, d_{n^l+1}, b_{n^l+3})\). The master secret key \(MSK\) is \((b_1^*, \ldots, b_{n^l}^*, b_{n^l+1}^*, b_{n^l+2}^*, b_{n^l+3}^*)\).

**GenKey** \((MSK, (\overline{x}_1, \ldots, \overline{x}_l)) = ((x_1^{(1)}, \ldots, x_n^{(1)}), \ldots, (x_1^{(l)}, \ldots, x_n^{(l)})) \in \mathbb{F}_{q}^{n^l}\): Choose random \(\sigma, \eta \in \mathbb{F}_q\) and compute decryption key \(k^*\):

\[
k^* = \sigma \left( \sum_{l \leq i_1 + i_2 + \ldots + i_l \leq n^l} x_{i_1}^{(1)} \ldots x_{i_l}^{(l)} b_{i_1}^* (i_2-1) n + \ldots + (i_l-1) n^{i_l-1} \right) + \eta b_{n^l+1}^* + (1 - \eta) b_{n^l+2}^*
\]
Encrypt \( \left( PK, M \in \mathbb{G}_T, (\vec{y}_1, \ldots, \vec{y}_l) = ((y_{1}^{(1)}, \ldots, y_{n}^{(1)}), \ldots, (y_{1}^{(l)}, \ldots, y_{n}^{(l)})) \in \mathbb{F}_q^n \right) \): Choose random \( \delta_1, \delta_2, \zeta \in \mathbb{F}_q \), then compute ciphertext \( C = (c_1, c_2) \):

\[
c_1 = \delta_1 \left( \sum_{l \leq i_1 + i_2 + \ldots + i_l \leq nl, 1 \leq i_1 \leq \ldots \leq i_l \leq [1,n]} y_{i_1}^{(1)} \cdots y_{i_l}^{(l)} b_{i_1 + (i_2 - 1)n + \ldots + (i_l - 1)n^{l-1}} \right) + \zeta d_{n^l + 1} + \delta_2 b_{n^l + 3}
\]

\[
c_2 = g_T^\zeta M
\]

Decrypt \( (k^*, C = (c_1, c_2)) \): Given a ciphertext \( C = (c_1, c_2) \) and a secret key \( k^* \), compute message \( M \):

\[
M = c_2/e(c_1, k^*)
\]

**Correctness.** To verify the correctness property holds for the scheme, let \( C \) and \( k^* \) be as above. Based on the explanations in Section 3.4 if \( \prod_{j=1}^{l} f j / y_j = 0 \), \( M \) can be recovered by computing \( c_2/e(c_1, k^*) \), since

\[
e(c_1, k^*) = \prod_{l \leq i_1 + i_2 + \ldots + i_l \leq nl, 1 \leq i_1 \leq \ldots \leq i_l \leq [1,n]} e(\delta_1 y_{i_1}^{(1)} \cdots y_{i_l}^{(l)} b_{i_1 + (i_2 - 1)n + \ldots + (i_l - 1)n^{l-1}}, \sigma \delta x_{i_1}^{(1)} \cdots x_{i_l}^{(l)} b_{n^l + 1} \cdot (1 - \eta) b_{n^l + 2})
\]

\[
= g_T^{\delta_1 \sigma} (\prod_{j=1}^{l} f j / y_j) + \zeta + (1 - \eta) = g_T^{\delta_1 \sigma} (\prod_{j=1}^{l} f j / y_j) + \zeta = g_T^\zeta.
\]

### 3.5.1 Security Proof of Scheme 1

**Theorem 3.1.** The scheme 1 described in Section 3.5 is selectively attribute hiding against chosen plaintext attacks under the RDSP and IDSP assumptions.

**Proof of Theorem 3.1**

**Lemma 3.1.** The scheme 1 is selectively attribute hiding against chosen plaintext attacks if the scheme of Okamoto and Takashima [42] is selectively attribute hiding against chosen plaintext attacks.

**Proof.** This proof is by contradiction. In order to prove Lemma 3.1 suppose there exists a polynomial time adversary \( A \) with non-negligible advantage \( \epsilon \) in the security Definition 3.4.

\( A \) outputs challenge vectors \( \mathcal{Y}^{(0)} = (\vec{y}_{1}^{(0)}, \ldots, \vec{y}_{l}^{(0)}), \mathcal{Y}^{(1)} = (\vec{y}_{1}^{(1)}, \ldots, \vec{y}_{l}^{(1)}) \), where \( (\vec{y}_{1}, \ldots, \vec{y}_{l}) = ((y_{1}^{(1)}, \ldots, y_{n}^{(1)}), \ldots, (y_{1}^{(l)}, \ldots, y_{n}^{(l)})) \in \mathbb{F}_q^{nl} \), then \( A \) passes \( \mathcal{Y}^{(0)} \) and \( \mathcal{Y}^{(1)} \) to \( A_0 \). \( A_0 \) uses \( A \) as a black box.

**Init.** \( A \) outputs challenge vectors \( \mathcal{Y}^{(0)} = (\vec{y}_{1}^{(0)}, \ldots, \vec{y}_{l}^{(0)}), \mathcal{Y}^{(1)} = (\vec{y}_{1}^{(1)}, \ldots, \vec{y}_{l}^{(1)}) \), where

\( (\vec{y}_{1}, \ldots, \vec{y}_{l}) = ((y_{1}^{(1)}, \ldots, y_{n}^{(1)}), \ldots, (y_{1}^{(l)}, \ldots, y_{n}^{(l)})) \in \mathbb{F}_q^{nl} \), then \( A \) passes \( \mathcal{Y}^{(0)} \) and \( \mathcal{Y}^{(1)} \) to \( A_0 \). \( A_0 \) computes:

\( \hat{\mathcal{Y}}^{(j)} = (y_{1}^{(1)} y_{2}^{(1)} \cdots y_{i_1}^{(1)} y_{1}^{(2)} \cdots y_{i_2}^{(2)} \cdots y_{i_l}^{(l)}, \ldots, y_{1}^{(l)} y_{2}^{(l)} \cdots y_{i_1}^{(l)} y_{1}^{(2)} \cdots y_{i_2}^{(2)} \cdots y_{i_l}^{(l)} y_{n}^{(1)} y_{n}^{(2)} \cdots y_{n}^{(l)}) \)

where \( j \in [1, 2], l \leq i_1 + i_2 + \ldots + i_l \leq nl \) and \( i_1, \ldots, i_l \in [1, n] \).
\( A_0 \) then gives \( \hat{Y}^{(0)} \) and \( \hat{Y}^{(1)} \) to challenger \( C \).

**Setup** \((1^\lambda, n, l)\). A challenger \( C \) runs the **Setup** algorithm to generate public key \( PK \) and master secret key \( MSK \). \( C \) sends \( PK \) to adversary \( A_0 \), then \( A_0 \) sends it to \( A \).

**Query phase 1.** \( A \) adaptively makes a polynomial number of secret key queries for any predicates \( f_1, \ldots, f_n \in \mathcal{F} \) subject to the restriction such that \( f_i(\hat{Y}^{(0)}) = f_i(\hat{Y}^{(1)}) = 0 \) for all \( i \). The query \( \mathcal{X}_i = (\hat{x}_{i1}, \ldots, \hat{x}_{il}) \) for predicate \( f_i \) is then sent to \( A_0 \). \( A_0 \) computes:

\[
\hat{X}_i = (x_{11} \hat{x}_{11}, \ldots, x_{1l} \hat{x}_{1l}, \ldots, x_{il} \hat{x}_{il}, \ldots, x_{nl} \hat{x}_{nl})
\]

where \( l \leq i_1 + i_2 + \ldots + i_l \leq nl \) and \( i_1, \ldots, i_l \in [1, n] \), and sends \( \hat{X}_i \) to challenger \( C \) to request the associated secret key. Upon receiving the secret key from \( C \), \( A_0 \) sends it to \( A \).

**Challenge.** \( A \) outputs two equal-length messages \( M^{(0)} \) and \( M^{(1)} \) and sends them to \( A_0 \). \( A_0 \) then passes \( M^{(0)} \) and \( M^{(1)} \) to challenger \( C \). \( C \) flips a random coin \( b \). If \( b = 0 \) then \( A_0 \) is given \( C = \text{Encrypt}(PK, \hat{Y}^{(0)}, M^{(0)}) \), otherwise \( A_0 \) is given \( C = \text{Encrypt}(PK, \hat{Y}^{(1)}, M^{(1)}) \). Upon receiving \( C \), \( A_0 \) sends it to \( A \).

**Query phase 2.** Repeat the **Query phase 1** subject to the restrictions as before.

**Guess.** \( A \) outputs a guess \( b' \) of \( b \), \( A_0 \) passes \( b' \) to challenger \( C \). \( A (A_0) \) succeeds if \( b' = b \). \( \square \)

### 3.6 Scheme 2

In this section, we present a variant of scheme 1. Scheme 2 has an advantage over scheme 1 in the sense that the size of the public key in the second scheme is much smaller than that in the first one. Meanwhile, the sizes of the ciphertext and secret key in scheme 2 are comparable to those in scheme 1. Instead of using dual orthonormal bases \( \mathbb{B} \) and \( \mathbb{B}^* \) with the size of \((n^l + 3) \times (n^l + 3)\), we use \( \{\mathbb{B}^{(i)}, \mathbb{B}^{* (i)}\}_{i=1}^{n^l - 1} \), where each \( \mathbb{B}^{(i)}, \mathbb{B}^{* (i)} \) is of size \((n + 3) \times (n + 3)\). We can see that the size of public key drops from \( O(n^{2l}) \) to \( O(n^l) \). One concern in the second scheme is that we should prevent the adversary using part of the secret key to test for the inner product vector. To solve this problem, we use a binding technique to prevent such attack: we use re-randomization to tie together the \( n^l - 1 \) sub-keys to ensure the integrity. Before describing the new scheme, we define a random orthonormal basis generator \( G_{\text{ob}} \) as follows:

\[
G_{\text{ob}}'(1^\lambda, N, l) : \text{param} = (q, \mathbb{V}, \mathbb{G}_T, \mu, e) \xleftarrow{\$} G_{\text{dpa}v}(1^\lambda, N)
\]

For \( k = 1, \ldots, (N - 3)^l \):

\[
\lambda^{(k)} = (\lambda^{(k)}_{i,j}) \xleftarrow{\$} GL(N, \mathbb{F}_q), \quad \mu^{(k)}_{i,j} = (\mu^{(k)}_{i,j})^{-1},
\]

\[
b^{(k)}_j = \sum_{j=1}^{N} \lambda^{(k)}_{i,j} a^{(k)}_j, \quad \mathbb{B}^{(k)} = (b^{(k)}_1, \ldots, b^{(k)}_N),
\]

\[
b^{*(k)}_i = \sum_{j=1}^{N} \mu^{(k)}_{i,j} d^{(k)}_j, \quad \mathbb{B}^{*(k)} = (b^{*(k)}_1, \ldots, b^{*(k)}_N),
\]
Next, we present our scheme.

Setup\((1^\lambda,n,l)\): Run \(G'_{ob}(1^\lambda,n+3,l-1)\) and output \((\text{param},\{\mathbb{B}^{(i)},\mathbb{B}^{(i)}_d\}_{i=1}^{n-1})\). Let \(\{d_n^{(i)} = b_n^{(i)}+b_n^{(i)}\}_{i=1}^{n-1}\). The public key \(PK\) is \((\text{param},\{b_i^{(i)},b_n^{(i)},d_n^{(i)},b_n^{(i)}\}_{i=1}^{n-1})\). The master secret key \(MSK\) is \((\{b_1^{(i)},b_1^{(i)},b_n^{(i)},b_n^{(i)}\}_{i=1}^{n-1})\).

GenKey\((MSK,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l) = ((x^{(1)}_1,\ldots,x^{(1)}_n),(x^{(l)}_1,\ldots,x^{(l)}_n)) \in \mathbb{F}_q^{nl})\): Choose random \(\eta_i \in \mathbb{F}_q\) for all \(i \in [1,n^{l-1}]\) and set \(\eta = \sum_{i=1}^{n^{l-1}} \eta_i\), also choose random \(\epsilon_i \in \mathbb{F}_q\) for all \(i \in [1,n^{l-1}]\) such that \(\sum_{i=1}^{n^{l-1}} \epsilon_i = 1 - \eta\), then choose random \(\sigma \in \mathbb{F}_q\) and compute decryption key \(k^*\):

\[
\forall i \in [1,n^{l-1}]: \ k_i = \sigma \left( \sum_{1 \leq j \leq n \atop i_2 + (i_3 - 1) \cdot n + \ldots + (i_l - 1) \cdot n^{l-2} = i} x^{(1)}_{i_2} x^{(2)}_{i_3} \ldots x^{(l)}_{i_l} b_j^{(i)} \right) + \eta_i b_n^{(i)} + \epsilon_i b_{n+2}^{(i)}.
\]

Encrypt\((PK,M \in \mathbb{G}_T, (\overrightarrow{y}_1,\ldots,\overrightarrow{y}_l) = ((y^{(1)}_1,\ldots,y^{(1)}_n),(y^{(l)}_1,\ldots,y^{(l)}_n)) \in \mathbb{F}_q^{nl})\): Choose random \(\delta^{(i)}_j \in \mathbb{F}_q\) for all \(i \in [1,n^{l-1}]\), and choose random \(\delta_1,\zeta \in \mathbb{F}_q\), then compute ciphertext \(C = (c_0,c_1,\ldots,c_{n^{l-1}})\):

\[
c_0 = g_T^\sigma M,\]

\[
\forall i \in [1,n^{l-1}]: \ c_i = \delta_1 \left( \sum_{1 \leq j \leq n \atop i_2 + (i_3 - 1) \cdot n + \ldots + (i_l - 1) \cdot n^{l-2} = i} y^{(1)}_j y^{(2)}_{i_2} \ldots y^{(l)}_{i_l} b_j^{(i)} \right) + \zeta b_{n+1}^{(i)} + \delta^{(i)}_2 b_{n+3}^{(i)}.
\]

Decrypt\((k^*,C)\): Given a ciphertext \(C\) and a secret key \(k^*\), compute message \(M\):

\[
M = c_0 \prod_{1 \leq i \leq n^{l-1}} e(c_i,k_i).
\]

It is straightforward to verify that scheme 2 is correctly constructed, since \(\prod_{1 \leq i \leq n^{l-1}} e(c_i,k_i) = g_T^\sigma (\prod_{j=1}^{l} \overrightarrow{x}_j, \overrightarrow{y}_j) + \zeta\), the correctness follows.

### 3.6.1 Security Proof of Scheme 2

To prove the security of the scheme, we introduce some variants of the RDSP and IDSP assumptions. We call them \(l\)-RDSP and \(l\)-IDSP assumptions. \(l\)-RDSP (resp. \(l\)-IDSP), which consists of multiple RDSP (resp. IDSP) instances, can be viewed as the generalized RDSP (resp. IDSP).
We now define the l-RDSP instance generators $G^l_{β-DSP}$.

\[ G^l_{β-DSP}(1^λ, n, l) : (\text{param, } \{ B(j), B^*(j) \}_{j=1}^{n^l}) \xleftarrow{R} G_{αβ}^l(1^λ, n + 3, l), \]
\[ \delta_1 \xleftarrow{U} \mathbb{F}_q, \quad (ω(k), γ_1(k), γ_2(k))_{k=1,2,3} \xleftarrow{U} GL(\mathbb{F}_q, 3), \]

For $j = 1, \ldots, n^l$;
\[ \mathbb{F}_q \ni \overrightarrow{v}_j = (v_1^{(j)}, \ldots, v_{n}^{(j)}) \xleftarrow{U} \mathbb{F}_q \setminus \{ \mathbf{0} \}, \]
\[ \mathbb{F}_q \ni \overrightarrow{d}_j = b_{1}^{(j)} + b_{1+2}^{(j)}, \quad \overrightarrow{B}_{j} = (b_1^{(j)}, \ldots, b_{n+2}^{(j)}), \]
\[ \mathbb{F}_q \ni \overrightarrow{e}_j = (e_0^{(j)}, \overrightarrow{e}_j^{(j)})_{j=1, \ldots, n^l}. \]

The l-IDSP instance generator $G^l_{β-IDSP}$ is defined as follows:

\[ G^l_{β-IDSP}(1^λ, n, l) : (\text{param, } \{ B(j), B^*(j) \}_{j=1}^{n^l}) \xleftarrow{R} G_{αβ}^l(1^λ, n + 3, l), \delta_1 \xleftarrow{U} \mathbb{F}_q, \]
\[ (ω(k), γ_1(k), γ_2(k))_{k=1,2,3} \xleftarrow{U} GL(\mathbb{F}_q, 3), \]

For $j = 1, \ldots, n^l$;
\[ \mathbb{F}_q \ni \overrightarrow{v}_j = (v_1^{(j)}, \ldots, v_{n}^{(j)}) \xleftarrow{U} \mathbb{F}_q \setminus \{ \mathbf{0} \}, \quad \mathbb{F}_q \ni \overrightarrow{d}_j = b_{1}^{(j)} + b_{1+2}^{(j)}, \quad \overrightarrow{B}_{j} = (b_1^{(j)}, \ldots, b_{n+2}^{(j)}), \]
\[ \mathbb{F}_q \ni \overrightarrow{e}_j = (e_0^{(j)}, \overrightarrow{e}_j^{(j)})_{j=1, \ldots, n^l}. \]

**Definition 3.5 (l-Decisional Subspace Problem with Relevant Dual Vector Tuples (l-RDSP)).**

For all security parameter $λ \in \mathbb{N}$, we define l-RDSP advantage of a probabilistic machine $B$ as follows:

\[ \text{Adv}^{l-RDSP}_{B}(λ) = | \Pr[B(1^λ, ρ) \to 1 | ρ \xleftarrow{U} G^l_{β-RDSP}(1^λ, n, l)] - \Pr[B(1^λ, ρ) \to 1 | ρ \xleftarrow{U} G^l_{αβ}-DSP(1^λ, n, l)] |. \]

The l-RDSP assumption is: for any probabilistic polynomial-time adversary $B$, $\text{Adv}^{l-RDSP}_{B}(λ)$ is negligible in $λ$. 

Definition 3.6 (l-Decisional Subspace Problem with Irrelevant Dual Vector Tuples (l-IDSP)). The l-IDSP advantage of \( B \), \( \text{Adv}_B^{l-IDSP}(\lambda) \), and the l-IDSP assumption are defined similarly as in Definition 3.3.

Remark 3.2. Here we explain the intuition of l-RDSP by using \( n' \) parallel instances of simplified RDSP. For \( j = 1, \ldots, n' \), the simplified l-RDSP assumption is that given \( (\mathbb{H}(j), \overrightarrow{v}_j, \{h_i^{(j)}\}_{i=1}^{n}, e_{\beta}^{(j)}) \), where \( \mathbb{H}(j) = (b_1^{(j)}, \ldots, b_{n+1}^{(j)}) \), \( \overrightarrow{v}_j = (v_1^{(j)}, \ldots, v_n^{(j)}) \), \( h_i^{(j)} = \omega b_i^{(j)} + v_i^{(j)}b_{n+1}^{(j)} \), \( i = 1, \ldots, n; \omega \leftarrow F_q \), and \( e_{\beta}^{(j)} = \delta_i (\sum_{i=1}^n v_i^{(j)}b_i^{(j)}) + \beta \zeta^{(j)}b_{n+1}^{(j)} + \delta_2 b_{n+2}^{(j)}(\beta \leftarrow \{0,1\}, \delta_1, \delta_2, \zeta^{(j)} \leftarrow F_q) \), it is hard for an adversary \( A \) to correctly guess \( \beta \). \( \{h_i^{(j)}\}_{i=1}^{n} \) is used to simulate the key generation in the security proof: for predicate \( l-\text{IDSP} \) assumption, since a secret key \( k \) to identify \( \beta \), except the following procedures: Game 1 is conceptually changed from Game 0. Two games are exactly the same except the following procedures:

1. When challenger \( \mathcal{C} \) gets challenge vectors \( Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_l^{(0)}) \) and \( Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_l^{(1)}) \) at the beginning of the game, \( \mathcal{C} \) selects random bit \( b \in \{0,1\} \), and computes

Theorem 3.2. Scheme 2 described in Section 3.6 is selectively attribute-hiding against chosen plaintext attacks under the l-RDSP and l-IDSP assumptions. For any adversary \( A \), there exist probabilistic machines \( B_1 \) and \( B_2 \), whose running times are essentially the same as that of \( A \), such that for any security parameter \( \lambda \),

\[
\text{Adv}_A(\lambda) \leq \text{Adv}_{B_1}^{l-RDSP}(\lambda) + \text{Adv}_{B_2}^{l-IDSP}(\lambda).
\]

Proof of Theorem 3.2. To prove Theorem 3.2, we consider the following games:

Game 0. Let Game 0 denote the real selective security game defined in Definition 3.3.

Game 1. Game 1 is conceptually changed from Game 0. Two games are exactly the same except the following procedures:

1. When challenger \( \mathcal{C} \) gets challenge vectors \( Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_l^{(0)}) \) and \( Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_l^{(1)}) \) at the beginning of the game, \( \mathcal{C} \) selects random bit \( b \in \{0,1\} \), and computes
Let $\text{Adv}_A$ will prove in 3.6. Scheme 2

$\text{Game 3}$ is almost identical to $\text{Game 2}$, except in the way the target ciphertext $C$ is computed and returned to $\text{Adv}_A$. It is clear that $C$, $\text{Game 0}$, and $\text{Game 1}$ for all $i \in [1, n^{l-1}]$.

2. When $C$ gets challenge plaintexts $M^{(0)}$ and $M^{(1)}$ from adversary $A$, target ciphertext $C$ is computed and returned to $A$.

$$C = \left( c_0 = g_T^\zeta M^{(b)}, \quad \forall i \in [1, n^{l-1}]: \quad c_i = \left( \sum_{1 \leq j \leq n} y_{i,j}^+ \cdot b_j^{(i)} + \zeta_i a_{n+1}^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} \right) \right)$$

where $\delta_2^{(i)} \in \mathbb{F}_q$ for all $i \in [1, n^{l-1}]$, and $\zeta \in \mathbb{F}_q$.

**Game 2.** Game 2 is almost identical to Game 1, except in the way the target ciphertext $C$ is generated.

$$C = \left( c_0 = g_T^\zeta M^{(b)}, \forall i \in [1, n^{l-1}]: c_i = \left( \sum_{1 \leq j \leq n} y_{i,j}^+ \cdot b_j^{(i)} + \zeta_1^{(i)} b_{n+1}^{(i)} + \zeta_2^{(i)} b_{n+2}^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} \right) \right)$$

where $\zeta_1^{(i)}, \zeta_2^{(i)}, \delta_2^{(i)} \in \mathbb{F}_q$ for all $i \in [1, n^{l-1}]$.

**Game 3.** Game 3 is almost identical to Game 2, except in the way the target ciphertext $C$ is generated.

$$C = \left( c_0 = g_T^\zeta M^{(b)}, \forall i \in [1, n^{l-1}]: c_i = \left( \sum_{1 \leq j \leq n} u_{i,j}^{(i)} b_j^{(i)} + \zeta_1^{(i)} b_{n+1}^{(i)} + \zeta_2^{(i)} b_{n+2}^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} \right) \right)$$

where $\zeta_1^{(i)}, \zeta_2^{(i)}, \delta_2^{(i)} \in \mathbb{F}_q$ for all $i \in [1, n^{l-1}]$, and $\overrightarrow{u}_i = (u_i^{(i)}, \ldots, u_n^{(i)}) \in \mathbb{F}_q^n \setminus \{0\}$ for all $i \in [1, n^{l-1}]$.

Let $\text{Adv}_A^{(0)}(\lambda)$ be $\text{Adv}_A(\lambda)$ in Game 0, and $\text{Adv}_A^{(i)}(\lambda)$ $(i = 1, \ldots, 3)$ be the advantage of $\text{Adv}_A$ in Game $i$. It is clear that $\text{Adv}_A^{(0)}(\lambda) = \text{Adv}_A^{(1)}(\lambda)$, since it is conceptually changed. We will prove in Lemma 3.4 that $\text{Adv}_A^{(3)}(\lambda) = 0$. We will show two lemmas (Lemmas 3.2
which evaluate the gaps between pairs of $\text{Adv}_A^{(i)}(\lambda)$ ($i = 1, 2, 3$). From these lemmas, we obtain $\text{Adv}_A(\lambda) = \text{Adv}_A^{(0)}(\lambda) = \text{Adv}_A^{(1)}(\lambda) \leq \sum_{i=1}^{3} \left| \text{Adv}_A^{(i)}(\lambda) - \text{Adv}_A^{(i+1)}(\lambda) \right| + \text{Adv}_A^{(3)}(\lambda) \leq \text{Adv}_B^{\text{R-DSP}}(\lambda) + \text{Adv}_B^{\text{L-DSP}}(\lambda).$

\[ \text{Lemma 3.2.} \quad \text{For any adversary } A, \text{ there exists a probabilistic machine } B_1, \text{ whose running time is the same as that of } A, \text{ such that for any security parameter } \lambda, \left| \text{Adv}_A^{(1)}(\lambda) - \text{Adv}_A^{(2)}(\lambda) \right| = \text{Adv}_B^{\text{R-DSP}}(\lambda). \]

\[ \text{Proof.} \quad \text{Suppose a polynomial time adversary } A \text{ can successfully distinguish between Game 1 and Game 2. We construct a simulator } B_1 \text{ that leverages } A \text{ to break the } l\text{-RDSP assumption. The procedure is shown as follows:} \]

1. $B_1$ is given $(\text{param}, \{\tilde{h}^{(i)}_j, b^{(i)}_j, e^{(i)}_{\beta_j} \}_{i=1, \ldots, n; \ j=1, 2, 3; \ \beta_j = 0, 1})$ which is a $l\text{-RDSP instance. ( } B_1 \text{ is a challenger in the security game against adversary } A).$

2. When challenger $B_1$ gets challenge vectors $Y^{(0)} = (\bar{y}^{(0)}_1, \ldots, \bar{y}^{(0)}_l)$ and $Y^{(1)} = (\bar{y}^{(1)}_1, \ldots, \bar{y}^{(1)}_l)$ at the beginning of the game, $B_1$ selects random bit $b \in \{0, 1\}$, and computes

\[
\begin{align*}
\bar{y}^+_1 &= (y^+_1, \ldots, y^+_n) = (\delta_1y_1^{(1)}(b), \ldots, y_l^{(1)}(b), \ldots, y_1(1)(b), \ldots, y_l^{(1)}(b)) \\
\vdots &= \vdots \\
\bar{y}^+_i &= (y^+_i, \ldots, y^+_n) = (\delta_1y_l^{(1)}(b), \ldots, y_i^{(1)}(b), \ldots, y_1(1)(b), \ldots, y_l^{(1)}(b)) \\
\vdots &= \vdots \\
\bar{y}^+_n &= (y^+_n, \ldots, y^+_n) = (\delta_1y_n^{(1)}(b), \ldots, y_n^{(1)}(b), \ldots, y_1^{(1)}(b), \ldots, y_l^{(1)}(b))
\end{align*}
\]

where $\delta_i \in \mathbb{F}_q$, $i_2 + (i_3 - 1) \cdot n + \ldots + (i_t - 1) \cdot n^{t-2} = i$, $i \in [1, n^{t-1}]$, and $i_2, \ldots, i_t \in [1, n]$.

Let $\pi^{(k)}_{i,j} \overset{U}{\leftarrow} \{\Pi^{(k)} \in \text{GL}(n, \mathbb{F}_q) \mid \bar{v}_k = \bar{y}_k^+ \cdot \Pi^{(k)} \cdot \Pi^{T(k)} = \Pi^{(k)}\}$, and $\Pi^*(k) = (\pi^{*}_{i,j}^{(k)})^{-1}$ for all $k \in [1, n^{t-1}]$. Note that $\bar{y}_k^+ = \bar{v}_k \cdot \Pi^*(k)$ for all $k \in [1, n^{t-1}]$. $B_1$ computes the public key $PK$ as follows and gives it to $A$:

\[
PK = (1^\lambda, \text{param}, \tilde{h})
\]

\[
\begin{align*}
\left\{ b^{(k)}_j = \sum_{\theta=1}^{n} \pi^{(k)}_{j,\theta} b^{(k)}_{\theta}, \quad b^*_j = \sum_{\theta=1}^{n} \pi^{*}_{j,\theta} b^*_{\theta} \right\}_{j=1, ..., n}^{k=1, ..., n^{t-1}} \\
\tilde{h} = \left\{ \tilde{b}^{(k)}_1, \ldots, \tilde{b}^{(k)}_n, d^{(k)}_{n+1}, b^{(k)}_{n+3} \right\}_{k=1, ..., n^{t-1}}
\end{align*}
\]
3. When $A$ makes a query for a predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$, $B_1$ answers as follows: $B_1$ computes

\[
\overrightarrow{x}_1^+ = (x_{1,1}, \ldots, x_{1,n}) = (\sigma x_1^{(1)} x_1^{(2)} \ldots x_1^{(l)}), \ldots, \sigma x_n^{(1)} x_1^{(2)} \ldots x_1^{(l)})
\]

\[
\vdots
\]

\[
\overrightarrow{x}_l^+ = (x_{l,1}, \ldots, x_{l,n}) = (\sigma x_1^{(1)} x_{l+1}^{(2)} \ldots x_{l+1}^{(l)})
\]

\[
\vdots
\]

\[
\overrightarrow{x}_{l-1}^+ = (x_{l-1,1}, \ldots, x_{l-1,n}) = (\sigma x_1^{(1)} x_{l+2}^{(2)} \ldots x_{l+2}^{(l)})
\]

where $\sigma \in \mathbb{F}_q$, $i_2 + (i_3 - 1) \cdot n + \ldots + (i_l - 1) \cdot n^{l-2} = i$, $i \in [1, n^{l-1}]$, and $i_2, \ldots, i_l \in [1, n]$. Then, $B_1$ calculates and returns $k^*$ using $\{|h^{(k^*)_j}_q^{(j)}|_{i=1, \ldots, n}; k=1,2,3\}_j=1,\ldots,n^{l-1}$ in the $(l-1)$-RDSP instance:

\[
\theta = \sum_{k=1}^{3} a_k \sum_{j=1}^{n^{l-1}} \sum_{i=1}^{n} x_{j,i}^+ \sum_{q=1}^{n} \pi_i^{(j)} r_q^{(k)}
\]

\[
k^* = \left( \forall j \in [1, n^{l-1}] : k_j = \theta^{-1} \sum_{k=1}^{3} a_k \sum_{i=1}^{n} x_{j,i}^+ \sum_{q=1}^{n} \pi_i^{(j)} h_q^{(k)} \right)
\]

where $a_k \in \mathbb{F}_q$ for $k = 1, 2, 3$. If $\theta = 0, a_1, a_2, a_3 \in \mathbb{F}_q$ is selected again.

4. When $B_1$ receives challenge plaintexts $(M^{(0)}, M^{(1)})$ from $A$, $B_1$ computes and returns $C$ s.t. $e_0 = \zeta M^{(0)}$, and $e_j = \epsilon_j^{(j)} + \epsilon d_j^{(j)}$ for $j = 1, \ldots, n^{l-1}$ using $e^{(j)}_\beta$ in the $(l-1)$-RDSP instance, $\zeta$, and $M^{(0)}$, where $\zeta \in \mathbb{F}_q$.

5. After the challenge phase, GenKey oracle simulation for a reveal key query is executed as above.

6. $A$ outputs bit $b'$. If $b = b'$, $B_1$ outputs $\beta = 1$. Otherwise, $B_1$ outputs $\beta = 0$.

To prove Lemma 3.2, we show Claims 3.1, 3.2, and 3.3.

**Claim 3.1.** Public key $PK$ generated in the step 2 above has the same distribution as that in Game 1 and Game 2.

**Proof.** For $j = 1, \ldots, n^{l-1}$, let $D^{(j)} = \begin{pmatrix} \Pi^{(j)} & 0 \\ 0 & I_3 \end{pmatrix}$ be $(n+3) \times (n+3)$ matrix that consists of $\Pi^{(j)}$ and the identity matrix $I_3$. Since the basis $(\tilde{b}_1^{(j)}, \ldots, \tilde{b}_{n+1}^{(j)}, \tilde{b}_{n+2}^{(j)}, \tilde{b}_{n+3}^{(j)})$ of $\mathbb{V}$ is obtained from basis $B$ by the linear transformation determined by $D^{(j)}$, it is uniformly distributed. Hence, $\tilde{B} = (\tilde{b}_1^{(j)}, \ldots, \tilde{b}_{n+1}^{(j)}, \tilde{b}_{n+2}^{(j)}, \tilde{b}_{n+3}^{(j)})$ in step 2 has the same distribution as that in Game 1 and Game 2. □

**Claim 3.2.** Secret key generated in steps 3 and 5 above has the same distribution as that in Game 1 and Game 2.
Proof. For \( j = 1, \ldots, n^{l-1} \), it can be verified that basis \( \tilde{b}_1^{(j)}, \ldots, \tilde{b}_n^{(j)}, b_{n+1}^{(j)}, b_{n+2}^{(j)}, b_{n+3}^{(j)} \) of \( \mathcal{V} \) is obtained by the linear transformation \((D^{(j)})^{-1}\), where \( D^{(j)} \) is defined in the proof of Claim 3.3. Therefore, it is dual orthonormal to basis \(( \tilde{b}_1^{(j)}, \ldots, \tilde{b}_n^{(j)}, b_{n+1}^{(j)}, b_{n+2}^{(j)}, b_{n+3}^{(j)} )\). We can compute the secret key using this dual orthonormal basis.

The secret key \( k^* \) generated in the steps 3 and 5 is \( \forall j \in [1, n^{l-1}]\), \( k_j = \theta^{-1}\left( \sum_{k=1}^{3} a_k \omega^{(k)} \right) \)

\[
\sum_{i=1}^{n} x_{j,i} \tilde{b}_i^{(j)} + \theta^{-1} \theta_1^{(j)} b_{n+1}^{(j)} + \theta^{-1} \theta_2^{(j)} b_{n+2}^{(j)}, \text{ where } \theta_1^{(j)} = \left( \sum_{k=1}^{3} a_k \gamma_1^{(k)} \right) \tilde{x}_{j}^{+} \cdot \tilde{y}_{j}^{+}, \theta_2^{(j)} = \left( \sum_{k=1}^{3} a_k \gamma_2^{(k)} \right) \tilde{x}_{j}^{+} \cdot \tilde{y}_{j}^{+} \text{, and } \theta = \sum_{j=1}^{n^{l-1}} (\theta_1^{(j)} + \theta_2^{(j)}) \text{, Let } \sigma = \theta^{-1}\left( \sum_{k=1}^{3} a_k \omega^{(k)} \right) \text{, Then, } \sigma, \theta_1^{(j)}, \theta_2^{(j)} \text{ are independently uniform, because } a_k \text{ are independently uniform. The sum of the coefficients of } b_{n+1}^{(j)} \text{ and } b_{n+2}^{(j)} \text{ is 1, i.e., } \sum_{j=1}^{n^{l-1}} (\theta^{-1} \theta_1^{(j)} + \theta^{-1} \theta_2^{(j)}) = 1 \text{. We establish that the secret key } k^* \text{ has the same distribution as in Game 1 and Game 2.}
\]

Claim 3.3. When \( \beta = 0 \) the challenge ciphertext \( C \) generated in step 4 is distributed exactly as in Game 1, whereas if \( \beta = 1 \), the challenge ciphertext \( C \) generated in step 4 is identical as distributed in Game 2.

Proof. If \( \beta = 0 \), \( c_0 = g_2^{\theta} M(b) \), and \( \forall i \in [1, n^{l-1}] \) : \( c_i = \delta_1 \left( \sum_{1 \leq j \leq n} v_j^{(i)} \cdot b_j^{(i)} \right) + \zeta d_1^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} = \delta_1 \left( \sum_{1 \leq j \leq n} v_{i,j}^{+} \cdot \tilde{b}_j^{(i)} \right) + \zeta d_1^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} \). This is the challenge ciphertext in Game 1 with \( PK = (1^3, \text{param}, \tilde{b}) \).

If \( \beta = 1 \), \( c_0 = g_2^{\theta} M(b) \), and \( \forall i \in [1, n^{l-1}] \) : \( c_i = \delta_1 \left( \sum_{1 \leq j \leq n} v_{i,j}^{+} \cdot \tilde{b}_j^{(i)} \right) + (\zeta + \delta_1^{(i)}) b_{n+1}^{(i)} + (\zeta + \delta_2^{(i)}) b_{n+2}^{(i)} + \delta_2^{(i)} b_{n+3}^{(i)} \). This is the challenge ciphertext in Game 2 with \( PK = (1^3, \text{param}, \tilde{b}) \).

From Claims 3.1, 3.2, and 3.3 when \( \beta = 0 \), the above simulation is distributed exactly as in Game 1, whereas if \( \beta = 1 \) the above simulation is identically distributed as in Game 2. Therefore, \( | \text{Adv}_{A}^{1}(\lambda) - \text{Adv}_{A}^{2}(\lambda) | = | \text{Pr}[B_1(1^{3}, \rho) \to 1 | \rho \xleftarrow{R} G_0^{\text{RDSP}}(1^{3}, n, l)] - \text{Pr}[B_2(1^{3}, \rho) \to 1 | \rho \xleftarrow{R} G_1^{\text{RDSP}}(1^{3}, n, l)] | = \text{Adv}_{B_1}^{1-\text{RDSP}}(\lambda) \), which completes our proof of Lemma 3.2.

Lemma 3.3. For any adversary \( A \), there exists a probabilistic machine \( B_2 \), whose running time is the same as that of \( A \), such that for any security parameter \( \lambda \), \( | \text{Adv}_{A}^{2}(\lambda) - \text{Adv}_{A}^{3}(\lambda) | = \text{Adv}_{B_2}^{1-\text{IDSP}}(\lambda) \).

Proof. The proof of Lemma 3.3 is similar to that of Lemma 3.2 we omit it here.

Lemma 3.4. For any adversary \( A \), \( \text{Adv}_{A}^{3}(\lambda) = 0 \).

Proof. The adversary \( A \)’s output \( b’ \) is independent of the hidden bit \( b \) in Game 3. Therefore, \( \text{Adv}_{A}^{3}(\lambda) = 0 \).
3.7 Applications of Our Schemes

In this section, we show how to leverage our proposed schemes to achieve systems evaluating disjunctive comparison or disjunctive subset queries. Firstly, we introduce the related predicates including the predicates corresponding to hidden vector encryption (HVE) [15], disjunctive comparison and disjunctive subset queries, then we combine HVE with our schemes to realize the disjunctive predicate evaluations.

Hidden-Vector Encryption. Let \( \Sigma_* = \Sigma \cup \{\ast\} \), where \( \Sigma \) is a set. Hidden vector encryption (HVE) corresponds to a predicate encryption scheme for the class of predicates \( \psi_{HVE} = \{ \phi_{HVE}^{a_1, \ldots, a_l} \mid a_1, \ldots, a_l \in \Sigma_* \} \), where
\[
\phi_{HVE}^{a_1, \ldots, a_l}(x_1, \ldots, x_l) = \begin{cases} 
1, & \text{if either } a_i = x_i \text{ or } a_i = \ast, \text{ for all } i = 1, \ldots, \tilde{l}, \\
0, & \text{otherwise}.
\end{cases}
\]

Disjunctive Comparison Predicates. Let a set \( \Sigma = \{1, \ldots, \tilde{N}\}^{\tilde{l}} \). The disjunctive comparison predicates are the class of predicates \( \Phi_{\tilde{l}} = \{ P_{(a_1, \ldots, a_l)} \mid (a_1, \ldots, a_l) \in \Sigma \} \), where
\[
P_{(a_1, \ldots, a_l)}(x_1, \ldots, x_l) = \begin{cases} 
1, & \text{if } x_1 \geq a_1 \text{ or } x_2 \geq a_2 \text{ or } \ldots \text{ or } x_l \geq a_l, \\
0, & \text{otherwise}.
\end{cases}
\]

Disjunctive Subset Predicates. Let \( T \) be a set of size \( \tilde{N} \), \( \sigma = (\tilde{A}_1, \ldots, \tilde{A}_{\tilde{l}}) \), where \( \tilde{A}_i \subseteq T \) for all \( i = 1, \ldots, \tilde{l} \). The disjunctive subset predicates are the class of predicates \( \Upsilon_{\tilde{l}} = \{ P'_{(\tilde{A}_1, \ldots, \tilde{A}_{\tilde{l}})} \mid \tilde{A}_1, \ldots, \tilde{A}_{\tilde{l}} \subseteq T \} \), where
\[
P'_{(\tilde{A}_1, \ldots, \tilde{A}_{\tilde{l}})}(b_1, \ldots, b_{\tilde{l}}) = \begin{cases} 
1, & \text{if } b_1 \in \tilde{A}_1 \text{ or } b_2 \in \tilde{A}_2 \text{ or } \ldots \text{ or } b_{\tilde{l}} \in \tilde{A}_{\tilde{l}}, \\
0, & \text{otherwise}.
\end{cases}
\]

3.7.1 The System Evaluating Disjunctive Comparison Predicates

We show how to leverage our PEM to construct a system for disjunctive comparison queries.

Let \((\text{Setup}_{PEM}, \text{GenKey}_{PEM}, \text{Encrypt}_{PEM}, \text{Decrypt}_{PEM})\) be a secure PEM.

- Setup is the same as \( \text{Setup}_{PEM} \).
- GenKey\((MSK, \vec{a})\), where \( \vec{a} = (a_1, \ldots, a_{\tilde{l}}) \in \{1, \ldots, \tilde{N}\}^{\tilde{l}} \). First construct vectors corresponding to HVE encryption. Define \( \sigma_\ast(a_i) = (\sigma_{i,j}) \in \{\ast, 1\}^{\tilde{N}} \) for \( i = 1, \ldots, \tilde{l} \) as follows:
\[
\sigma_{i,j} = \begin{cases} 
1 & \text{if } a_i = j \\
\ast & \text{otherwise}
\end{cases}
\]
Chapter 3. Predicate Encryption for Multi Inner Product

Then transform \((\sigma_{i,1}, \ldots, \sigma_{i,\tilde{N}})\) to vectors \(\overrightarrow{A}_i = (A_{i,1}, \ldots, A_{i,2\tilde{N}})\) for \(i = 1, \ldots, \tilde{t}\) corresponding to PEM as follows:

\[
\begin{align*}
\text{if } \sigma_{i,j} \neq *: & \quad A_{i,2j-1} = 1, \quad A_{i,2j} = \sigma_{i,j} \\
\text{if } \sigma_{i,j} = *: & \quad A_{i,2j-1} = 0, \quad A_{i,2j} = 0
\end{align*}
\]

Output the key by running \(k^* \leftarrow \text{GenKey}_{PEM}(MSK, (\overrightarrow{A}_1, \ldots, \overrightarrow{A}_{\tilde{t}}))\).

For example, let \(\tilde{t} = 2\) and \(\overrightarrow{a} = (a_1, a_2)\), the vectors \(\sigma_s(a_1)\) and \(\sigma_s(a_2)\) for the HVE encryption are as follows:

\[
\begin{align*}
\sigma_s(a_1) &= \begin{pmatrix} 1 & * & * & \cdots & * & \cdots & * \end{pmatrix} \\
\sigma_s(a_2) &= \begin{pmatrix} 1 & * & * & \cdots & * & \cdots & * \end{pmatrix}
\end{align*}
\]

Next construct the vectors \(\overrightarrow{A}_1\) and \(\overrightarrow{A}_2\) for PEM from \(\sigma_s(a_1)\) and \(\sigma_s(a_2)\) respectively:

\[
\overrightarrow{A}_1 = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \quad 2a_1 \quad 2\tilde{N} \quad \overrightarrow{A}_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \quad 2a_2 \quad 2\tilde{N}
\]

- **Encrypt** \((PK, M, \overrightarrow{b})\), where \(\overrightarrow{b} = (b_1, \ldots, b_\tilde{N})\). To encrypt a message \(M\) for the attribute \(\overrightarrow{b} = (b_1, \ldots, b_\tilde{N})\), first construct the attribute vectors corresponding to HVE encryption. Build vectors \(\sigma(b_i) = (\sigma_{i,j}) \in \{0, 1\}^\tilde{N}\) for \(i = 1, \ldots, \tilde{t}\) as follows:

\[
\sigma_{i,j} = \begin{cases} 
1, & \text{if } j \geq b_i, \\
0, & \text{otherwise}.
\end{cases}
\]

Then convert \((\sigma_{i,1}, \ldots, \sigma_{i,\tilde{N}})\) to vectors \(\overrightarrow{B}_i = (B_{i,1}, \ldots, B_{i,2\tilde{N}})\) for \(i = 1, \ldots, \tilde{t}\) corresponding to PEM as follows: select random \(r_{i,1}, \ldots, r_{i,\tilde{N}} \in \mathbb{F}_q\) for \(i = 1, \ldots, \tilde{t}\), let \(B_{i,2j-1} = -r_{i,j} \cdot \sigma_{i,j}, \quad B_{i,2j} = r_{i,j}\). Finally, output the ciphertext \(C \leftarrow \text{Encrypt}_{PEM}(PK, M, (\overrightarrow{B}_1, \ldots, \overrightarrow{B}_{\tilde{t}}))\).

For example, let \(\tilde{t} = 2\) and \(\overrightarrow{b} = (b_1, b_2)\), the vectors \(\sigma(b_1)\) and \(\sigma(b_2)\) for the HVE encryption are as follows:

\[
\begin{align*}
\sigma(b_1) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \quad b_1 \quad 2\tilde{N} \\
\sigma(b_2) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \quad b_2 \quad 2\tilde{N}
\end{align*}
\]

Next select random \(r_{i,1}, \ldots, r_{i,\tilde{N}} \in \mathbb{F}_q\) for \(i = 1, 2\), and construct the vectors \(\overrightarrow{B}_1\) and \(\overrightarrow{B}_2\) for PEM from \(\sigma(b_1)\) and \(\sigma(b_2)\) respectively:

\[
\begin{align*}
\overrightarrow{B}_1 &= \begin{pmatrix} 1 & 0 & r_{1,1} & \cdots & -r_{1,b_1} & r_{1,b_1} & \cdots & -r_{1,\tilde{N}} & r_{1,\tilde{N}} \\
1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & 1 \\
2b_1 & 2\tilde{N}
\end{pmatrix} \\
\overrightarrow{B}_2 &= \begin{pmatrix} 1 & 0 & r_{2,1} & \cdots & -r_{2,b_2} & r_{2,b_2} & \cdots & -r_{2,\tilde{N}} & r_{2,\tilde{N}} \\
2b_2 & 2\tilde{N}
\end{pmatrix}
\end{align*}
\]
3.7. Applications of Our Schemes

- Decrypt($k^*_o, C$) output Decrypt_{PEM}($k^*_o, C$).

We follow the notions above. To verify the correctness of the system, we have:

$$P_{\tilde{\sigma}}(\tilde{b}) = 1 \iff \exists i : \phi_{\sigma(i)}^{HVE}(b_i) = 1 \iff \exists i : \vec{A}_i \vec{B}_i = 0 \iff f_{(\vec{B}_1, \ldots, \vec{B}_{\tilde{l}})}(\vec{B}_1, \ldots, \vec{B}_{\tilde{l}}) = 1$$

The system achieves both ciphertexts and user private keys of size $O((\tilde{N})^\tilde{l})$.

3.7.2 The System Evaluating Disjunctive Subset Predicates

The PEM can also be applied to a system for disjunctive subset queries.

Let $(\text{Setup}_{PEM}, \text{GenKey}_{PEM}, \text{Encrypt}_{PEM}, \text{Decrypt}_{PEM})$ be a secure PEM.

- **Setup** is the same as Setup_{PEM}.
- **GenKey**(MSK, $P^*_o$), where $\sigma = (\tilde{A}_1, \ldots, \tilde{A}_{\tilde{l}})$. First construct vectors corresponding to HVE encryption. Define $\sigma_*(\tilde{A}_i) = (\sigma_{i,j}) \in \{0, \star\}^{\tilde{N}}$ for $i = 1, \ldots, \tilde{l}$ as follows:

$$\sigma_{i,j} = \begin{cases} 0, & \text{if } j \notin \tilde{A}_i, \\ \star, & \text{otherwise.} \end{cases}$$

Then transform $(\sigma_{i,1}, \ldots, \sigma_{i,\tilde{N}})$ to vectors $\vec{A}_i = (A_{i,1}, \ldots, A_{i,\tilde{N}})$ for $i = 1, \ldots, \tilde{l}$ corresponding to PEM as follows:

$$\begin{array}{l}
\text{if } \sigma_{i,j} \neq \star: \quad A_{i,2j-1} = 1, \quad A_{i,2j} = \sigma_{i,j} \\
\text{if } \sigma_{i,j} = \star: \quad A_{i,2j-1} = 0, \quad A_{i,2j} = 0
\end{array}$$

Output the key by running $k^*_o \leftarrow \text{GenKey}_{PEM}(\text{MSK}, (\vec{A}_1, \ldots, \vec{A}_{\tilde{l}}))$.

For example, let $\tilde{l} = 2$ and $\sigma = (\tilde{A}_1, \tilde{A}_2)$, where $\tilde{A}_1 = \{2, \tilde{N}\} \subseteq T$ and $\tilde{A}_2 = \{1, 2, 3\} \subseteq T$, the vectors $\sigma_*(\tilde{A}_1)$ and $\sigma_*(\tilde{A}_2)$ for the HVE encryption are as follows:

$$\sigma_*(\tilde{A}_1) = \begin{bmatrix} 1 & 2 & 3 & 4 & \tilde{N} \\ 0 & * & 0 & 0 & \ldots & 0 & * \end{bmatrix} \quad \sigma_*(\tilde{A}_2) = \begin{bmatrix} 1 & 2 & 3 & 4 & \tilde{N} \\ * & * & * & 0 & 0 & \ldots & 0 \end{bmatrix}$$

Next construct the vectors $\vec{A}_1$ and $\vec{A}_2$ for PEM from $\sigma_*(\tilde{A}_1)$ and $\sigma_*(\tilde{A}_2)$ respectively:

$$\vec{A}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \quad 2\tilde{N} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \quad \vec{A}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}$$

- **Encrypt**(PK, $M$, $\vec{b}$), where $\vec{b} = (b_1, \ldots, b_{\tilde{l}}) \in T^\tilde{l}$. To encrypt a message $M$ for the attribute $\vec{b} = (b_1, \ldots, b_{\tilde{l}}) \in T^\tilde{l}$, first construct the attribute vectors corresponding to HVE encryption. Build vectors $\sigma(b_i) = (\sigma_{i,j}) \in \{0, 1\}^{\tilde{N}}$ for $i = 1, \ldots, \tilde{l}$ as follows:
Then convert \((\sigma_{i,1}, \ldots, \sigma_{i,\tilde{N}})\) to vectors \(\vec{B}_i = (B_{i,1}, \ldots, B_{i,\tilde{N}})\) for \(i = 1, \ldots, \tilde{l}\) corresponding to PEM as follows: select random \(r_{i,1}, \ldots, r_{i,\tilde{N}} \in \mathbb{F}_q\) for \(i = 1, \ldots, \tilde{l}\), let \(B_{i,2j-1} = -r_{i,j} \cdot \sigma_{i,j}, B_{i,2j} = r_{i,j}\). Finally, output the ciphertext \(C \leftarrow \text{Encrypt}_{PEM}(PK, M, (\vec{B}_1, \ldots, \vec{B}_{\tilde{l}}))\).

For example, let \(\tilde{l} = 2\) and \(\vec{b} = (b_1, b_2)\), the vectors \(\sigma(b_1)\) and \(\sigma(b_2)\) for the HVE encryption are as follows:

\[
\sigma(b_1) = \begin{bmatrix} 1 & b_1 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix} \quad \sigma(b_2) = \begin{bmatrix} 1 & b_2 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}
\]

Next select random \(r_{1,1}, \ldots, r_{1,\tilde{N}} \in \mathbb{F}_q\) for \(i = 1, 2\), and construct the vectors \(\vec{B}_1\) and \(\vec{B}_2\) for PEM from \(\sigma(b_1)\) and \(\sigma(b_2)\) respectively:

\[
\vec{B}_1 = \begin{bmatrix} 1 & 2b_1 & 2\tilde{N} \\ 0 & r_{1,1} & \ldots & -r_{1,b_1} & r_{1,b_1} & \ldots & 0 & r_{1,\tilde{N}} \end{bmatrix} \\
\vec{B}_2 = \begin{bmatrix} 1 & 2b_2 & 2\tilde{N} \\ 0 & r_{2,1} & \ldots & -r_{2,b_2} & r_{2,b_2} & \ldots & 0 & r_{2,\tilde{N}} \end{bmatrix}
\]

- Decrypt\((k^*_\sigma, C)\) output Decrypt\(_{PEM}(k^*_\sigma, C)\).

We follow the notions above. To verify the correctness of the system, we have:

\[
P'_{\vec{b}} = 1 \iff \exists i: \sigma_{\sigma_i(A_i)}(b_i) = 1 \iff \exists i: \tilde{A}_i \vec{B}_i = 0 \iff f(A_1, \ldots, A_{\tilde{l}}, (\tilde{B}_1, \ldots, \tilde{B}_{\tilde{l}})) = 1
\]

The system achieves both ciphertexts and user private keys of size \(O((\tilde{N})^{\tilde{l}})\).

### 3.8 Arbitrary Combinations of Conjunctive and Disjunctive Predicates

Our techniques can be applied to support richer sets of predicates. More specifically, we can construct systems for arbitrary combinations of conjunctive and disjunctive predicate evaluations. For example, in a payroll system, a supervisor may want to retrieve the staff who satisfy some specific conditions: \((\text{Age} > 30 \lor (\text{Position} = \text{Senior Engineer} \land \text{Wage Scale} \in \{4, 5, 8\}))\). We let \(\vec{x}_1\) associate with the predicate “\(\text{Age} > 30\)”, \(\vec{x}_2\) associate with “\(\text{Position} = \text{Senior Engineer}\)”, and \(\vec{x}_3\) associate with “\(\text{Wage Scale} \in \{4, 5, 8\}\)”. \(\vec{y}_1, \vec{y}_2,\) and \(\vec{y}_3\) are the corresponding attributes in a ciphertext. To evaluate the predicate, we need to construct a system so that the Decrypt algorithm will output \(g^{D_1(rD_2+D_3)}\), where \(r\) is randomly selected, \(D_1 = \vec{x}_1 \vec{y}_1, D_2 = \vec{x}_2 \vec{y}_2,\) and \(D_3 = \vec{x}_3 \vec{y}_3\). We see that the above predicate
3.8. Arbitrary Combinations of Conjunctive and Disjunctive Predicates

evaluates to 1 iff \((\text{Age} > 30 \lor (\text{Position} = \text{Senior Engineer} \land \text{Wage Scale} \in \{4, 5, 8\})\)\), which is equivalent to say \((D_1 = 0 \lor (D_2 = 0 \land D_3 = 0))\). Hence, if the predicate evaluates to 1, we know that \(D_1 \cdot (r D_2 + D_3) = 0\). The system can be constructed with the similar technique for the disjunctive predicates specified in Section 3.4. For the formulae \(\vec{x}_1 \vec{y}_1 \cdot (r \vec{x}_2 \vec{y}_2 + \vec{x}_3 \vec{y}_3)\), we rephrase it and distill the attribute vector and predicate vector for the ciphertext and secret key respectively as we did before. We note that the sizes of the ciphertext and secret key depend on the number of disjunctive operators in the predicates.

In above example, there is one disjunctive operator and the sizes of the ciphertext and secret key are \(O(N^2)\) (the exponent is the number of disjunctive operators plus 1), where \(N\) is the length of the maximum vector among \(\vec{x}_1, \vec{x}_2\), and \(\vec{x}_3\). More generally, given a predicate of conjunction and disjunction, we can always convert it into either conjunctive normal form (CNF)

\[
\bigwedge_{i=1}^{n} \left( \bigvee_{j=1}^{m_i} s_{i,j} \right)
\]

or disjunctive normal form (DNF)

\[
\bigvee_{i=1}^{n'} \left( \bigwedge_{j=1}^{m'_i} s_{i,j} \right)
\]

where \(s_{i,j}\) is one of the equality, comparison and subset predicate for all \(i, j\). In CNF, the sizes of the ciphertext and secret key are \(O(N^{l_{\max}})\), where \(l_{\max} = \max\{m_1, \ldots, m_n\}\). Meanwhile, if the predicate formula is in DNF, the corresponding sizes will be \(O(N^{n'})\). Minimizing the number of disjunctions plays an important role in reducing the sizes of the ciphertext and secret key in our system. Hence, when given a predicate, firstly we convert it into CNF and DNF. If \(l_{\max} \leq n'\), we use the predicate in CNF to construct the system, otherwise we encode the predicate in DNF.

Next, we introduce a method to reduce disjunctions. Smart [59] gave a heuristic method to reduce the disjunctions in a CDNF (or conjunctive-DNF). We implement an algorithm to reduce the disjunctions in a DCNF (or disjunctive-CNF). A DCNF is an expression of the form:

\[
\bigvee_{i=1}^{n} \left( \bigwedge_{j=1}^{m_i} \left( \bigvee_{k=1}^{m_{i,j}} s_{i,j,k} \right) \right)
\]

where \(s_{i,j,k}\) is one of the equality, comparison and subset predicate for all \(i, j, k\). The DCNF is flexible. When \(n = 1\), we obtain CNF, and when \(m_{i,j} = 1\) for all \(i, j\), we obtain DNF.

We first describe our algorithm, then present an instance for our specific predicates.

- Set \(m_{i,j} = 1\) and express the value of the DCNF in disjunctive normal form as

\[
C = \bigvee_{i=1}^{n} \left( \bigwedge_{j=1}^{m_i} t_{i,j} \right)
\]

where

\[
t_{i,j} = \bigvee_{k=1}^{m_{i,j}} s_{i,j,k}
\]
Chapter 3. Predicate Encryption for Multi Inner Product

If
\[ \bigwedge_{j \neq j_1} t_{i_1,j} = \bigwedge_{j \neq j_2} t_{i_2,j} \text{ for } i_1 \neq i_2 \]
we set \[ C = \left[ \bigvee_{i \neq i_1, i_2} \left( \bigwedge_{j=1}^{m} t_{i,j} \right) \right] \bigvee \left[ \left( \bigwedge_{j \neq j_1} t_{i_1,j} \right) \bigwedge (t_{i_1,j_1} \lor t_{i_2,j_2}) \right] \]

We note that \( t_{i,j} \) is specified as a equality predicate in Smart’s algorithm \[59\]. However, it can be extended to represent other predicates such as comparison or subset predicates. For example: we denote \( a \) as \textbf{Age} > 30, \( b \) as \textbf{Position} = Senior Engineer, \( c \) as \textbf{Wage Scale} \( \in \{4, 5, 8\} \), and \( d \) as \textbf{Sex} = Male. The following predicate can be simplified with our proposed algorithm.

\[
(a \land c) \lor (a \land d) \lor (b \land c) \lor (b \land d) = \left[ (a \land c) \lor (a \land d) \right] \lor \left[ b \land (c \lor d) \right] = \left[ a \land (c \lor d) \right] \lor \left[ b \land (c \lor d) \right] = (a \lor b) \land (c \lor d)
\]

In this case, we reduce the disjunctive operators.

### 3.9 Achieving Adaptive and CCA security

In our systems, the adversary must commit ahead of time to the target attribute vectors, which is called selective security. A stronger notion of security can be defined by modifying the game in \textbf{Definition 3.4} so that the adversary outputs the target attribute vectors in the challenge phase. Recently, Lewko \textit{et al} \[35\] proposed an adaptive secure predicate encryption system. By applying our technique to their system, we can construct a full secure encryption system supporting arbitrary combinations of conjunctive and disjunctive predicates. On the other hand, our system can be extended to hierarchical settings. Hence, it is also possible to achieve CCA secure system by employing a generic conversion such as that by Canetti, Halevi and Katz \[18\].

### 3.10 Conclusion

In this chapter, we study the encryption systems supporting a wide class of predicate evaluations. More specifically, we propose schemes for multi inner-product predicates, which can be leveraged to achieve disjunctions of comparison and disjunctions of subset queries. Furthermore, we show that our systems can also be used to realize arbitrary conjunctive and disjunctive queries. Finally, we propose some possible methods to improve the performance of our systems. While expressiveness is an important issue in PE, there are other open problems in PE, e.g., key revocation in PE. In the next chapter, we will introduce a revocable PE with attribute hiding and provide the security proof for the proposed scheme.
Chapter 4

Revocable Predicate Encryption with Attribute Hiding

In this chapter, we investigate revocable predicate encryption (RPE) with attribute hiding. We provide syntax and security definitions of RPE. We then propose an efficient RPE with constant-size private and public keys, and the length of its ciphertexts is linear in the number of revoked keys. We also prove that our scheme is secure under the Decision Linear assumption in the standard model.

4.1 Introduction

Revocation in Functional Encryption The revocation challenge in FE schemes turns out to be more subtle than in previous encryption paradigms, e.g. in comparison to CRL-based revocation mechanisms used in traditional PKE schemes (within public key infrastructures) or to the time-based revocation approach suggested by Boneh and Franklin for IBE schemes, where the identities of receivers are linked to time periods and unrevoked users must be in possession of up-to-date private keys, obtained from the Private Key Generator (PKG). The revocation problem in FE is apparent in that FE ciphertexts are encrypted for predicates $f$ that can possibly be satisfied by multiple recipients, all in possession of suitable attributes $a$. Using time-based revocation for users’ attributes is inappropriate here for several reasons: First, a user may be in possession of several attributes and if time periods for all attributes in the system are not synchronized then unrevoked users would have to update their private keys whenever any of their attributes expires. Note that due to the necessary collusion-resistance property of FE schemes a user’s private key must depend on all attributes of that user. Second, even if time periods are synchronized then the problem with scalability still remains. Indeed, the PKG would have to be regularly contacted by all unrevoked users in the system to obtain updates for their private keys. This would require online presence of the PKG, establishment of secure channels between the PKG and each user for the transmission
Chapter 4. Revocable Predicate Encryption with Attribute Hiding

of updated private keys, and authentication of users towards the PKG to prove eligibility with regard to the update procedure. The amount of work performed by the PKG is then linear in the number of (unrevoked) users and attributes available in the system. A more efficient approach for handling revocation in IBE systems was suggested by Boldyreva, Goyal, and Kumar (BGK) [8], where the PKG on each time period publishes some update information that is then used by unrevoked users to update their private keys locally. The amount of work performed by PKG is logarithmic and, more importantly, no online communication between the PKG and unrevoked users is required. The BGK approach [8] could also be applied to ABE systems, in which case, however, it would result in a significant limitation — while in IBE systems revoking user identities is sufficient, revoking attributes in ABE systems would implicitly revoke private keys of all users with those attributes. That is revocation of users (which is possible with the time-based approach of Boneh and Franklin [12] when applied to ABE systems) would no longer be possible with the BGK approach. Another limitation of the BGK approach is that unrevoked users still have to update their private keys for each time period.

To alleviate this limitation, Attrapadung and Imai [3] suggested another way for revocation in ABE schemes: Instead of enforcing revocation via an authority, the revocation is carried out by the senders directly, i.e., the senders encrypt a message under a normal attribute set, as well as a revocation list. Each user’s private key has an associated policy and some unique identifier. A private key can be used to decrypt the ciphertext if the attributes in the ciphertext satisfy the policy associated with the key and the identifier of the key is not contained in the revocation list encoded into the ciphertext. This method solves the mentioned problem behind the BGK approach, namely each user’s private key can now be issued by PKG once and need not be updated thereafter. Later on, Attrapadung and Imai [2] proposed another system by combining techniques from [8] with their previous work from [3], which inherits the advantages of both approaches.

Revocation in PE Schemes and Privacy The different ABE revocation techniques mentioned above, aside from their scalability issues, are only partially applicable to PE schemes due to the distinguished attribute-hiding property of the latter. In particular, care should be taken to ensure that by introducing revocation to a PE system this privacy property is preserved. To the best of our knowledge, revocation in PE schemes has not been investigated so far and it is not clear whether revocation introduces further privacy challenges, in addition to the challenge of preserving their basic attribute-hiding property. We observe that additional privacy problems may arise in scenarios where revocation is performed for individual private keys. For example, in the revocable ABE scheme of Attrapadung and Imai [3], each sender builds a revocation list on-the-fly, using unique identifiers of users’ private keys, and encodes this list into the ciphertext. However, a close inspection of the scheme shows that ciphertexts reveal information about the encoded key identifiers and by this leak information about the revoked users. In this chapter we explore the concept of privacy-preserving revocation in PE schemes.
4.1.1 Contributions of this chapter

We formalize the concept of Revocable Predicate Encryption (RPE) and propose a RPE scheme allowing for efficient revocation of individual private keys. The underlying revocation mechanisms do not require any private key update procedures on the recipient’s side and more importantly preserve the expected privacy properties of PE schemes. The scheme uses revocation information on the sender’s side only to perform the encryption operation. Only holders of unrevoked private keys are able to decrypt the ciphertext, still provided that their keys also match the decryption policy. In the following we discuss the notion of privacy behind our scheme, the usage, and underlying techniques used.

Attribute-Hiding RPE. Our proposed scheme, termed AH-RPE, offers attribute-hiding, which is the standard PE property (and further implies payload-hiding used in the context of ABE). The revocation concept behind AH-RPE uses revocation lists (RL) and is mostly suitable for applications where revocation management is handled centrally by the PKG. It is assumed that senders obtain up-to-date RL published by the PKG prior to encryption. The attribute-hiding property of our AH-RPE scheme is proven against adaptive adversaries in the standard model under the established DLIN assumption. The AH-RPE scheme has constant-size private and public keys while the length of its ciphertexts remains linear in the number of revoked keys.

Techniques. Our scheme is based on the Dual System Encryption of Waters [61] and the Dual Pairing Vector Spaces (DPVS) of Okamoto and Takashima [42]. The scheme deploys the revocation system of Lewko, Sahai and Waters [36], introduced originally for public-key broadcast encryption, and modified here for an integration with the (payload-hiding) FE scheme of Okamoto and Takashima [43] in a way that achieves attribute-hiding by further using some techniques underlying the PE scheme by Lewko et al. [35]. In order to prove security of our RPE schemes we utilize the modular approach from Okamoto and Takashima [43] that breaks the proof down into several higher-level (artificially looking) assumptions and proves them to be secure under the DLIN assumption. The technical challenge in our proofs is to actually adopt the proving techniques from Okamoto and Takashima [43] towards the RPE requirements of attribute-hiding (for AH-RPE), whose definition has more sophisticated “win conditions”.

4.2 Security Model

In predicate encryption for the inner-product relation, an attribute is expressed as a vector \( \vec{y} \in \mathbb{F}_q^n \setminus \{ \overrightarrow{0} \} \) and a predicate \( f_\mathcal{P} \) is associated with a vector \( \vec{x} \in \mathbb{F}_q^n \setminus \{ \overrightarrow{0} \} \), where \( f_\mathcal{P}(\overrightarrow{y}) = 1 \), iff \( \vec{y} \cdot \vec{x} = 0 \). Let \( A = \mathbb{F}_q^n \setminus \{ \overrightarrow{0} \} \) be the attribute space, and \( \mathcal{P} = \{ f_\mathcal{P} | \vec{x} \in \mathbb{F}_q^n \setminus \{ \overrightarrow{0} \} \} \) be the predicate space. We assume that indexes are in the set \( \Gamma = \{ 1, \ldots, N \} \), where \( N \) is the number of keys in the system. In our definitions and scheme, we assume that the attribute vector, \( \overrightarrow{y} = (y_1, \ldots, y_n) \), is normalized such that \( y_1 = 1 \) (If \( \overrightarrow{y} \) is not normalized, change it to a normalized one by \( (1/y_1) \cdot \overrightarrow{y} \), assuming that \( y_1 \) is non-zero). \( \overrightarrow{e}_i^{(k)} \) is the canonical basis
vector \((0,\ldots,0,1,0,\ldots,0)\) \(\in \mathbb{F}_q^{n_k}\) for \(k = 1,2\) and \(i = 1,\ldots,n_k\). For base \(\mathbb{B} = (b_1,\ldots,b_N)\) and \(\mathbb{B}^* = (b_1^*,\ldots,b_N^*)\), \((x_1,\ldots,x_N)_{\mathbb{B}} = \sum_{i=1}^{N} x_i b_i\) and \((y_1,\ldots,y_N)_{\mathbb{B}^*} = \sum_{i=1}^{N} y_i b_i^*\).

4.2.1 Syntax

**Definition 4.1.** A Revocable Predicate Encryption (RPE) comprises of four algorithms (Setup, GenKey, Encrypt, Decrypt) and has associated attribute space \(\mathcal{A}\), predicate space \(\mathcal{P}\) and index space \(\Gamma\).

**Setup**\((1^\lambda, \Delta)\) The Setup algorithm takes as input a security parameter \(1^\lambda\) and format \(\Delta\) of attribute and index. It outputs a public key \(PK\), a master secret key \(MSK\), and state information \(S\).

**GenKey**\((MSK, S, \vec{x})\) The GenKey algorithm takes as input a master secret key \(MSK\), a state information \(S\), and a predicate vector \(\vec{x}\). It outputs an updated state \(S\) and a secret key \(k_{\vec{x},I}^*\), where \(I \in \Gamma\) denotes the associated index of the key and is included in the key.

**Encrypt**\((PK, L, \vec{y}, M)\) The Encrypt algorithm takes as input a public key \(PK\), a revocation list \(L \subseteq \Gamma\), an attribute vector \(\vec{y}\), and a message \(M\) in some associated message space. It outputs a ciphertext \(C\).

**Decrypt**\((C, k_{\vec{y},I}^*)\) The Decrypt algorithm takes as input a ciphertext \(C\) and a secret key \(k_{\vec{y},I}^*\). It outputs either a message \(M\) or the distinguished symbol \(\perp\).

**Correctness.** The correctness property of the schemes says that for all \(PK\) and \(MSK\) output by Setup algorithm, all predicate \(f_{\vec{x}} \in \mathcal{P}\), all message \(M\), all attribute \(\vec{y} \in \mathcal{A}\), and all possible valid state information \(S\) output by Setup or GenKey algorithm, if the key \(k_{\vec{x},I}^*\) was not revoked, i.e., \(I \notin L\), then for correctly generated \(k_{\vec{y},I}^* \overset{R}{\leftarrow} \text{GenKey}(MSK, S, \vec{x})\) and \(C \overset{R}{\leftarrow} \text{Encrypt}(PK, L, \vec{y}, M)\):

- If \(f_{\vec{x}}(\vec{y}) = 1\) then \(\text{Decrypt}(C, k_{\vec{y},I}^*) = M\).
- If \(f_{\vec{x}}(\vec{y}) = 0\) then \(\text{Decrypt}(C, k_{\vec{y},I}^*) = \perp\) with all but negligible probability.

4.2.2 Security Definition

The security definition for AH-RPE is derived from that of PE, which we extend to address revocation. We allow the adversary to specify the revocation list used to create the challenge ciphertext but we do not require ciphertexts to hide information about revoked key indices. This definition suits applications where revocation lists are managed and published by the master authority.

**Definition 4.2** (Attribute-Hiding RPE). An RPE scheme is adaptively attribute hiding against chosen plaintext attacks if for all PPT adversaries \(A\), the advantage \(\text{Adv}_{A,\text{RPE}}^\text{AH}(\lambda)\) of \(A\) in the following game is negligible in the security parameter \(\lambda\):
4.3. Our Scheme

Setup. A challenger $C$ runs the Setup algorithm to generate a public key $PK$, a master secret key $MSK$, and state information $S$. $PK$ is given to $A$.

Query phase 1. $A$ adaptively makes a polynomial number of GenKey queries: $A$ produces a predicate $\vec{f}$, $C$ computes the key $k_{\vec{f},I} \rightleftharpoons \text{GenKey}(MSK, S, \vec{f})$ associated with an index $I$, and gives it to $A$.

Challenge. $A$ outputs challenge attribute vectors $(\vec{y}(0), \vec{y}(1))$, challenge plaintexts $(M(0), M(1))$, and a revocation list $L$, subject to one of the following restrictions for each queried key $k_{\vec{f},I}$:

1. $I \in L$
2. $I \notin L$ and $f_{\vec{y}(0)} = f_{\vec{y}(1)} = 0$.

$C$ flips a random coin $b$. If $b = 0$ then $A$ is given $C = \text{Encrypt}(PK, L, \vec{y}(0), M(0))$. If $b = 1$ then $A$ is given $C = \text{Encrypt}(PK, L, \vec{y}(1), M(1))$.

Query phase 2. Repeat the Query phase 1 subject to the restrictions as in the challenge phase.

Guess. $A$ outputs a guess $b'$ of $b$, and succeeds if $b' = b$.

The advantage of $A$ is defined to be $\text{Adv}^{\text{AH-RPE}}_{\lambda,RPE}(A) = |\Pr[b = b'] - 1/2|$.

Remark 4.1. Definition 4.2 can be easily extended to capture chosen-ciphertext attacks (CCA) by allowing decryption queries (for all but the challenge ciphertext). The advantage of $A$ in such CCA game is defined to be $\text{Adv}^{\text{AH-RPE}_{\text{CCA}}}_{\lambda,RPE}(A) = |\Pr[b = b'] - 1/2|$. Our CPA-secure AH-RPE from Section 4.3 can be strengthened to resist CCA by applying the well-known CHK transformation from [18] that uses one-time signatures to authenticate the ciphertext.

4.3 Our Scheme

We present our AH-RPE scheme, which achieves the property of attribute hiding. We construct a system in which the sizes of public and private keys are small and constant. The size of the ciphertext is linear in the number of revoked keys, which is small relative to the total number of users. The efficiency of our scheme is comparable to the existing schemes of Lewko et al. [36] and Attrapadung et al. [3]. The broadcast encryption scheme proposed by Boneh et al. [14] produces ciphertexts and private keys of constant size, however the size of the public keys is linear in the number of users in the system.

Our construction uses the “two equation” revocation technique of the public broadcast encryption system of Lewko, Sahai and Waters (LSW) [36]. In the LSW scheme the secret $s$ that allows decryption of the ciphertext is broken into as many shares as revoked indexes. Using the “two equation” technique, a key whose index is not revoked in the ciphertext can be used to compute all the shares of the secret $s$. We combine LSW’s “two equation” concept [36] and Okamoto and Takashima’s FE scheme [43] to construct our first scheme. Informally, we compute a key for a predicate $\vec{f}$ and an associated index $I$, the ciphertext is encrypted with
two indexes are not equal), then we can recover the share $s_i$. If the inner product of two vectors is non zero (signifying that the two indexes are not equal), then we can recover the share $s_i$. Otherwise the decryption fails.

The detailed specification of our AH-RPE scheme follows:

In our scheme, we will use the following probabilistic generator $G_{\text{ob}}$ for dual orthonormal bases:

This generator is different from the one constructed in Chapter 3).

$G_{\text{ob}}(1^\lambda, \overrightarrow{n}) = (d; n_1, \ldots, n_d)$:

For $l = 0, \ldots, d$:

- $\text{param}_{\gamma_l} = (q, \mathcal{G}_T, \gamma, \epsilon) \overset{R}{\leftarrow} \mathcal{G}_{\text{dews}}(1^\lambda, N_l, \text{param}_G)$.
- $A_l = (\lambda^{(l)}_j)^\epsilon \mathcal{G}_l(N_l, F_q)$, $(\mu^{(l)}_i, j) = \psi \cdot (\lambda^{(l)}_i)^{-1}$,
- $b^{(l)}_i = \sum_{j=1}^{N_l} \lambda^{(l)}_j a^{(l)}_{ij}$ for $i = 1, \ldots, N_l$, $B^{(l)} = (b^{(l)}_1, \ldots, b^{(l)}_{N_l})$,
- $b^{*l}_i = \sum_{j=1}^{N_l} \mu^{(l)}_i a^{(l)}_{ij}$ for $i = 1, \ldots, N_l$, $B^{*l} = (b^{*l}_1, \ldots, b^{*l}_{N_l})$,
- $g_T = e(G, G)^{\psi} \cdot \text{param}_{\gamma_l} = ([\text{param}_{\gamma_l}]_{l=0, \ldots, d}, g_T)$.

Output $(\text{param}_{\gamma_T}, \{B^{(l)}, B^{*l}\}_{l=0, \ldots, d})$. (Note that $g_T = e(b^{(l)}_i, b^{*l}_i)$ for $l = 0, \ldots, d; i = 1, \ldots, N_l$.)

**Setup** ($1^\lambda, \Delta = (\overrightarrow{n} = (2; n_1, n_2 = 2), N)$): Perform the following computations:

$G_{\text{ob}}(1^\lambda, \overrightarrow{n})$:

Let $S$ denote the (initially empty) state information on the so far assigned indices $I$. The output of the algorithm is given by the public key $PK = (1^\lambda, N, \text{param}_{\gamma_T}, \{\tilde{B}_{\epsilon(k)}\}_{k \in \{0, 1, 2\}})$, the master secret key $MSK = \{\{\tilde{B}_{\epsilon(k)}\}_{k \in \{0, 1, 2\}}\}$, and the state information $S$.

**GenKey**(MSK, $S$, $\overrightarrow{x} = (x_1, \ldots, x_{n_1}) \in \mathbb{F}_{q}^{n_1} \setminus \{\overrightarrow{0}\}$): Choose $s, \eta, \beta, \eta_1, \ldots, \eta_{n_1}, \rho_1, \rho_2 \overset{R}{\leftarrow} \mathbb{F}_q$,

Output the updated state information $S$ and the secret key $k^*_{I, i} = (I, k_0, k_1, k_2)$. 

a message $M$, an attribute $\overrightarrow{a}$ and the set of indexes $\{I_1, \ldots, I_r\}$ of the revoked keys, where $r$ is the number of the revoked keys. The ciphertext can be decrypted with the key if $I \neq I_i$ for all $i \in [1, r]$. In our scheme, we realize the “two equation” technique by employing non zero inner product evaluations. If the inner product of two vectors is non zero (signifying that the two indexes are not equal), then we can recover the share $s_i$. otherwise the decryption fails.
Encrypt($PK, L, \mathbf{y} = (y_1, \ldots, y_{n_1}) \in \mathbb{F}_q^{n_1} \setminus \{ 0 \}, M \in G_T$): If $L$ is empty, set $L = \{ N + 1 \}$, where $N + 1$ is a dummy index. Choose $\delta, \zeta, \varphi, \varphi' \in \mathbb{F}_q$, also choose $\varphi_r, \delta_r \in \mathbb{F}_q$ for all $r \in L$ such that $\delta = \sum_{r \in L} \delta_r$, and compute:

$$c_0 = (\delta, 0, \zeta, 0, \varphi)_{G(0)}.$$  
$$c_1 = (\delta \mathbf{y}, \mathbf{0}^{n_1}, \mathbf{0}^{n_1}, 0^{n_1}, 0, \varphi')_{G(1)},$$  
$$\forall r \in L: \quad c_r = (\delta_r(-r, 1), 0^2, 0^2, 0^2, \varphi_r)_{G(2)},$$  
$$c_M = g_T \cdot M.$$  

Output the ciphertext $C = (L, c_0, c_1, \{ c_r \}_{r \in L}, c_M)$.

Decrypt($C, k_{z, l}^\ast$): Given a ciphertext $C = (L, c_0, c_1, \{ c_r \}_{r \in L}, c_M)$ and a secret key $k_{z, l}^\ast = (I, k_0, k_1, k_2)$, if $I \in L$, output $\perp$; otherwise compute and output message $M$:

$$M = c_M / \prod_{r \in L} e(c_r, k_2)^{\frac{1}{\varphi_r}} e(c_0, k_0) e(c_1, k_1).$$

The correctness of our scheme holds due to the following observation. Let $C$ and $k_{z, l}^\ast$ be as above. If $\mathbf{y} \cdot \mathbf{y} = 0$ and $I \notin L$ then $M$ can be recovered by compute $c_M / \prod_{r \in L} e(c_r, k_2)^{\frac{1}{\varphi_r}}$, since

$$e(c_0, k_0) e(c_1, k_1) \prod_{r \in L} e(c_r, k_2)^{\frac{1}{\varphi_r}} = g_T^{-s_\delta + \zeta} g_T^{s_\beta + \delta \mathbf{y} \cdot \mathbf{y}} g_T^{s_\gamma \sum_{r \in L} \delta_r} = g_T^{-s_\delta + \zeta} g_T^{s_\beta} = g_T^{\zeta}.$$  

Remark 4.2. In the Encrypt algorithm, if the revocation list $L$ is empty, i.e., no key is revoked, a dummy index $N + 1$ is placed into the revocation list. Since $N + 1$ is not in the index space $\Gamma$, the ciphertext computed from $L = \{ N + 1 \}$ and an attribute $\mathbf{y}$ can be decrypted by any key $k_{z, l}^\ast$ provided $\mathbf{y} \cdot \mathbf{y} = 0$.

In our scheme, the size of private key is small and constant, i.e., $(13 + 3n_1)|G|$, and size of the ciphertext is linear in the number of revoked keys, i.e., $(6 + 3n_1 + 7r)|G| + |G_T|$, where $r, |G|$ and $|G_T|$ denotes the number of revoked keys, size of group element in $G$ and size of group element in $G_T$ respectively.

4.4 Proof of Security

Theorem 4.1. Our AH-RPE is adaptively attribute hiding against chosen plaintext attacks under the DLIN assumption. For any adversary $A$, there exists a probabilistic polynomial time machine $D$ such that for any security parameter $\lambda$,

$$\text{Adv}^{\text{AH}}_{A, \text{AH-RPE}}(\lambda) \leq (2\nu + 1)\text{Adv}^{\text{DLIN}}_D(\lambda) + \psi.$$
where \( \nu \) is the maximum number of \( \mathcal{A} \)'s key queries and \( \psi = (2\nu|L| + 18\nu + 10)/q \) (\(|L| \) denotes the number of revoked keys).

**Outline of the Proof of Theorem 4.1** We start by defining two high-level computational problems, Problem 1 and 2, and show that each of these problems is hard under the classical DLIN assumption. Our proof uses a sequence of games. In our games we define semi-functional keys expressed by Eq. (4.7). Semi-functional ciphertexts are defined in Eqs. (4.1)-(4.4). Semi-functional keys can decrypt all normal ciphertexts, but not semi-functional ciphertexts. Semi-functional ciphertexts can be decrypted only by normal keys. We also introduce nominal semi-functional ciphertexts and keys in Eq. (4.6) and Eq. (4.5), similar to [37].

Normal ciphertexts and keys are used in Game 0 (the original game of Definition 4.2), while their nominal semi-functional or semi-functional counterparts are used in subsequent games only. In Game 1, the challenge ciphertext is changed to a semi-functional one. We then consider 2\( \nu \) game hops from Game 1 (Game 2-0), Game 2-0', Game 2-1, Game 2-1', Game 2-2, Game 2-2', ..., to Game 2-(\( \nu - 1 \))' and Game 2-\( \nu \). In Game 2-\( m \), the first \( m \) keys are semi-functional and the rest of the keys are normal, and challenge ciphertext is semi-functional. In Game 2-\( m' \), the first \( m \) keys are semi-functional and the \( (m + 1) \)-th key is nominal semi-functional while the remaining keys are normal, and challenge ciphertext is nominal semi-functional. In Game 3, all queried keys are semi-functional Eq. (4.7). In the last game, Game 3, all keys and challenge ciphertext are semi-functional, hence the adversary has zero advantage.

We then show that the difference in the adversary’s advantage between Games 0 and 1 is bounded by the advantage of any adversary against Problem 1. The advantage difference between Games 2-\( m' \) and 2-\( m \) is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption). Here, we introduce special forms of nominal semi-functional keys \( k^*_{\mathcal{T}} \) and ciphertext \( C_{\text{spec-nom-semi}} \), respectively. They equal their counterparts in semi-functional forms except that there is correlation between coefficients over base \( b_2^{(0)}, b_2^{(1)}, b_3^{(1)} \) and \( b_3^{(2)} \), i.e., \( \epsilon \omega = \gamma = \gamma^{(1)} + \gamma^{(2)} \) (note that \( \epsilon, \gamma^{(1)} \) and \( \gamma^{(2)} \) are the randomness in \( k^*_{\mathcal{T}} \) Eq. (4.5) and \( w \) is the randomness in \( C_{\text{nom-semi}} \) Eq. (4.6)). \( k^*_{\mathcal{T}} \) and \( C^*_{\text{spec-nom-semi}} \) are simulated using Problem 2 instance when \( \beta = 1 \). Due to the algebraic structure, \( k^*_{\mathcal{T}} \) can always decrypt \( C_{\text{spec-nom-semi}} \) when the predicate holds and the key is not revoked. Therefore, it is hard for the simulator to identify if \( (k^*_{\mathcal{T}}, C^*_{\text{spec-nom-semi}}) \) or \( (k^*_{\mathcal{T}}, C_{\text{nom-semi}}) \) for Game 2-\( m' \) or \( (k^*_{\mathcal{T}}, C_{\text{semi}}) \) for Game 2-\( m \) under Problem 2. On the other hand, \( \gamma \) is independently distributed from the other variables when either the predicate does not hold or the key is revoked. That is, the joint distribution of \( (k^*_{\mathcal{T}}, C^*_{\text{spec-nom-semi}}) \) is equivalent to that of \( (k^*_{\mathcal{T}}, C_{\text{nom-semi}}) \) when either condition holds. Hence, both of them appear identical from the adversary’s view, since from the security definition the adversary’s queries should satisfy at least one of the conditions (predicate does not hold and key is revoked). With a similar argument, we show that the advantage difference between Games 2-\( m' \) and 2-\( (m + 1) \) is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption). We also show that Game 2-\( \nu \) can be conceptually changed to Game 3 whose advantage is 0.

**Definition 4.3** (Problem 1). Problem 1 is to decide on bit \( \beta \in \{0, 1\} \), given \((\text{param}_{\mathcal{T}}, \{\mathcal{B}^{(k)}},\)
4.4. Proof of Security

For any adversary \( \mathcal{B} \) (Problem 2), there exists a probabilistic machine \( \mathcal{D} \), whose running time is essentially the same as that of \( \mathcal{B} \), such that for any security parameter \( \lambda \), \( \text{Adv}^{P1}_{\mathcal{B}}(\lambda) \leq \text{Adv}^{2\text{LIN}}_{\mathcal{D}}(\lambda) + 8/\lambda \).

**Definition 4.4** (Problem 2). Problem 2 is to decide on bit \( \beta \in \{0, 1\} \), given (param\( \mathcal{P}_{\mathcal{B}} \), \( \hat{\beta}^{(k)} \), \( t^{(0)} \), \( t^{(1)}_{\beta,1}, \cdots, t^{(1)}_{\beta,n_1}, t^{(2)}_{\beta,1}, \cdots, t^{(2)}_{\beta,n_2} \) ) \( R \) \( \mathcal{G}^{P1}_\beta(1^\lambda, \vec{t}) = (2; n_1, n_2 = 2) \), where

\[
\mathcal{G}^{P1}_\beta(1^\lambda, \vec{t}) = (2; n_1, n_2 = 2) : \quad (\text{param}\mathcal{P}_{\mathcal{B}}, \mathcal{B}^{(0)}, \mathcal{B}^{(1)}, \mathcal{B}^{(2)}, \mathcal{B}^{(3)}) \overset{R}{\rightarrow} \mathcal{G}^{P1}_\beta(1^\lambda, \vec{t})
\]

Let \( \mathcal{B} \) be a probabilistic machine, we define the advantage of \( \mathcal{B} \) for Problem 1 as follows:

\[
\text{Adv}^{P1}_{\mathcal{B}}(\lambda) = \left| \Pr \left[ \mathcal{B}(1^\lambda, \vec{w}) \rightarrow 1 \right] - \Pr \left[ \mathcal{B}(1^\lambda, \vec{w}) \rightarrow 1 \right] \right| \overset{R}{\rightarrow} \mathcal{G}^{P1}_\beta(1^\lambda, \vec{t})
\]

**Lemma 4.1.** For any adversary \( \mathcal{B} \), there exists a probabilistic machine \( \mathcal{D} \), whose running time is essentially the same as that of \( \mathcal{B} \), such that for any security parameter \( \lambda \), \( \text{Adv}^{P1}_{\mathcal{B}}(\lambda) \leq \text{Adv}^{2\text{LIN}}_{\mathcal{D}}(\lambda) + 8/\lambda \).
\[ h_{i_1,i_2}^{\ast(k)} = (\omega^{e_i(k)} \cdot \omega^{e_{i_1,i_2}(k)} \cdot \omega^{e_{i_1}(k)} \cdot \omega^{e_{i_2}(k)} \cdot \omega^{e_{i_1,n_{i_2}}(k)} \cdot \omega^{e_{i_2,n_{i_1}}(k)} \cdot \omega^{e_{i_1,n_{i_1}}(k)} \cdot \omega^{e_{i_2,n_{i_2}}(k)} \cdot \omega^{e_{i_1,n_{i_2}}(k)} \cdot \omega^{e_{i_2,n_{i_1}}(k)} \cdot \omega^{e_{i_1,n_{i_1}}(k)} \cdot \omega^{e_{i_2,n_{i_2}}(k)} \cdot \omega^{e_{i_1,n_{i_1}}(k)} \cdot \omega^{e_{i_2,n_{i_2}}(k)} \cdot \omega^{e_{i_1,n_{i_1}}(k)} \cdot \omega^{e_{i_2,n_{i_2}}(k)} \cdot \omega^{e_{i_1,n_{i_1}}(k)} \cdot \omega^{e_{i_2,n_{i_2}}(k)}, 1 \bigg|_{B^\ast(k)}, \]

\[ t_i^{(k)} = (\delta^{e_i(k)} \cdot \pi_{u_i(k)} \cdot \pi_{u_{i_1}(k)} \cdot \pi_{u_{i_2}(k)} \cdot 0^{\pi u_i(k)}, 1 \bigg|_{B^{(k)}}, \]

Output \((\text{param}, \hat{B}^{(0)}, \hat{B}^{\ast(0)}, h_{i_1,i_2}^{\ast(0)}, t^{(0)}, \{\hat{h}_{\beta,i}^{(k)}, \hat{t}_i^{(k)}\}_{i=1,\ldots,n_{i_2}} \}_{k=1,2})\).

Let B be a probabilistic machine, we define the advantage of B for Problem 2 as follows:

\[
\text{Adv}_B^{P_2}(\lambda) = \left| \Pr[B(1^\lambda, x) \rightarrow 1 \big| x \xleftarrow{R} \mathcal{G}_0^{P_2}(1^\lambda, \pi)] - \Pr[B(1^\lambda, x) \rightarrow 1 \big| x \xleftarrow{R} \mathcal{G}_1^{P_2}(1^\lambda, \pi)] \right|.
\]

**Lemma 4.2.** For any adversary B, there exists a probabilistic machine D, whose running time is essentially the same as that of B, such that for any security parameter \(\lambda\), \(\text{Adv}_B^{P_2}(\lambda) \leq \text{Adv}_D^{\text{DLIN}}(\lambda) + 5/q\).

**Proof of Lemmas 4.1 and 4.2.** In order to reduce the DLIN problem to Problems 1 and 2 from Definitions 4.3 and 4.4, respectively, we further introduce three “basic problems” that will serve in intermediate steps of the reduction:

- Basic Problem 0 in Definition 4.5
- Basic Problem 1 in Definition 4.6
- Basic Problem 2 in Definition 4.7

In order to prove Lemmas 4.1 and 4.2, we also use two intermediate lemmas (Lemmas 4.3 and 4.4) that are common lemmas used in the proofs of Lemmas 4.1 and 4.2.

**Lemma 4.3.** Let \((q, V, G_T, A, e)\) be dual pairing vector spaces by direct product of symmetric pairing groups. Using \(\{\phi_{i,j}\}\), we can efficiently sample a random linear transformation \(W = \sum_{i=1}^{N} r_{i,j} \phi_{i,j} \) of \(V\) with random coefficients \(\{r_{i,j}\}_{i,j=1}^{N} \xleftarrow{U} \mathcal{G}(N, \mathbb{F}_q)\). The matrix \((r_{i,j}^{*}) = (\{r_{i,j}\}_1^N)^T\) defines the adjoint action on \(V\) for pairing e, i.e., \(e(W(x), W^{-1}(y)) = e(x, y)\) for any \(x, y \in V\), where \((W^{-1})^T = \sum_{i=1}^{N} r_{i,j}^* \phi_{i,j}\).

The proof of Lemma 4.3 can be found in [43].

**Definition 4.5** (Basic Problem 0). Basic Problem 0 is to decide bit \(\beta\), given \((\text{param}_{BP0}, \hat{B}, B^\ast, y_{\beta}^\ast, f, bG, aG, \delta aG) \xleftarrow{R} \mathcal{G}_{BP0}^\beta(1^\lambda)\) for \(\beta \xleftarrow{U} \{0, 1\}\) with probability non-negligibly better than by a random guess, where

\[
\mathcal{G}_{BP0}^\beta(1^\lambda) : \\
\text{param}_{\mathcal{G}} = (q, G, G_T, G, e) \xleftarrow{R} \mathcal{G}_{\text{bgg}}(1^\lambda), \\
\text{param}_{\mathcal{M}} = (q, V, G_T, A, e) \xleftarrow{R} \mathcal{G}_{\text{dpvs}}(1^\lambda, 3, \text{param}_{\mathcal{G}}), \\
\Lambda = (\lambda_{i,j}) \xleftarrow{U} \mathcal{G}(3, \mathbb{F}_q), \ (\mu_{i,j}) = (\Lambda^T)^{-1}, \ b, a \xleftarrow{U} \mathbb{F}^\times, \\
b_i = b \sum_{j=1}^{3} \lambda_{i,j} a_j, \ i = 1, 3, \ \hat{B} = (b_3, b_3), \]
4.4. Proof of Security

**Lemma 4.4.** For any adversary $\mathcal{F}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{D}$, such that for any security parameter $\lambda$, $\text{Adv}_{\mathcal{F}}^{\text{BP0}}(\lambda) \leq \text{Adv}_{\mathcal{D}}^{\text{DLIN}}(\lambda) + 5/q$.

The proof of Lemma 4.4 can be found in [43].

**Proof of Lemma 4.4.** Combining Lemma 4.3, 4.4, 4.5 and 4.6, we obtain Lemma 4.1.

**Definition 4.6 (Basic Problem 1).** Basic Problem 1 is to decide bit $\beta$, given $(\text{param}_{\mathcal{F}}, \{\mathbb{B}^{(k)}\}_k)\rangle = (f_0^{(0)}, f_{1,1}^{(1)}, f_{1,2}^{(2)}, \{f_i^{(1)}\}_{i=2,\ldots,n_1}, f_2^{(2)}) \overset{R}{\leftarrow} \mathcal{G}_{\beta}^{\text{BP1}}(1^\lambda, \tau) = (2, n_1, n_2 = 2)$ for $\beta \overset{V}{\leftarrow} \{0,1\}$, with probability non-negligibly better than by a random guess, where

$$
\begin{align*}
&\mathbb{B}^{(k)} \\
&\mathcal{G}_{\beta}^{\text{BP1}}(1^\lambda, \tau = (2, n_1, n_2 = 2)) : \\
&(\text{param}_{\mathcal{F}}, \mathbb{B}^{(0)}, \mathbb{B}^{(*0)}, \mathbb{B}^{(1)}, \mathbb{B}^{(*1)}, \mathbb{B}^{(2)}, \mathbb{B}^{(*2)}) \overset{R}{\leftarrow} \mathcal{G}_{\text{ob}}(1^\lambda, \tau), \\
&\mathbb{B}^{(*0)} = (b_1^{(*0)}, b_2^{(*0)}, b_3^{(*0)}, b_4^{(*0)}), \\
&\mathbb{B}^{(*1)} = (b_1^{(*1)}, \ldots, b_{n_1}^{(*1)}, b_{n_1+1}^{(*1)}, \ldots, b_{n_2}^{(*1)}), \\
&\mathbb{B}^{(*2)} = (b_1^{(*2)}, b_2^{(*2)}, b_3^{(*2)}, b_4^{(*2)}, b_5^{(*2)}, b_6^{(*2)}, b_7^{(*2)}), \\
&\omega, \gamma \overset{V}{\leftarrow} \mathbb{F}_q, \tau \overset{V}{\leftarrow} \mathbb{F}_q^\times, f_0^{(0)} = (\omega, 0, 0, 0, \gamma)_{\mathbb{B}^{(0)}}, f_1^{(0)} = (\omega, \tau, 0, 0, \gamma)_{\mathbb{B}^{(0)}}, \\
&\rho^{(1)} \overset{U}{\leftarrow} \mathbb{B}^{(1)}.
\end{align*}
$$

For $k = 1, 2: \mathbb{B}^{(*k)} \overset{U}{\leftarrow} \mathbb{F}_q^{\mathbb{B}^{(*k)}}$,

$$
\begin{align*}
f_{0,1}^{(1)} &= (\omega \mathbb{G}^{(1)}, 0^n, 0^n, 0^n, 1)_{\mathbb{B}^{(1)}}, \\
f_{1,1}^{(1)} &= (\omega \mathbb{G}^{(1)}, \tau \mathbb{G}^{(1)}, 0^n, 0^n, 1)_{\mathbb{B}^{(1)}}, \\
\text{For } i = 2, \ldots, n_1: f_i^{(1)} &= \omega \mathbb{G}^{(1)}; \\
f_{0,1}^{(2)} &= (\omega, 0, 0^2, 0^2, 1)_{\mathbb{B}^{(2)}}, \\
f_{1,1}^{(2)} &= (\omega, \tau, 0, 0^2, 1)_{\mathbb{B}^{(2)}}, \\
f_{0,1}^{(2)} &= (\omega, 0, 0^2, 0^2, 1)_{\mathbb{B}^{(2)}}, \\
f_{2}^{(2)} &= \omega \mathbb{G}^{(2)}.
\end{align*}
$$

Output $(\text{param}_{\mathcal{F}}, \mathbb{B}^{(k)}, \mathbb{B}^{(*k)}) \overset{\tau}{\leftarrow} (f_0^{(0)}, f_{1,1}^{(1)}, f_{1,2}^{(2)}, \{f_i^{(1)}\}_{i=2,\ldots,n_1}, f_2^{(2)})$. 

Let $\text{Adv}_{\mathcal{F}}^{\text{BP0}}(\lambda)$ denote the corresponding advantage of a PPT algorithm $\mathcal{F}$ for the Basic Problem 0.
Let $\text{Adv}_C^{\text{BP1}}(\lambda)$ denote the advantage of a PPT algorithm $C$ for the Basic Problem 1.

**Lemma 4.5.** For any adversary $C$, there exists a probabilistic machine $F$, whose running time is essentially the same as that of $C$, such that for any security parameter $\lambda$, $\text{Adv}_C^{\text{BP1}}(\lambda) \leq \text{Adv}_F^{\text{BP0}}(\lambda)$ for $n_1 = 2$.

**Proof.** $F$ is given a Basic Problem 0 instance $(\text{param}_{\text{BP0}}, \hat{B}, B^*, y^*_b, f, bG, aG, acG)$. Using $\text{param}_G = (q, G, G_T, G, e)$ contained in $\text{param}_{\text{BP0}}$, $F$ computes:

$$
\text{param}_{\nu_0} = (q, V_0, G_T, A_0, e) \overset{R}{\leftarrow} G_{dpsv}(1^\lambda, 5, \text{param}_G),
$$

$$
\text{param}_{\nu_l} = (q, V_l, G_T, A_l, e) \overset{R}{\leftarrow} G_{dpsv}(1^\lambda, 3n_l + 1, \text{param}_G), \ l = 1, 2,
$$

$$
\text{param}_{\pi_l} = (\{\text{param}_{\nu_l}\}_{l=0,1,2}, g_T),
$$

where $G_T$ is contained in $\text{param}_{\text{BP0}}$. $F$ generates random linear transformation $W_l$ on $V_l(l = 0, 1, 2)$ given in Lemma 4.3, then sets

$$
\begin{align*}
\hat{d}_l^{(0)} &= W_0(b_l^*, 0, 0), \quad l = 1, 2; \quad \hat{d}_1^{(0)} = W_0(0, 0, 0, 0, aG), \\
\hat{d}_2^{(0)} &= W_0(0, 0, 0, 0, aG, 0), \quad \hat{d}_3^{(0)} = W_0(b_3^*, 0, 0), \\
\hat{d}_4^{(0)} &= (W_0^{-1})^T(b_0, 0, 0), \quad l = 1, 2; \quad \hat{d}_3^{(0)} = (W_0^{-1})^T(0, 0, 0, 0, bG), \\
\hat{d}_4^{(0)} &= (W_0^{-1})^T(0, 0, 0, bG, 0), \quad \hat{d}_5^{(0)} = (W_0^{-1})^T(b_3, 0, 0), \\
\hat{g}_b^{(0)} &= W_0(y_b^*, 0, 0), \\
\hat{d}_1^{(1)} &= W_1(b_1^*, 0^{N_1-3}), \quad \hat{d}_2^{(1)} = W_1(b_2^*, 0^{N_1-3}), \quad \hat{d}_3^{(1)} = W_1(b_3^*, 0^{N_1-3}), \\
\hat{d}_4^{(1)} &= W_1(0^m, aG, 0^{N_1-m-1}) \text{ where } m = l + 1 \text{ if } l \in \{2, \ldots, n_l\}, \\
\hat{d}_5^{(1)} &= (W_1^{-1})^T(b_0, 0^{N_1-3}), \quad \hat{d}_3^{(1)} = (W_1^{-1})^T(b_2, 0^{N_1-3}), \quad \hat{d}_4^{(1)} = (W_1^{-1})^T(b_3, 0^{N_1-3}), \\
\hat{d}_5^{(1)} &= (W_1^{-1})^T(0^m, bG, 0^{N_1-m-1}) \text{ where } m = l + 1 \text{ if } l \in \{2, \ldots, n_l\}, \\
\hat{g}_b^{(1)} &= W_1(y_b^*, 0^{N_1-3}), \\
\hat{g}_l^{(1)} &= W_1(0^{l+1}, acG, 0^{N_1-l-2}), \quad l = 2, \ldots, n_1;
\end{align*}
$$

$$
\begin{align*}
\hat{d}_2^{(2)} &= W_2(b_1^*, 0^4), \quad \hat{d}_3^{(2)} = W_2(b_2^*, 0^4), \quad \hat{d}_4^{(2)} = W_2(b_3^*, 0^4), \\
\hat{d}_4^{(2)} &= W_2(0^m, aG, 0^{7-m-1}) \text{ where } m = 3 \text{ if } l = 2, \\
\hat{d}_5^{(2)} &= (W_2^{-1})^T(b_0, 0^4), \quad \hat{d}_3^{(2)} = (W_2^{-1})^T(b_2, 0^4), \quad \hat{d}_4^{(2)} = (W_2^{-1})^T(b_3, 0^4), \\
\hat{g}_l^{(2)} &= W_2(0^{l+3}, acG, 0^{N_1-l-2}), \quad l = 2, \ldots, n_1.
\end{align*}
$$
4.4. Proof of Security

\[ d_i^{(2)} = (W_2^{-1})^T (0^m, bG, 0^{7-m-1}) \] where
\[
m = 3 \text{ if } l = 2,
\] \[
m = l \text{ if } l \in \{4, \ldots, 6\},
\]
\[ g^{(2)}_{b, 1} = W_2 (a^{(2)}_{b, 1}, 0^3), \]
\[ g^{(2)}_2 = W_2 (0^3, acG, 0^3), \]

where \((v, 0^{N_v-3}) = (G', G'', G'''', 0^{N_v-3})\) for any \(v = (G', G'', G''') \in V = G^3\). In this way bases \(\mathbb{D}^{(0)} = (d_i^{(0)})_{i=1, \ldots, 5}\) and \(\mathbb{D}^{(j)} = (d_i^{(j)})_{i=1, \ldots, 3n_j + 1}\) and \(\mathbb{D}^{(j)} = (d_i^{(j)})_{i=1, \ldots, 3n_j + 1}\) are dual orthonormal bases.

Therefore, from \(F = (b_1, b_2), B^*, bG, \) and \(aG\) the algorithm \(F\) can compute \(\mathbb{D}^{(j)}, j = 0, 1, 2; \mathbb{D}^{(j)} = (d_1^{(0)}, d_3^{(0)}, d_4^{(0)}, d_5^{(0)}), \) and \(\mathbb{D}^{(j)} = (d_1^{(j)}, \ldots, d_{n_j}, a_{n_j + 2}, \ldots, d_{3n_j + 1}), j = 1, 2.\)

Finally, \(F\) hands \(\{\text{param}_{F}, \{\mathbb{D}^{(k)}, \mathbb{D}^{(k)}\}_{k=0, 1, 2}, b_{g_1}^{(0)}: b_{g_{\beta, 1}}^{(0)}: b_{g_{\beta, 1}}^{(2)}: b_{g_1}^{(1)}: b_{g_2}^{(2)}\} \) over to \(C\) and, if \(C\) outputs its bit \(\beta'\) then \(F\) forwards this bit as its own output.

We observe that:
\[ g_0^{(0)} = (\omega', 0, 0, 0, \gamma')_{\mathbb{D}^{(0)}}, \]
\[ g_1^{(0)} = (\omega', \tau', 0, 0, 0, \gamma')_{\mathbb{D}^{(2)}}, \]
\[ g_0^{(1)} = \omega' b_1^{(1)}, \text{ with } i = 2, \ldots, n_1; \]
\[ g_1^{(1)} = (\omega', 0, 0, 0, \gamma')_{\mathbb{D}^{(2)}}, \]
\[ g_2^{(1)} = (\omega', 0, 0, 0, \gamma')_{\mathbb{D}^{(2)}}, \]

de where \(\omega' = \delta, \tau' = \rho, \gamma' = \sigma\) are distributed uniformly in \(F_q\). Therefore, the distribution of \(\{\text{param}_{F}, \{\mathbb{D}^{(k)}, \mathbb{D}^{(k)}\}_{k=0, 1, 2}, b_{g_1}^{(0)}: b_{g_{\beta, 1}}^{(0)}: b_{g_{\beta, 1}}^{(2)}: b_{g_1}^{(1)}: b_{g_2}^{(2)}\} \) is exactly the same as in the instance of Basic Problem 1. \(\square\)

Lemma 4.6. For any adversary \(B\), there exists a probabilistic machine \(C\), whose running time is essentially the same as that of \(B\), such that for any security parameter \(\lambda\), \(\text{Adv}_C^{BP1}(\lambda) \leq \text{Adv}_B^{BP1}(\lambda) + 3/4 \) for \((\mathbb{H} = (2; n_1, n_2 = 2))\).

Proof. \(C\) is given an instance of the Basic Problem 1, i.e., \(\{\text{param}_{F}, \{\mathbb{D}^{(k)}, \mathbb{D}^{(k)}\}_{k=0, 1, 2}, f_0^{(0)}, f_1^{(0)}, f_2^{(0)}, f_1^{(2)}, f_2^{(1)}\}_{i=2, \ldots, n_2}, f_{\beta, 1}, f_{\beta, 2}, \) and computes \(r \leftarrow \text{span} < b_{1}^{(2)}, r' \leftarrow \text{span} < b_{2}^{(2)} >, \) and sets \(t_{\beta, 1}^{(2)} = f_{\beta, 1}^{(1)} + r \) and \(t_{\beta, 1}^{(2)} = f_{\beta, 2}^{(2)} + r'. \)

\(C\) then chooses \(u_0 \leftarrow F_q, (u_{D, 1}^{(k)}) \leftarrow GL(F_q, n_k), (z_{D, 1}^{(k)}) = ((u_{1}^{(k)})^{-1})^T \) for \(i = 1, \ldots, n_k, j = 1, \ldots, n_k, \) and \(k = 1, 2, \) and computes:
\[ d_0^{(0)} = (0, 0, 0, 0, 0)_{\mathbb{D}^{(0)}}, \]
\[ d_{n_1 + i}^{(k)} = (0, 0, 0, 0, 0)_{\mathbb{D}^{(0)}}, \]
Definition 4.7

Proof of Lemma 4.2

Then, \( C \) sets bases \( B^{(0)} = (b_1^{(0)}, b_2^{(0)}, b_3^{(0)}, b_4^{(0)}) \), \( \hat{B}^{(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}) \), \( D^{(k)} = (b_1^{(k)}, \ldots, b_\nu^{(k)}, d_{n+1}^{(k)}, \ldots, d_{2n+1}^{(k)}, \ldots, d_{3n+1}^{(k)}) \), \( \hat{D}^{(k)} = (b_1^{(k)}, b_\nu^{(k)}, b_\nu^{(k)}, \ldots, b_{3n+1}^{(k)}) \), \( k = 1, 2 \).

Finally, \( C \) hands \( (\text{param}_\beta, \{D^{(k)}, \hat{D}^{(k)}\})_{k=0,1,2}, f^{(0)}, \{f_i^{(0)}, \{f_i^{(1)} \}_{i=2,\ldots,n_1}, f_2^{(2)} \} \) over to \( B \) and, if \( B \) outputs its bit \( \beta' \) then \( C \) forwards this bit as its own output. Note that with respect to \( D^{(k)}, \hat{D}^{(k)}, \beta' \), \( k = 0, 1, 2 \), the above input to \( B \) has the same distribution as the instance of the Problem 1 unless following events occur: \( u = 0, \overrightarrow{u}^{(1)} = \overrightarrow{0} \) or \( \overrightarrow{u}^{(2)} = \overrightarrow{0} \). Those events occur with probability \( 3/q \) when \( \beta = 1 \).

\[ \square \]

Proof of Lemma 4.2

Combining Lemmas 4.3, 4.4, 4.7 and 4.8 we obtain Lemma 4.2

Definition 4.7 (Basic Problem 2). Basic Problem 2 is to find bit \( \beta \), given \( (\text{param}_\beta, \{\hat{B}^{(0)}, B^{(0)}, y_{\beta}^{(0)}, f^{(0)}, \{\hat{y}_\beta^{(0)}, y_\beta^{(0)}, f^{(0)}, \{y_\beta^{(0)}, f^{(0)}, \{y_\beta^{(0)}, f^{(0)}, \{y_{\beta, i}^{(k)}, f_i^{(k)} \}_{i=1,\ldots,n_k} \}_{k=1,2} \} \) ) ^{R} \) \( \text{ob}^{(1, \overrightarrow{u})} = (\beta = 1; n_1, n_2 = 2) \) for \( \beta \in \{0, 1\} \) with probability non-negligibly better than by a random guess, where

\[ \mathcal{G}_{\beta}^{BP2} \]

\[ (\overrightarrow{1}, \overrightarrow{u}) = (2; n_1, n_2 = 2) : \]

\[ (\text{param}_\beta, B^{(0)}, \hat{B}^{(0)}, B^{(1)}, \hat{B}^{(1)}, B^{(2)}, \hat{B}^{(2)}) \leftarrow \mathcal{G}_{\text{ob}}(1, \overrightarrow{u}), \]

\[ \hat{B}^{(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}), \]

\[ \hat{B}^{(1)} = (b_1^{(1)}) \ldots, b_{n_1+1}^{(1)}, \ldots, b_{n_2+1}^{(1)}), \]

\[ \hat{B}^{(2)} = (b_1^{(2)}, b_3^{(2)}, b_5^{(2)}, b_7^{(2)}), \]

\[ \omega, \xi, \delta \leftarrow F_q, z, \pi \leftarrow B_q, \]

\[ y_{\beta}^{(0)} = (\omega, 0, 0, 0, 0, B^{(0)}), \quad y_\beta^{(0)} = (\omega, z, 0, 0, 0, B^{(0)}), \quad f^{(0)} = (\delta, \pi, 0, 0, 0, B^{(0)}), \]

For \( k = 1, 2 \) and \( i = 1, \ldots, n_k \):

\[ y_\beta^{(k)} = (\omega e_i^{(k)}), \quad 0^{n_k}, \quad \xi e_i^{(k)}, \quad 0^{1} \}_{B^{(k)}}, \]

\[ y_{\beta, i}^{(k)} = (\omega e_i^{(k)}), \quad 0^{n_k}, \quad \xi e_i^{(k)}, \quad 0^{1} \}_{B^{(k)}}, \]

\[ f^{(k)} = (\sigma e_i^{(k)}, \pi e_i^{(k)}), \quad 0^{n_k}, \quad 1 \}_{B^{(k)}}, \]

Output \( (\text{param}_\beta, \hat{B}^{(0)}, B^{(0)}, y_{\beta}^{(0)}, f^{(0), \{\hat{y}_\beta^{(k)}, y_\beta^{(k)}, f_i^{(k)} \}_{i=1,\ldots,n_k} \}_{k=1,2} \). \]
Let $\text{Adv}^{\text{BP2}}_{\mathcal{C}}(\lambda)$ denote the corresponding advantage of a PPT algorithm $\mathcal{C}$ for the Basic Problem 2.

**Lemma 4.7.** For any adversary $\mathcal{C}$, there exists a probabilistic machine $\mathcal{F}$, whose running time is essentially the same as that of $\mathcal{C}$, such that for any security parameter $\lambda$, $\text{Adv}^{\text{BP2}}_{\mathcal{C}}(\lambda) = \text{Adv}^{\text{BP0}}_{\mathcal{F}}(\lambda)$ for $n = (2; n_1, n_2 = 2)$.

**Proof.** $\mathcal{F}$ is given an instance of the Basic Problem 0, i.e. $(\text{param}_{\text{BP0}}, \tilde{B}, \mathbb{B}^*, y_{\tilde{B}}, f, bG, aG, acG)$. Using $\text{param}_{\mathcal{C}} = (q, G, G_T, e)$ contained in $\text{param}_{\text{BP0}}$ it computes

$$\text{param}_{V_0} = (q, V_0, G_T, A_0, e) \overset{R}{\leftarrow} \mathcal{G}_0^\text{dpa}(1^\lambda, 5, \text{param}_{\mathcal{C}}),$$

$$\text{param}_{V_1} = (q, V_1, G_T, A_1, e) \overset{R}{\leftarrow} \mathcal{G}_0^\text{dpa}(1^\lambda, 3n_1 + 1, \text{param}_{\mathcal{C}}), \quad l = 1, 2,$$

$$\text{param}_{\overrightarrow{n}} = \{\text{param}_{V_l}\}_{l=0,1,2,3} G_T,$$

where $g_T$ is contained in $\text{param}_{\text{BP0}}$. Then, $\mathcal{F}$ generates random linear transformation $W_l$ on $V_l(l = 0, 1, 2)$ given in Lemma 1.3 and sets

$$d_l^{(0)} = W_0(b_l, 0, 0), \quad l = 1, 2; \quad d_3^{(0)} = W_0(0, 0, 0, bG),$$

$$d_4^{(0)} = W_0(b_2, 0, 0), \quad d_5^{(0)} = W_0(0, 0, 0, bG),$$

$$d_i^{(0)} = (W_0^{-1})^T(b_i, 0, 0), \quad l = 1, 2; \quad d_3^{(0)} = (W_0^{-1})^T(0, 0, 0, aG),$$

$$d_4^{(0)} = (W_0^{-1})^T(b_2, 0, 0), \quad d_5^{(0)} = (W_0^{-1})^T(0, 0, 0, aG),$$

$$p_{\beta}^{(0)} = (W_0^{-1})^T(y_{\tilde{B}}, 0, 0), \quad g^{(0)} = W_0(f, 0, 0),$$

For $k = 1, 2$:

For $l = 1, 2, 3$ and $i = 1, \ldots, n_k$:

$$d_{l(i-1)n_k + i}^{(k)} = W_k(0^{3(i-1)}, b_i, 0^3(n_k-i), 0),$$

$$d_{l(i-1)n_k + i}^{(k)} = W_k(0^{3(i-1)}, b_i, 0^3(n_k-i), 0),$$

For $l = 1, 2, 3$ and $i = 1, \ldots, n_k$:

$$d_{l(i-1)n_k + i}^{(k)} = (W_k^{-1})^T(0^{3(i-1)}, b_i, 0^3(n_k-i), 0),$$

$$d_{l(i-1)n_k + i}^{(k)} = (W_k^{-1})^T(0^{3(n_k-i)}, aG),$$

For $i = 1, \ldots, n_k$:

$$p_{\beta, i}^{(k)} = (W_k^{-1})^T(0^{3(i-1)}, y_{\tilde{B}}, 0^3(n_k-i), 0),$$

$$g_{i}^{(k)} = W_1(0^{3(i-1)}, f, 0^3(n_k-i), 0).$$

Observe that $\mathbb{D}^{(0)} = (d_{l(i-1)n_k + i})_{l=1,\ldots,5}$ and $\mathbb{D}^{(0)} = (d_{l(i-1)n_k + i})_{l=1,\ldots,5}$ and $\mathbb{D}^{(j)} = (d_{l(i-1)n_k + i})_{l=1,\ldots,3n_j+1}$ and $\mathbb{D}^{(j)} = (d_{l(i-1)n_k + i})_{l=1,\ldots,3n_j+1}$, $j = 1, 2$ are dual orthonormal bases.

Therefore, $\mathcal{F}$ can use $\tilde{B} = (b_l, b_2, b_3, b_4, b_5, bG)$, and $aG$ to compute bases $\mathbb{D}^{(j)}$, $j = 0, 1, 2$:

$$\tilde{B} = (d_{l(i-1)n_k + i})_{l=1,\ldots,5}$, and $\tilde{B} = (d_{l(i-1)n_k + i})_{l=1,\ldots,5}$, $d_{l(i-1)n_k + i}$, $d_{l(i-1)n_k + i}$, $d_{l(i-1)n_k + i}$, $j = 1, 2$.

Finally, $\mathcal{F}$ hands $(\text{param}_{\overrightarrow{n}}, \tilde{B}^{(0)}, \tilde{D}^{(0)}, p_{\beta}^{(0)}, g^{(0)}, \{\tilde{B}^{(k)}, \tilde{D}^{(k)}, p_{\beta, i}^{(k)}, g_{i}^{(k)}\}_{i=1,\ldots,n_k})_{k=1,2}$ over to $\mathcal{C}$ and, if $\mathcal{C}$ outputs a bit $\beta'$, forwards this bit as its own output.
We observe that:

\[ p_0^{(0)} = (ω, 0, 0, ξ, 0)_{D^{(0)}}, \quad p_1^{(0)} = (ω, 0, 0, ξ, 0)_{D^{(0)}}. \]

For \( k = 1, 2 \) and \( i = 1, \ldots, n_k \):

\[
\begin{align*}
p_{0, i}^{(k)} &= (ω\, e_i^{(k)}, \ 0, n_k, \ ξ\, e_i^{(k)}, \ 1)_{D^{(k)}}, \\
p_{1, i}^{(k)} &= (ω\, e_i^{(k)}, \ ξ\, e_i^{(k)}, \ 0, n_k, \ 1)_{D^{(k)}}, \\
g_i^{(k)} &= (β\, e_i^{(k)}, \ ξ\, e_i^{(k)}, \ 0, n_k, \ 1)_{D^{(k)}}.
\end{align*}
\]

Therefore, the distribution of \( \{\text{param}_π, \hat{D}_β^{(0)}, \hat{D}_β^{(0)}, p^{(0)}, p^{(0)}, \hat{D}_β^{(k)}, \hat{D}_β^{(k)}, \{p^{(k)}, g_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2} \) is exactly the same as in the instance of the Basic Problem 2.

\[ \square \]

**Lemma 4.8.** For any adversary \( B \), there exists a probabilistic machine \( C \), whose running time is essentially the same as that of \( B \), such that for any security parameter \( λ \), \( \text{Adv}^{P2}_B(λ) = \text{Adv}^{P2}_C(λ) \).

**Proof.** Given an instance of the Basic Problem 2, i.e., \( \{\text{param}_π, \hat{D}_β^{(0)}, \hat{D}_β^{(0)}, p^{(0)}, p^{(0)}, \hat{D}_β^{(k)}, \hat{D}_β^{(k)}, \{y_{β,i}^{(k)}, f_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2} \) the algorithm \( C \) computes \( r_i^{(k)} \leftarrow \text{span} < b_{n_k+1}^{(k)}, \ldots, b_{n_k}^{(k)} > \) and sets \( y_{β,i}^{(k)} = y_{β,i}^{(k)} + r_i^{(k)}, k = 1, 2. \)

Then, \( C \) chooses \( z_0' \leftarrow F_q \times \hat{D}_β^{(0)}, \ (z_i^{(k)}) \leftarrow GL(F_q, n_k), i = 1, \ldots, n_k, j = 1, \ldots, n_k, k = 1, 2 \), and computes:

\[
\begin{align*}
d_2^{(0)} &= (0, z_0, 0, 0, 0)_{D^{(0)}}, \\
d_{n_k+i}^{(k)} &= (0, n_k, z_i^{(k)}, 1, 0)_{D^{(k)}}, \quad i = 1, \ldots, n_k, \quad k = 1, 2.
\end{align*}
\]

Then, \( C \) sets \( z_0 = z^{-1}z_0', u_0 = z_0^{-1}, (z_i^{(k)}) = z^{-1}(z_i^{(k)}), \) and \( (u_i^{(k)}) = (z_i^{(k)})^{-1} \).

Finally, \( C \) hands \( \{\text{param}_π, \hat{D}_β^{(0)}, \hat{D}_β^{(0)}, p^{(0)}, p^{(0)}, \hat{D}_β^{(k)}, \hat{D}_β^{(k)}, \{y_{β,i}^{(k)}, f_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2} \) over
to $\mathcal{B}$ and outputs $\beta' \in \{0, 1\}$ if $\mathcal{B}$ outputs $\beta'$.

For $\pi$ in Basic Problem 2, let $\pi' = \pi$. Then, with respect to $\pi', \mathcal{G}(k), \mathbb{D}(k), k = 0, 1, 2$, the above answer to $\mathcal{B}$ has the same distribution as in the instance of Problem 2.

Next, we will prove our scheme $\text{AH-RPE}$ using a sequence of games under Problem 1 in Definition 4.3 and Problem 2 in Definition 4.4.

**Lemma 4.9.** For $p \in \mathbb{F}_q$, let $C_p = \{(\mathcal{F}, \mathcal{V})|\mathcal{F} \cdot \mathcal{V} = p\} \subset V \times V^*$ where $V$ is $n$-dimensional vector space $\mathbb{F}_q^n$, and $V^*$ its dual. For all $(\mathcal{F}, \mathcal{V}) \in C_p$, for all $(\mathcal{F}', \mathcal{V}') \in C_p$, $\Pr[\mathcal{F} \cdot \mathcal{V} = \mathcal{F}' \land \mathcal{V} \cdot \mathcal{V} = \mathcal{V}'] = \Pr[\mathcal{F} \cdot \mathcal{Z} = \mathcal{F}' \land \mathcal{V} \cdot \mathcal{U} = \mathcal{U}'] = 1/|\mathbb{C}_p|$, where $\mathbb{C} \leftarrow GL(n, \mathbb{F}_q)$, $U = (Z^{-1})^T$, and $\mathbb{Z} \cup \mathbb{C}_p$ denotes the number of elements in $C_p$.

The proof of Lemma 4.9 was given in [43]. Lemma 4.9 will be used in the proof of Lemma 4.10 and Lemma 4.11 shown later.

We then consider the following games:

**Game 0.** Let Game 0 denote the real security game defined in Definition 4.2.

**Game 1.** Game 1 is almost identical to Game 0, except that the target ciphertext $C = (L, c_0, c_1, \{c_r\}_{r \in L}, c_M)$ for challenge attribute vectors $(\mathcal{F}(0), \mathcal{F}(1))$, challenge plaintexts $(M(0), M(1))$ and a revocation list $L$ is

$$c_0 = (\delta, w, \zeta, \varphi)_{\mathbb{F}(0)},$$

$$c_1 = (\delta \mathcal{F}(b), w^{(1)}_1, \ldots, w^{(1)}_{n_1}, 0^{n_1}, \varphi^{(1)})_{\mathbb{F}(1)},$$

$$\forall r \in L : \quad c_r = (\delta, (\mathcal{F}(1), 1), w^{(2)}_{1,r}, w^{(2)}_{2,r}, 0^{2}, \varphi^{(2)}),$$

$$c_M = \delta \mathcal{F}(0),$$

where $\delta, w, \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)}, w^{(1)}_1, w^{(2)}_1 \leftarrow \mathbb{F}_q, b \leftarrow \{0, 1\}, \mathcal{F}(b) = (y^{(b)}_1, \ldots, y^{(b)}_{n_1})$, and $(w^{(1)}_1, \ldots, w^{(1)}_{n_1}) \leftarrow \mathbb{F}_q^{n_1}$.

**Game 2-$m'$ ($m = 0, \ldots, \nu - 1$).** Game 2-0 is Game 1. Game 2-$m'$ is almost identical to Game 2-$m$, except the reply to the $(m + 1)$-th GenKey query for $\mathcal{F} = (x_1, \ldots, x_{n_1})$, and challenge ciphertext are

$$k_0 = (-s, \epsilon, 1, \eta, 0)_{\mathbb{B}(0)},$$

$$k_1 = (s_1 \mathcal{F}(1) + \beta \mathcal{F}(0), 1^{n_1}, \sigma^{(1)}(\sigma^{(1)} \mathcal{F}(1), \mathcal{Z}(1), \mathcal{Z}(1), \epsilon, \epsilon, \eta^{(1)}_1, \eta^{(1)}_2, 0)_{\mathbb{B}(1)},$$

$$k_2 = (s_2(1, I), \mathcal{F}(2), 1^{n_1}, \eta^{(2)}_1, \eta^{(2)}_2, 0)_{\mathbb{B}(2)}.$$
\[ c_0 = \langle \delta, w, \zeta, 0, \varphi \rangle_{\mathbb{B}^{(0)}}, \]
\[ c_1 = \langle \delta \gamma y^{(b)}, \gamma y^{(b)}, U^{(1)}, 0^{n_1}, 1 \varphi^{(1)} \rangle_{\mathbb{B}^{(1)}}, \]
\[ \forall r \in L : c_r = \langle \delta_r (-r, 1), \delta_r (-r, 1) \cdot U^{(2)}, 0^2, 1 \varphi_r \rangle_{\mathbb{B}^{(2)}}, \]
\[ c_M = g_f^\zeta M^{(b)}, \]
\[ (4.6) \]

where \( \epsilon, \gamma^{(1)}, \gamma^{(2)}, \sigma^{(1)} \subseteq \mathbb{F}_q, I \subseteq \Gamma, Z^{(k)} \subseteq GL(\mathbb{F}_q, n_k), U^{(k)} = (Z^{(k)})^{-1} T \), \( k = 1, 2 \); \( \delta_r \subseteq \mathbb{F}_q \) for \( r \in L \) such that \( \sum_{r \in L} \delta_r = 1 \) and all the other variables are generated as in Game 2-\( m \).

**Game 2-\( m + 1 \) \( (m = 0, \ldots, \nu - 1) \).** Game 2-\( m + 1 \) is almost identical to Game 2-\( m \)', except the reply to the \((m + 1)\)-th GenKey query for \( \mathcal{T} = (x_1, \ldots, x_{n_1}) \) is:

\[ k_0 = (-s, \epsilon, 1, \eta, 0)_{\mathbb{B}^{(0)}}, \]
\[ k_1 = \langle s_1 \epsilon (1) + \beta^{(1)} \mathcal{T}, \nu^{(1)}_1, \ldots, \nu^{(1)}_{n_1}, \eta^{(1)}_1, \ldots, \eta^{(1)}_{n_1}, 0 \rangle_{\mathbb{B}^{(1)}}, \]
\[ k_2 = \langle s_2 (1), \nu^{(2)}_1, \nu^{(2)}_2, \eta^{(2)}_1, \eta^{(2)}_2, 0 \rangle_{\mathbb{B}^{(2)}}. \]

The challenge ciphertext is the same as Eqs. [4.1]-[4.4], where \( (\nu^{(k)}_1, \ldots, \nu^{(k)}_{n_k}) \subseteq \mathbb{F}_q \setminus \{0\} \), \( k = 1, 2 \), and all the other variables are generated as in Game 2-\( m \)'.

**Game 3.** Game 3 is almost identical to Game 2-\( \nu \), except that the target ciphertext \( C \) for challenge attribute vectors \( (y^{(0)}, y^{(1)}) \), challenge plaintexts \( (M^{(0)}, M^{(1)}) \), and a revocation list \( L \) is:

\[ c_0 = \langle \delta, w, \zeta', 0, \varphi \rangle_{\mathbb{B}^{(0)}}, \]
\[ c_1 = \langle \delta \gamma y^{(b)}, \nu^{(1)}_1, \ldots, \nu^{(1)}_{n_1}, 0^{n_1}, 1 \varphi^{(1)} \rangle_{\mathbb{B}^{(1)}}, \]
\[ \forall r \in L : c_r = \langle \delta_r (-r, 1), w^{(2)}_1, w^{(2)}_r, 0^2, 1 \varphi_r \rangle_{\mathbb{B}^{(2)}}, \]
\[ c_M = g_f^\zeta M^{(b)}. \]

where \( \zeta' \subseteq \mathbb{F}_q, \nu^{(1)} = (y^{(1)}_1, \ldots, y^{(1)}_{n_1}) \subseteq \mathbb{F}_q^{n_1} \). We note that \( \zeta' \) and \( (y^{(1)}_1, \ldots, y^{(1)}_{n_1}) \) are chosen uniformly and independently from \( \zeta, (y^{(0)}, y^{(1)}) \) respectively.

Let \( \text{Adv}^{(0)}_{\mathcal{A}}(\lambda) \) be \( \text{Adv}^{\text{AH-RPE}}_{\mathcal{A}, \text{AH-RPE}}(\lambda) \) in Game 0, and \( \text{Adv}^{(1)}_{\mathcal{A}}(\lambda), \text{Adv}^{(2-m)}_{\mathcal{A}}(\lambda), \text{Adv}^{(2-m')}_{\mathcal{A}}(\lambda), \text{Adv}^{(3)}_{\mathcal{A}}(\lambda) \) be the advantage of \( \mathcal{A} \) in Game 1, 2-\( m \), 2-\( m' \) and 3 respectively. We first prove Lemmas 4.10 - 4.14 that evaluate consecutive probability gaps between \( \text{Adv}^{(0)}_{\mathcal{A}}(\lambda), \text{Adv}^{(1)}_{\mathcal{A}}(\lambda), \text{Adv}^{(2-m')}_{\mathcal{A}}(\lambda), \text{Adv}^{(2-(m+1))}_{\mathcal{A}}(\lambda) \) for \( m = 0, \ldots, \nu - 1 \), and \( \text{Adv}^{(3)}_{\mathcal{A}}(\lambda) \), respectively.
Based on Lemmas 4.10–4.14 shown below and Lemmas 4.1 and 4.2 we then obtain:

\[
\text{Adv}_{A,AH,\text{RPE}}^\lambda = \text{Adv}_A^{(0)}(\lambda)
\]

\[
\leq | \text{Adv}_A^{(0)}(\lambda) - \text{Adv}_A^{(1)}(\lambda) | + \sum_{m=0}^{\nu-1} | \text{Adv}_A^{(2,m)}(\lambda) - \text{Adv}_A^{(2,m+1)}(\lambda) | + \text{Adv}_A^{(3)}(\lambda)
\]

\[
\leq \text{Adv}_{B_1}^{P_i}(\lambda) + \sum_{m=0}^{\nu-1} \text{Adv}_{B_2}^{P_2}(\lambda) + \sum_{m=0}^{\nu-1} \text{Adv}_{B_3}^{P_{3,m}}(\lambda) + (2\nu|L| + 8\nu + 2)/q
\]

\[
\leq (2\nu + 1)\text{Adv}_{D,\text{LIN}}^\lambda + (2\nu|L| + 18\nu + 10)/q.
\]

This completes the proof of Theorem 4.1.

**Lemma 4.10.** For any adversary \( A \), there exists a probabilistic machine \( B_1 \), whose running time is essentially the same as that of \( A \), such that for any security parameter \( \lambda \), \(| \text{Adv}_A^{(0)}(\lambda) - \text{Adv}_A^{(1)}(\lambda) | \leq \text{Adv}_{B_1}^{P_i}(\lambda) \).

**Proof.** Suppose a polynomial time adversary \( A \) can successfully distinguish between Game 0 and Game 1. We construct a simulator \( B_1 \) that leverages \( A \) as a black box to solve Problem 1. The procedure is shown as follows:

1. \( B_1 \) is given an instance of Problem 1, i.e. \((\text{param}_{\mathbb{P}}, \{\mathbb{B}^{(k)}_{\text{SUP}}\}_{k=0,1,2,3}, \hat{\mathbb{B}}^{(1)}, \hat{\mathbb{B}}^{(2)}, \{t^{(1)}_i\}_{i=2,\ldots,n_1}, t^{(2)}_2\)\), and plays the role of the challenger in the security game against adversary \( A \).

2. At the beginning of the game, \( B_1 \) gives \( A \) the public key \( PK = (1^\lambda, \text{param}_{\mathbb{P}}, \{(b^{(0)}_1, b^{(0)}_2, \ldots, b^{(0)}_{n_1+1}, b^{(1)}_1, \ldots, b^{(1)}_{n_1+1}, b^{(2)}_1, \ldots, b^{(2)}_{n_1+1})\})\), which is obtained from the Problem 1 instance.

3. When a GenKey query is issued, \( B_1 \) computes a normal secret key using \((\hat{\mathbb{B}}^{(0)}, \hat{\mathbb{B}}^{(1)}, \hat{\mathbb{B}}^{(2)}))\), which is obtained from the Problem 1 instance.

4. When \( B_1 \) receives challenge attribute vectors \((\hat{y}^{(0)}, \hat{y}^{(1)})\), challenge plaintexts \((M^{(0)}, M^{(1)})\) and a revocation list \( L \) from \( A \), \( B_1 \) computes and returns \( C = (L, c_0, c_1, \{c_r\}_{r\in L}, c_M) \) s.t. \( c_0 = t^{(0)}_0 + \zeta \beta^{(0)}_3, c_1 = y^{(1)}_1 t^{(1)}_1(\lambda) + \sum_{i=2}^{n_1} y^{(1)}_i t^{(1)}_i(\lambda), \forall r \in L : c_r = (pr/p)(-r)t^{(1)}_{\beta_1} + (pr/p)t^{(2)}_2 \) and \( c_M = \gamma^{(0)}_2 \hat{M}(\lambda) \), using \((\hat{y}^{(0)}_r, t^{(1)}_{\beta_1}, t^{(2)}_{\beta_1}, t^{(1)}_r)_{r=2,\ldots,n_1}, t^{(2)}_2, \hat{y}^{(0)}_3\) from the instance of Problem 1, \((\hat{y}^{(b)}, M^{(b)})\) and \( L \), where \( p, p_r \leftarrow \mathbb{F}_q \) for all \( r \in L \) such that \( p = \sum_{r \in L} p_r \) and \( p \leftarrow \mathbb{F}_q \).

5. After the challenge phase, GenKey oracle simulation for a key query is executed in the same manner as step 3.

6. \( A \) outputs a bit \( b' \). If \( b = b' \), \( B_1 \) outputs 1. Otherwise, \( B_1 \) outputs 0.

**Claim 4.1.** For \( \beta = 0 \) the challenge ciphertext \( C = (L, c_0, c_1, \{c_r\}_{r \in L}, c_M) \) generated in step 4 is distributed exactly as in Game 0, whereas if \( \beta = 1 \), the challenge ciphertext \( C = (L, c_0, c_1, \{c_r\}_{r \in L}, c_M) \) generated in step 4 is identically distributed as in Game 1.
Proof. First recall that $y_i^{(b)} = 1$. If $\beta = 0$, $c_0 = (\delta, 0, \zeta, 0, \rho)_{\mathbb{B}(0)}$, $c_1 = (\delta y_i^{(b)}, 0, 0, 0, \rho^{(1)})_{\mathbb{B}(1)}$, $\forall r \in L : c_r = (\delta_r (r, -1), 0, 0, 0, \varphi_r)_{\mathbb{B}(2)}$, and $c_M = g^{\hat{L}}_{1} M^{(b)}$ where $\delta_r = (p_r \delta)/p$, $\varphi_r = ((-r) p_r \varphi^{(2)})/p$. It is the challenge ciphertext in Game 0. If $\beta = 1$, $c_0 = (\delta, u, \zeta, 0, \rho)_{\mathbb{B}(0)}$, $c_1 = (\delta y_i^{(b)}, 0, 0, 0, \rho^{(1)})_{\mathbb{B}(1)}$, $\forall r \in L : c_r = (\delta_r (r, 1), u^{(2)}_1, u^{(2)}_r, 0, 0, \varphi_r)_{\mathbb{B}(2)}$, where $\overrightarrow{u}^{(1)} = (u_1^{(1)}, \ldots, u_{n_1}^{(1)})$, $\delta_r = (p_r \delta)/p$, $\varphi_r = ((-r) p_r \varphi^{(2)})/p$, $u^{(2)}_{1,r} = (u^{(2)}_r)_{\mathbb{F}_q}$, $u^{(2)}_r \in \mathbb{F}_q$ are independently uniform, it is the challenge ciphertext in Game 1.

From the above claim, if $\beta = 0$, the distribution of simulated values in the above simulation is exactly as in Game 0, whereas if $\beta = 1$, this simulation is distributed identically to Game 1. Therefore, $| \text{Adv}_{\lambda}^{(0)} (\lambda) - \text{Adv}_{\lambda}^{(1)} (\lambda) | = | Pr[B_1 (1^\lambda, x) \rightarrow 1] - \Pr [B_1 (1^\lambda, x) \rightarrow 1 - \frac{\epsilon}{\delta} g^{P_1}_{1} (1^\lambda, y)] | = \text{Adv}_{\lambda}^{(2, m)} (\lambda)$, which completes our proof of **Lemma 4.10**.

**Lemma 4.11.** For any adversary $A$, there exists a probabilistic machine $B'_{2, m}$, whose running time is essentially the same as that of $A$, such that for any security parameter $\lambda$, $| \text{Adv}_{\lambda}^{(2, m)} (\lambda) - \text{Adv}_{\lambda}^{(2, m')} (\lambda) | \leq \text{Adv}_{\lambda}^{(2, m)} (\lambda) + (|L| + 4)/q$.

**Proof.** Suppose a polynomial time adversary $A$ can successfully distinguish between Game 2-$m$ and Game 2-$m'$. We construct a simulator $B'_{2, m}$ that leverages $A$ as a black box to solve Problem 2. The procedure is shown as follows:

1. $B'_{2, m}$ is given an instance of Problem 2, i.e., $(\text{param}_{\mathbb{P}}, \hat{B}^{(0)}, \hat{B}^{+(0)}, \hat{B}^{(1)}, \hat{t}^{(0)}, \{\hat{B}^{(k)}, \hat{B}^{* (k)}\}, \{h^{* (k)}_{\beta, i}, k^{(k)}_{i} = 1, \ldots, n_k\}_{k = 1, 2})$ and plays the role of the challenger in the security game against adversary $A$.

2. At the beginning of the game, $B'_{2, m}$ gives $A$ the public key $PK = (1^\lambda, \text{param}_{\mathbb{P}}, (b^{(0)}_1, b^{(0)}_2, b^{(1)}_1, b^{(1)}_2, b^{(2)}_1, b^{(2)}_2), \text{which is obtained from the Problem 2 instance}$.  

3. When the $s$-th GenKey query is issued for a predicate $\overrightarrow{x} = (x_1, \ldots, x_{n_1})$, $B'_{2, m}$ answers as follows:

   a) For $1 \leq s \leq m$, $B'_{2, m}$ computes a semi-functional key using $\{\hat{B}^{* (k)}\}_{k = 0, 1, 2}$ of the problem 2 instance.

   b) For $s = m+1$, $B'_{2, m}$ computes $k^{*}_{\overrightarrow{x}, t}$, where $k^{*}_{\overrightarrow{x}, t} = (l, k_0, k_1, k_2)$ using $\{h^{* (0)}_{\beta, i}, b^{* (0)}_{1}, b^{* (1)}_{3}, \{\hat{h}^{* (i)}_{\beta, j}, b^{* (i)}_{j}\}_{i = 1, 2, j = 1, \ldots, n_i}\}$ of the problem 2 instance as follows:

   For $i = 2$:

   $$s^{(0)}_{\beta} = \sum_{i = 1}^{2} (g_{i} h^{* (0)}_{\beta, i} + v_i b^{* (0)}_{1}), \quad k_0 = -s^{(0)}_{\beta} + b^{* (0)}_{2},$$

   For $i = 1, 2$ and $j = 1, \ldots, n_i$:

   $$s^{(i)}_{\beta, j} = \theta_i h^{* (i)}_{\beta, j} + v_i b^{* (i)}_{j}, \quad s^{(i)}_{\beta, j} = g_{i} h^{* (i)}_{\beta, j} + v_i b^{* (i)}_{j},$$
Next we analyze the distribution of the same as the
It is clear that
Claim 4.2.
4.4. Proof of Security
b_i
k_1 = \sum_{j=1}^{n_1} x_j s_{\beta,j}^{(1)} + \hat{s}_{\beta,1}^{(1)},
\nonumber
k_2 = \hat{s}_{\beta,1}^{(2)} + I - \hat{s}_{\beta,2}^{(2)}.
\nonumber
c) For s \geq m + 2, B_{2m}' computes a normal key using \{\mathbb{B}_k^{(k)}\}_{k=0,1,2} of the problem 2 instance.

4. When B_{2m}' receives challenge attribute vectors (\hat{y}^{(0)}, \hat{y}^{(1)}), challenge plaintexts (M^{(0)}, M^{(1)}) and a revocation list L from A, B_{2m}' computes and returns C = (L, c_0, c_1, \{c_r\}_{r \in L}, c_M) s.t. c_0 = t^{(0)} + \zeta b_0^{(0)} + \varphi b_0^{(0)}, c_1 = \sum_{j=1}^{n_1} t_j^{(0)} + \varphi^{(1)} b_j^{(1)}, \forall r \in L : c_r = (-\varphi^{(2)} t_1^{(2)} + p r t_2^{(2)} + \varphi^{(2)} b_2^{(2)}, and c_M = \bar{g}_t M^{(0)}, using (t^{(0)}, \{t_j^{(0)}\}_{i=1,..,n_1}, t_1^{(2)}, t_2^{(2)}, b_3^{(0)}, \nonumber b_0^{(1)}, b_{\alpha+1}^{(1)}, b_r^{(2)} ) in the Problem 2 instance, \hat{y}^{(0)}, \hat{y}^{(1)}, M^{(0)} and L, where p, p r \in \mathbb{F}_q for all r \in L such that p = \sum_{r \in L} p_r, \zeta, \varphi, \varphi^{(1)}, \varphi r \in \mathbb{F}_q, b \in \{0, 1\}.

5. After the challenge phase, GenKey oracle simulation for a key query is executed in the same manner as step 3.

6. A outputs a bit b'. If b = b', B_{2m}' outputs 1. Otherwise, B_{2m}' outputs 0.

Claim 4.2. The distribution of the view of adversary A in the above-mentioned game simulated by B_{2m}' given a Problem 2 instance with \beta \in \{0, 1\} is the same as that in Game (2-m) (resp. Game (2-m')) if \beta = 0 (resp. \beta = 1) except with probability (3 + |L|)/q (resp. 1/q).

Proof. It is clear that B_{2m}'s simulation of the public key generation (step 2) and the answers to the i-th GenKey query where i \neq m + 1 (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and the GenKey oracles in Game 2-m and Game 2-m'.

Next we analyze the distribution of the i-th GenKey query where i = m + 1 (case (b) of steps (3) and (5)). Values \hat{s}_\beta^{(0)}, s_{\beta,j}^{(1)}, \hat{s}_{\beta,j}^{(1)} for i = 1, 2 and j = 1, \ldots, n_i in this case can be expressed as follows. Let \beta^{(i)} = \theta_i \omega + \nu_i, \alpha^{(i)} = \theta_i \omega + \nu_i, \alpha = \alpha^{(1)} + \alpha^{(2)}, \gamma = \theta_1 + \theta_2, and \epsilon = \gamma \epsilon.
\nonumber
s_0^{(0)} = (\alpha, 0, 0, \gamma \xi, 0)_{\mathbb{B}_0^{(0)}},
\nonumber
s_0^{(1)} = (\alpha, \epsilon, 0, \gamma \xi, 0)_{\mathbb{B}_0^{(1)}},
\nonumber
s_{\alpha,j}^{(1)} = (\frac{n_1}{n}, \frac{n_i}{n_i}, \frac{1}{0})_{\mathbb{B}_i^{(1)}},
\nonumber
s_{\beta,j}^{(1)} = (\frac{n_1}{n}, \frac{n_i}{n_i}, \frac{1}{0})_{\mathbb{B}_i^{(1)}},
\nonumber
\hat{s}_{\beta,j}^{(1)} = (\alpha^{(1)} \theta_i \omega + \nu_i, \alpha^{(1)} \theta_i \omega + \nu_i, 1)_{\mathbb{B}_i^{(1)}},
\nonumber
\hat{s}_{\beta,j}^{(1)} = (\alpha^{(1)} \theta_i \omega + \nu_i, \alpha^{(1)} \theta_i \omega + \nu_i, 1)_{\mathbb{B}_i^{(1)}},
\nonumber
where \zeta = \frac{n_2}{n_2}, \zeta = \frac{n_i}{n_i}, \omega, \xi, \xi \{\omega^{(1)}_{\beta,j}, \zeta^{(1)}_{\beta,j}\}_{i=1,2:j=1,..,n_i} are defined in Problem 2. When \beta = 1 in Problem 2 instance, \{k_{\beta,j}^{1}\} = (L, k_0, k_1, k_2) has the same distribution as Eq. 4.5 except that \epsilon w = \gamma, where \gamma and \epsilon = u \in \mathbb{F}_q of c_0 in Eq. 4.6.
Next, we show that the joint distribution of the answer to \((m + 1)\)-th GenKey query and the challenge ciphertext by \(B'_{2m}\)'s simulation given a Problem 2 instance with \(\beta\) is equivalent to that in Game 2-\(m\) (resp. Game 2-\(m'\)), when \(\beta = 0\) (resp. \(\beta = 1\)).

When \(\beta = 0\), it is straightforward to show that they are equivalent except that one of following conditions holds: (1) \(\omega\) defined in Problem 2 is zero, (2) \(w = 0\), (3) \((w_1^{(1)}, \ldots, w_1^{(1)}) = \vec{0}\), (4) \((w_2^{(2)}, w_2^{(2)}) = \vec{0}\), where \(r \in L\), and \(w, (w_1^{(1)}, \ldots, w_1^{(1)})\) and \((w_2^{(2)}, w_2^{(2)})\) are defined in Eqs. 4.1, 4.2 and 4.3 respectively. Those events occur with probability \((3 + |L|)/q\).

When \(\beta = 1\), \(B'_{2m}\)'s simulation for the key is the same as that expressed in Eq. 4.15 and \(B'_{2m}\)'s simulation for the the challenge ciphertext is that same as that expressed in Eq. 4.16 except that \(\epsilon w = \gamma\), where \(\gamma = \varrho_1 + \varrho_2\) and \(w \leftarrow U_q\) of \(c_0\) in Eq. 4.6.

Therefore, we will show that \(\gamma\) is uniformly and independently distributed from the other variables in \(B'_{2m}\)'s simulation. Note that \(\gamma\) is related to \(\vec{A}_1, \vec{A}_2, \vec{B}_1, \vec{B}_2, r\), where \(\vec{A}_1 = (\varrho_1 \vec{\varepsilon}_1^{(1)} + \theta_1 \vec{\varphi}) \cdot Z^{(1)}, \vec{A}_2 = (\varrho_2 (1, I)) \cdot Z^{(2)},\) and \(\vec{B}_1 = \vec{\varphi}^{(0)} \cdot U^{(1)}, \vec{B}_2, r = \delta_r (r, 1) \cdot U^{(2)}\) for all \(r \in L\), where \(b \in \{0, 1\}\). We show the joint distribution of these variables in the two cases that are defined in Definition 4.2.

1. When \(I \in L\), due to Lemma 4.9, the pair \((\vec{A}_2, \vec{B}_{2,r}\) is uniformly and independently distributed over \(C_0 = \{(\vec{\omega}, \vec{\varphi}) | \vec{\varphi} \cdot \vec{\varphi} = 0\} \) (over \(Z^{(2)} \leftarrow GL(F_q, 2)\)) for \(r = I\). When \(r \neq I\), the pair \((\vec{A}_2, \vec{B}_{2,r}\) is uniformly and independently distributed over \(C_{\varrho_2 \delta_r (I, r)} = \{(\vec{\omega}, \vec{\varphi}) | \vec{\varphi} \cdot \vec{\varphi} = \varrho_2 \delta_r (I, r)\} \) (over \(Z^{(2)} \leftarrow GL(F_q, 2)\)). The pair \((\vec{A}_2, \vec{B}_{2,r}\) (for \(r \in L\)) is uniformly and independently distributed over \(F_q^4\).

2. When \(I \notin L\) and \(f_{\vec{\varphi}}(\vec{\varphi}^{(0)}) = f_{\vec{\varphi}}(\vec{\varphi}^{(1)}) = 0\), due to Lemma 4.9, the pair \((\vec{A}_1, \vec{B}_1\) is uniformly and independently distributed on \(C_{\theta_1 (\vec{\varphi} \cdot \vec{\varphi}^{(0)}) + \varrho_1} \) (over \(Z^{(1)} \leftarrow GL(F_q, n_1)\)). Since \(\theta_1 \leftarrow F_q\), the pair \((\vec{A}_1, \vec{B}_1\) is uniformly and independently distributed over \(F_q^{4n_1}\).

Due to the restriction of adversary \(\mathcal{A}\)'s key queries defined in Definition 4.2, in each case, at least one of \((\vec{A}_1, \vec{B}_1\) and \((\vec{A}_2, \vec{B}_{2,r}\) is uniformly and independently distributed over \(F_q^{2n_k}\) for \(k = 1, 2\). Case 1 is obviously independent from \(\gamma\). Therefore, the distribution of \(\gamma = \varrho_1 + \varrho_2\) is independent from the distribution of \(\varrho_2\), which can be given by \((\vec{A}_2, \vec{B}_{2,r}\). Thus, \(\gamma\) is uniformly distributed and is independent from other variables used in the simulation performed by \(B'_{2m}\).

Therefore, the view of the adversary \(\mathcal{A}\) in the game simulated by \(B'_{2m}\) given an instance of Problem 2 with \(\beta = 1\) is the same as that in Game 2-\(m\) unless \(\omega = 0\). This event occurs with probability \(1/q\).

This completes the proof of Lemma 4.11.

Lemma 4.12. For any adversary \(\mathcal{A}\), there exists a probabilistic machine \(B_{2(m+1)}\), whose running time is essentially the same as that of \(\mathcal{A}\), such that for any security parameter \(\lambda\),
\[
|\text{Adv}_{\mathcal{A}}^{(2(m+1))}(\lambda) - \text{Adv}_{\mathcal{A}}^{(2(m+1))}(\lambda)| \leq \text{Adv}_{B_{2(m+1)}}^{P_2}(\lambda) + (|L| + 4)/q.
\]
Proof. Suppose a polynomial time adversary $A$ can successfully distinguish between Game 2-$m'$ and Game 2-$(m + 1)$. We construct a simulator $B_{2(m+1)}$ that leverages $A$ as a black box to solve Problem 2. The procedure is shown as follows:

1. $B_{2(m+1)}$ is given an instance of Problem 2, i.e., $(\text{param}_{2}, \hat{B}^{(0)}, \hat{B}^{*(0)}, \hat{h}^{*(0)}, t^{(0)}, \{\hat{h}^{*(k)}, \hat{B}^{*(k)}, i^{(k)}\}_{i=1,\ldots,n_k})_{k=1,2}$ and plays the role of the challenger in the security game against adversary $A$.

2. At the beginning of the game, $B_{2(m+1)}$ gives $A$ the public key $PK = (1^\lambda, \text{param}_{2}, (b_1^{*(0)}, b_2^{*(0)}, b_3^{*(0)}, h_1^{*(0)}, h_2^{*(0)}, h_3^{*(0)})_{i=1,2,j=1,\ldots,n_i})$ of the problem 2 instance.

3. When the $s$-th $\text{GenKey}$ query is issued for a predicate $\overline{z} = (x_1, \ldots, x_n)$, $B_{2(m+1)}$ answers as follows:

   a) For $1 \leq s \leq m$, $B_{2(m+1)}$ computes a semi-functional key using $(\hat{B}^{*(k)})_{k=0,1,2}$ of the problem 2 instance.

   b) For $s = m+1$, $B_{2(m+1)}$ computes $k_{\overline{z},t}$, where $k_{\overline{z},t} = (I, k_0, k_1, k_2)$ using $(h_1^{*(0)}, b_1^{*(0)}, b_2^{*(0)}, b_3^{*(0)}, (h_1^{*(i)}, h_2^{*(i)}, h_3^{*(i)})_{i=1,2,j=1,\ldots,n_i}; \{b_j^{*(i)}\}_{i=1,2,j=1,\ldots,2n})$ of the problem 2 instance as follows:

   
   $$\text{For } i = 1, 2: g_i, v_i, v_i', \theta_i \in \mathbb{F}_q;$$

   $$s_{\beta,0} = \sum_{i=1}^{2}(g_i h_\beta^{*(0)} + v_i b_1^{*(0)}), \quad k_0 = -s_{\beta,0} + \epsilon' b_2^{*(0)} + b_3^{*(0)},$$

   For $i = 1, 2$ and $j = 1, \ldots, n$:

   $$s_{\beta,j} = \theta h_\beta^{*(i)} + v_i b_1^{*(i)}, \quad s_{\beta,j} = g_i h_\beta^{*(i)} + v_i b_1^{*(i)},$$

   $$k_1 = \sum_{j=1}^{n} x_j s_{\beta,j} + s_{\beta,1} + \sum_{j=1}^{n_i} v_j b_{1,i+j},$$

   $$k_2 = s_{\beta,1} + I \cdot s_{\beta,2} + \sum_{j=1}^{2} v_j b_{2,j}^{*(2)},$$

   where $\epsilon' \in \mathbb{F}_q$, $v_1^{(2)}, v_2^{(2)} \in \mathbb{F}_q$, $(v_1^{(1)}, \ldots, v_{n_1}^{(1)}) \in \mathbb{F}_q^{n_1} \setminus \{0\}$.

   c) For $s \geq m+2$, $B_{2(m+1)}$ computes a normal key using $(\hat{B}^{*(k)})_{k=0,1,2}$ of the problem 2 instance.

4. When $B_{2(m+1)}$ receives challenge attribute vectors $(\overline{y}_1^{(0)}, \overline{y}_1^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$ and a revocation list $L$ from $A$, $B_{2(m+1)}$ computes and returns $C = (L, c_0, c_1, \{c_r\}_{r \in L}, c_M)$ s.t. $c_0 = t^{(0)} + \zeta b_3^{(0)} + \varphi b_5^{(0)}$, $c_1 = \sum_{j=1}^{n_1} y_j^{(0)} t^{(0)} + \varphi b_1^{(0)}$, $c_0 = \sum_{j=1}^{n_1} y_j^{(0)} t^{(0)} + \varphi b_1^{(0)}$, $c_M = \sum_{r \in L} c_r$, using $(t^{(0)}, \{t_1^{(0)}\}_{i=1,\ldots,n_i}, t_2^{(0)}, t_3^{(0)}, t_4^{(0)}, t_5^{(0)}, t_6^{(0)}, t_7^{(0)}, t_8^{(0)}, t_9^{(0)}, t_{10}^{(0)}, t_{11}^{(0)}, t_{12}^{(0)}, t_{13}^{(0)}, t_{14}^{(0)}, t_{15}^{(0)})$ in the Problem 2 instance, $\overline{y}^{(b)}$, $M^{(b)}$ and $L$, where $p, \varphi, \zeta \in \mathbb{F}_q$ for all $r \in L$ such that $p = \sum_{r \in L} p_r$, $\zeta, \varphi, \varphi^{(1)} \in \mathbb{F}_q$, $\varphi \in \{0, 1\}$.

5. After the challenge phase, $\text{GenKey}$ oracle simulation for a key query is executed in the same manner as step 3.
6. $A$ outputs a bit $b'$. If $b = b'$, $B_{2(m+1)}$ outputs 1. Otherwise, $B_{2(m+1)}$ outputs 0.

**Claim 4.3.** The distribution of the view of adversary $A$ in the above-mentioned game simulated by $B_{2(m+1)}$ given a Problem 2 instance with $\beta \in \{0, 1\}$ is the same as that in Game 2-$(m+1)$ (resp. Game 2-$m'$) if $\beta = 0$ (resp. $\beta = 1$) except with probability $(3 + |L|)/q$ (resp. $1/q$).

**Proof.** It is clear that $B_{2(m+1)}$’s simulation of the public key generation (step 2) and the answers to the $i$-th GenKey query where $i \neq m+1$ (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and the GenKey oracles in Game 2-$(m+1)$ and Game 2-$m'$.

Next we analyze the distribution of the $i$-th GenKey query where $i = m+1$ (case (b) of steps (3) and (5)). Values $s_{0}^{(i)}, s_{1}^{(i)}, z_{0,i}^{(j)}$ for $i = 1, 2$ and $j = 1, \ldots, n_i$ in this case can be expressed as follows. Let $\beta^{(i)} = \theta_{i} \omega + v_{i}, \; \alpha^{(i)} = \varphi_{i} \omega + v_{i}, \; \alpha = \alpha^{(1)} + \alpha^{(2)}, \; \gamma = \varphi_{1} + \varphi_{2}$. Then,

$$s_{0}^{(i)} = (\alpha, 0, \gamma \xi, 0)_{s_{2}^{(i)},} \quad s_{1}^{(i)} = (\alpha, \epsilon, 0, \gamma \xi, 0)_{s_{2}^{(i)},}$$

$$z_{0,i}^{(j)} = (\beta^{(i)} \tau_{j,i}^{(i)}, \theta_{j,i}^{(i)}, \theta_{j,i}^{(i)} \tau_{j,i}^{(i)}, 0)_{s_{2}^{(i)},} \quad z_{1,i}^{(j)} = (\beta^{(i)} \tau_{j,i}^{(i)}, \theta_{j,i}^{(i)}, \theta_{j,i}^{(i)} \tau_{j,i}^{(i)}, 1)_{s_{2}^{(i)},},$$

where $\tau_{j,i}^{(i)} = \tau_{j,i}^{(1)}, \ldots, \tau_{j,i}^{(i)} \omega, \xi, \{\tau_{j,i}^{(1)}, \tau_{j,i}^{(i)}\}_{i=1,2,j=1,\ldots,n_i}$ are defined in Problem 2. When $\beta = 1$ in Problem 2 instance, $k_{P1} = (I, k_0, k_1, k_2)$ has the same distribution as Eq. 4.5 except that $(\gamma^{(1)} \tau_{1}^{(1)} + \sigma^{(1)} \tau'_{1}) \cdot Z^{(1)} + \tau'_{1}$ and $\gamma^{(2)}(1, I) \cdot Z^{(2)} + \tau'_{2}$, where $\tau'_{1} \equiv \sigma^{(1)} Z^{(1)} + \tau'_{1} \equiv \sigma^{(2)} Z^{(2)} + \tau'_{2}$.

Next, we show that the joint distribution of the answer to $(m+1)$-th GenKey query and the challenge ciphertext by $B_{2(m+1)}$’s simulation given a Problem 2 instance with $\beta$ is equivalent to that in Game 2-$(m+1)$ (resp. Game 2-$m'$), when $\beta = 0$ (resp. $\beta = 1$).

When $\beta = 0$, it is straightforward to show that they are equivalent except that one of following conditions holds: (1) $\omega$ defined in Problem 2 is zero, (2) $w = 0$, (3) $(w^{(1)}_1, \ldots, w^{(1)}_n) = \emptyset$, (4) $(w^{(2)}_1, w^{(2)}_2) = \emptyset$, where $r \in L$, and $w$, $(w^{(1)}_1, \ldots, w^{(1)}_n)$ and $(w^{(2)}_1, w^{(2)}_2)$ are defined in Eqs. 4.3, 4.2 and 4.3 respectively. Those events occur with probability $(3 + |L|)/q$.

When $\beta = 1$, $B_{2(m+1)}$’s simulation for the key is the same as that expressed in Eq. 4.5 and $B_{2(m+1)}$’s simulation for the the challenge ciphertext is the same as that expressed in Eq. 4.6 except that $(\gamma^{(1)} \tau_{1}^{(1)} + \sigma^{(1)} \tau'_{1}) \cdot Z^{(1)} + \tau'_{1}$ and $\gamma^{(2)}(1, I) \cdot Z^{(2)} + \tau'_{2}$, where $\tau'_{1} \equiv \sigma^{(1)} Z^{(1)} + \tau'_{1} \equiv \sigma^{(2)} Z^{(2)} + \tau'_{2}$.

Therefore, we will show that $(\gamma^{(1)} \tau_{1}^{(1)} + \sigma^{(1)} \tau'_{1}) \cdot Z^{(1)} + \tau'_{1}$ and $\gamma^{(2)}(1, I) \cdot Z^{(2)} + \tau'_{2}$ are uniformly and independently distributed from the other variables in $B_{2(m+1)}$’s simulation. Let $\tilde{A}_1 = (\gamma^{(1)} \tau_{1}^{(1)} + \sigma^{(1)} \tau'_{1}) \cdot Z^{(1)} + \tau'_{1}$, $\tilde{A}_2 = \gamma^{(2)}(1, I) \cdot Z^{(2)} + \tau'_{2},$ and $\tilde{B}_1 = \tau'_{1}, \tilde{U}^{(1)}, \tilde{B}_2 = \tau'_{1}, \tilde{U}^{(2)}$ for all $r \in L$, where $b \in \{0, 1\}$. We show the joint distribution of these variables in the two cases that are defined in Definition 4.2.

1. When $I \in L$, due to Lemma 4.9 the pair $\tilde{A}_2, \tilde{B}_{2,r}$ is uniformly and independently distributed over $F_q^4$.

2. When $I \notin L$ and $f_\tau(\tilde{U}^{(0)}) = f_\tau(\tilde{U}^{(1)}) = 0$, due to Lemma 4.9 the pair $\tilde{A}_1, \tilde{B}_1$ is
uniformly and independently distributed over $\mathbb{F}_q^{2n_1}$, while the pair $(\overrightarrow{A}_2, \overrightarrow{B}_{2,r})$ (for $r \in L$) is uniformly and independently distributed over $\mathbb{F}_q^4$.

Due to the restriction of adversary $\mathcal{A}$’s key queries defined in Definition 4.2 in each case, at least one of $(\overrightarrow{A}_1, \overrightarrow{B}_1)$ and $(\overrightarrow{A}_2, \overrightarrow{B}_{2,r})$ is uniformly and independently distributed over $\mathbb{F}_q^{2n_k}$ for $k = 1, 2$. Therefore, $(\gamma(1)\overrightarrow{r}_1 + \sigma(1)\overrightarrow{r}^*_2, Z(1) + \overrightarrow{v}_1$ and $(\gamma(2)(1, I)\cdot Z(2) + \overrightarrow{v}_2$ are uniformly and independently distributed from the other variables in $B_{2(m+1)}$’s simulation.

Therefore, the view of the adversary $\mathcal{A}$ in the game simulated by $B_{2(m+1)}$ given an instance of Problem 2 with $\beta = 1$ is the same as that in Game 2-$m'$ unless $\omega = 0$. This event occurs with probability $1/q$.

This completes the proof of Lemma 4.12.

Lemma 4.13. For any adversary $\mathcal{A}$, $\text{Adv}^{(3)}_\mathcal{A} (\lambda) \leq \text{Adv}^{(2,\nu)}(\lambda) + 2/q$.

Proof. First we show the distribution $\text{param}_{\mathcal{A}} \{B((k))\}_{k=0,1,2}$, $\{k_{\mathcal{A}, j}((k))\}_{j=1, \ldots, \nu, C}$ of Game 3 is same as that of Game 2-$\nu$, where $k_{\mathcal{A}, j}((k))$ is the answer to the $j$-th key query, and $C$ is the challenge ciphertext. We will define new bases $\mathbb{B}((k))$ of $\mathbb{V}_k$ and $\mathbb{B}^*(k)$ of $\mathbb{V}_k$, $k = 0, 1, 2$.

For $k = 0$, we set $d_{1}^{(k)} = b_{1}^{(k)} - \lambda b_{2}^{(0)}$ and $d_{2}^{(0)} = b_{3}^{(0)} + \lambda b_{4}^{(0)}$, where $\lambda \not\in \mathbb{F}_q$. The new bases are $\mathbb{D}((0)) = (b_{1}^{(0)}, b_{2}^{(0)}, b_{3}^{(0)}, b_{4}^{(0)})$ and $\mathbb{D}^*(0) = (b_{1}^{(0)}, b_{2}^{(0)}, d_{3}^{(0)}, b_{4}^{(0)}, b_{5}^{(0)})$. We can easily verify that $\mathbb{D}(0)$ and $\mathbb{D}^*(0)$ are dual orthonormal, and are distributed the same as the original bases $\mathbb{B}(0)$ and $\mathbb{B}^*(0)$ respectively.

For $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_1$, choose $Q^{(i)} = (\mu^{(i)}_1) \leftarrow \mathbb{F}_q^{n_1 \times n_1}$, and compute $d_{1n_1+i}^{(1)} = b_{1n_1+i}^{(1)} + \sum_{j=1}^{n_1} \mu^{(i)}_j b_{1n_1+j}^{(1)}$, $d_{1n_1+i}^{(1)} = b_{1n_1+i}^{(1)} - \sum_{j=1}^{n_1} \mu^{(i)}_j b_{1n_1+j}^{(1)}$, which are equivalent to the following matrix computations:

\[
\begin{pmatrix}
\overrightarrow{B}_1^{(1)} \\
\overrightarrow{B}_2^{(1)}
\end{pmatrix}
= \begin{pmatrix}
I_{n_1} & 0_{n_1} \\
Q^{(1)} & I_{n_1}
\end{pmatrix}
\begin{pmatrix}
\overrightarrow{B}_1^{(1)} \\
\overrightarrow{B}_2^{(1)}
\end{pmatrix}
= \begin{pmatrix}
I_{n_1} & -Q^T^{(1)} \\
0_{n_1} & I_{n_1}
\end{pmatrix}
\begin{pmatrix}
\overrightarrow{B}_1^{(1)} \\
\overrightarrow{B}_2^{(1)}
\end{pmatrix},
\]

where $\overrightarrow{B}_1^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$, $\overrightarrow{B}_2^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$, $\overrightarrow{B}_1^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$, $\overrightarrow{B}_2^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$, $\overrightarrow{B}_1^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$, $\overrightarrow{B}_2^{(1)} = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})^T$.

The new bases are $\mathbb{D}((1)) = (b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)}, b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)}, b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)}, b_{1n_1+1}^{(1)}, \ldots, b_{2n_1}^{(1)})$. It is clear that $\mathbb{D}(1)$ and $\mathbb{D}^*(1)$ are dual orthonormal, and are distributed the same as the original bases $\mathbb{B}(1)$ and $\mathbb{B}^*(1)$ respectively.

We also set $\mathbb{B}(2) = \mathbb{B}(2)$ and $\mathbb{D}(2) = \mathbb{B}^*(2)$.

The secret keys and the challenge ciphertext $\{k_{\mathcal{A}, j}((k))\}_{j=1, \ldots, \nu, C}$ in Game 2-$\nu$ can be expressed over bases $\mathbb{B}((k))$ and $\mathbb{B}^((k))$, $k = 0, 1, 2$ as follows:

\[
k_{0,j} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{\mathbb{B}^*(0)},
k_{1,j} = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(1)},
k_{2,j} = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(2)},
\]

\[
k_{0,j}^* = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{\mathbb{B}^*(0)},
k_{1,j}^* = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(1)},
k_{2,j}^* = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(2)},
\]

\[
k_{0,j}^{**} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{\mathbb{B}^*(0)},
k_{1,j}^{**} = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(1)},
k_{2,j}^{**} = \left(\begin{array}{c}
\widetilde{\alpha}_j, \epsilon_j, 1, \eta_j, 0
\end{array}\right)_{\mathbb{B}^*(2)}.
\]
\[ c_0 = (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}(0)}, \]
\[ c_1 = (\delta y, w_1^{(1)}, \ldots, w_n^{(1)}, \eta, \varphi_{(1)})_{\mathbb{B}(1)}, \]
\[ \forall r \in L: c_r = (\delta_r(-r, 1), w_{1,r}^{(2)}, w_{2,r}^{(2)}, 0^2, \varphi_r)_{\mathbb{B}(2)}, \]
\[ c_M = g^j M_j^{(b)}. \]

These keys and the challenge ciphertext can also be expressed over bases \( D^{(k)} \) and \( D^{*(k)} \) for \( k = 0, 1, 2 \) as follows. The first component of each key can be represented as \( k_{0,j} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{\mathbb{D}^{(0)}} = (-\alpha_j, \theta_j, 1, \eta_j, 0)_{\mathbb{D}^{*(0)}}, \) where \( \theta_j = \epsilon_j - \lambda \) are uniform and independent due to \( \epsilon_j \leftarrow \mathbb{F}_q \). The remaining key components can be expressed in a similar way, i.e.

\[ k_{1,j} = \left( s_{1,j} \cdot \epsilon_1^{(1)} + \beta_j^{(1)} \cdot x_j, v_{1,j}, \ldots, v_{n_{1,j}}, \eta_{1,j}, \ldots, \eta_{n_{1,j}}, 1, 0 \right)_{\mathbb{D}^{(1)}} \]
\[ = \left( s_{1,j} \cdot \epsilon_1^{(1)} + \beta_j^{(1)} \cdot x_j, \mu_{1,j} \cdot s_{1,j}, \eta_{1,j}, \ldots, \eta_{n_{1,j}}, 1, 0 \right)_{\mathbb{D}^{*(1)}}, \] and

\[ k_{2,j} = \left( s_{2,j} \cdot (1, 1), v_{1,j}, \ldots, v_{n_{1,j}}, \eta_{1,j}, \ldots, \eta_{n_{1,j}}, 1, 0 \right)_{\mathbb{D}^{(2)}} \]
\[ = \left( s_{2,j} \cdot (1, 1), \mu_{1,j} - 1, \eta_{1,j}, \ldots, \eta_{n_{1,j}}, 1, 0 \right)_{\mathbb{D}^{*(2)}}, \] where \( \theta_{i,j} = \mu_{i,j} \cdot s_{1,j} + \beta_j^{(1)} \cdot x_j, \mu_{1,j}^{(1)} + v_{i,j}, i = 1, \ldots, n_1, j = 1, \ldots, \nu \) are uniformly distributed since \( v_{i,j}^{(1)} \leftarrow \mathbb{F}_q \).

The first component of the ciphertext can be expressed as \( c_0 = (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}(0)} = (\delta, w, \zeta', 0, \varphi)_{\mathbb{B}^{(0)}}, \) where \( \zeta' = \zeta + \lambda w \) is uniformly distributed since \( w, \zeta \leftarrow \mathbb{F}_q \). Similarly, other components of the ciphertext can be represented as

\[ c_1 = \left( \delta y, w_1^{(1)}, \ldots, w_n^{(1)}, 0^n, 1 \right)_{\mathbb{B}(1)}, \]
\[ \forall r \in L: c_r = (\delta_r(-r, 1), w_{1,r}^{(2)}, w_{2,r}^{(2)}, 0^2, \varphi_r)_{\mathbb{B}(2)}, \]
\[ c_M = g^j M_j^{(b)}. \]
where \( y' = (y'_1, \ldots, y'_{n_1}) \), \( y'_i = \delta y_{i}^{(b)} - \sum_{j=1}^{n_1} w_{j}^{(1)} \mu_{j,i}^{(1)}, i = 1, \ldots, n_1 \), are uniform and independent due to \((w_{1}^{(1)}, \ldots, w_{n_1}^{(1)}) \sim \mathbb{F}_{q}^{n_1}\).

In the light of the adversary’s view, both \((B^{(k)}, B^{*(k)})\) and \((D^{(k)}, D^{*(k)})\) for \(k = 0, 1, 2\) are consistent with public key \((1^\lambda, \text{param}, \{\hat{B}^{(k)}\}_{k=0,1,2})\). Therefore, \(\{k_{x,I}^{(j)}\}_{j=1,\ldots,\nu}\) and \(C\) can be expressed as keys and ciphertext in two ways, in Game 2-\(\nu\) over bases \((B^{(k)}, B^{*(k)})\) and in Game 3 over bases \((D^{(k)}, D^{*(k)})\). Thus, Game 2-\(\nu\) can be conceptually changed to Game 3 if \(w \neq 0\) and \((w_{1}^{(1)}, \ldots, w_{n_1}^{(1)}) \neq 0\), i.e., except with probability \(2/q\).

\[\square\]

\textbf{Lemma 4.14.} For any adversary \(A\), \(\text{Adv}_{A}^{(3)}(\lambda) = 0\).

\textit{Proof.} The value of \(b\) is independent from the adversary’s view in Game 3. Therefore, \(\text{Adv}_{A}^{(3)}(\lambda) = 0\). \(\square\)

\textbf{4.5 Conclusion}

We introduce the concept of revocable predicate encryption (RPE), which extends the previous PE setting with revocation support: private keys can be used to decrypt an RPE ciphertext only if they match the decryption policy (defined via attributes encoded into the ciphertext and predicates associated with private keys) and were not revoked by the time the ciphertext was created. We formalize the notion of attribute hiding in the presence of revocation and propose our scheme, called AH-RPE, which is attribute-hiding under the Decision Linear assumption in the standard model. In the next chapter, we will investigate an RPE with stronger security, namely revocable predicate encryption with full hiding.
Chapter 5

Revocable Predicate Encryption with Full Hiding

In this chapter, we continue our study in RPE. We propose a RPE with stronger security, namely RPE with full hiding. We first provide a security definition which is specific to our scheme. We then propose RPE with full hiding property. The scheme achieves logarithmic complexity for the sizes of key and ciphertext. We prove that our scheme is secure under the Decision Linear assumption in the standard model.

5.1 Introduction

We continue our investigation in the revocable PE. In AH-RPE, ciphertexts do not hide information about revoked key indexes and it suits applications where revocation lists are managed and published by the master authority. However, in addition to attribute hiding, we may require that ciphertexts do not leak any information about revoked indexes (so-called full hiding). This privacy goal becomes especially relevant when key indexes can be linked to users and whenever senders wish to exclude certain users from decryption — the latter concept of sender-local revocation (SLR) allows senders to define revocation lists (per ciphertext) during the encryption process and by this flexibly refine access control to encrypted data. The SLR property is particularly useful for broadcast systems, e.g. in Pay-TV, where the sender distributes the content and also manages revocation lists and keys (e.g. so-called target broadcast system from \[28\]). The sender could locally revoke certain customers (e.g. those in delay with payment) for a number of transmissions. The full hiding property is also relevant for applications where revocation lists are sensitive. Consider an illustrative example, where some intelligence agency may want to broadcast confidential information to all agents with certain attributes and yet still exclude James from accessing the information, i.e. irrespective of whether James’ key satisfies the policy of the ciphertext. The full hiding property would effectively hide the fact that James’ decrypting rights were revoked for that particular ciphertext, even from James...
himself, who wouldn’t know whether decryption failed because of revocation or due to a policy mismatch between his key and the ciphertext.

5.1.1 Contributions of this chapter

FULL-HIDING RPE SCHEME. The proposed scheme, termed FH-RPE, offers stronger privacy guarantees than AH-RPE. We introduce a strong notion of full-hiding, which we formalize as part of the RPE security model. The scheme ensures that no information about revoked users is leaked from a given ciphertext and is a natural extension in the context of PE that cares about privacy. Our FH-RPE scheme can be used in applications where senders may freely decide to exclude certain key holders from running a successful decryption operation, even if private keys of those holder match the ciphertext policy. Such sender-local revocation (SLR) allows for more flexible forms of access control to PE plaintexts and the requirement of full-hiding keeps revoked recipients undisclosed. The full-hiding property of FH-RPE also relies on the DLIN assumption; yet this stronger privacy property comes with additional performance overhead in comparison to our AH-RPE scheme in that the length of keys and ciphertexts becomes logarithmic in the number of decryption keys.

TECHNIQUES. Our FH-RPE scheme is based on the Dual System Encryption of Waters [61] and the Dual Pairing Vector Spaces (DPVS) of Okamoto and Takashima [42]. Our scheme is obtained from Okamoto and Takashima [43] and Lewko et al. [35] in a more direct way: Each private key corresponds to an index, which is defined at derivation time. Indexes of revoked keys are used to build the revocation list. The revocation mechanism extends individual private keys with additional index-dependent components that provide decryption capabilities as long as the index remains unrevoked — revoked indexes are encoded by the sender into the ciphertext in a privacy-preserving way. We note that to be able to create a ciphertext, the sender needs to know not only an attribute but also the indexes of the revoked keys. The use of indexes is essential in our scheme. If a sender wishes to revoke some particular key from decrypting a ciphertext then some identifier for that key must be used in the encryption procedure; otherwise revoking certain keys becomes impossible since the keys would not be distinguishable from other keys with the same predicates. We first discuss that applying Okamoto and Takashima [43] and Lewko et al. [35] directly would result in a linear complexity for the lengths of main parameters. The (better) logarithmic complexity of our FH-RPE scheme is due to the use of the complete-subtree technique by Naor et al. [40], whose integration preserving the full-hiding property was a challenge. We note that it is possible to obtain a FH-RPE with the combinations of a PE [35] and an anonymous broadcasting scheme [38]. However, in the standard model, the size of the ciphertext in the resulting system is linear in the number of keys. In order to prove security of our RPE schemes we utilize the modular approach from Okamoto and Takashima [43] that breaks the proof down into several higher-level (artificially looking) assumptions and proves them to be secure under the DLIN assumption.
5.2 Security Model

5.2.1 Security Definition

**Definition 5.1** (Full-Hiding RPE). An RPE scheme, as defined in definition 4.1, is adaptively full hiding against chosen plaintext attacks if for all PPT adversaries $A$, the advantage $\text{Adv}_{A,RPE}^{\text{FH}}(\lambda)$ of $A$ in the following game is negligible in the security parameter $\lambda$:

**Setup.** A challenger $C$ runs the Setup algorithm to generate a public key $PK$, a master secret key $MSK$, and $S$. $PK$ is given to $A$.

**Query phase 1.** $A$ adaptively makes a polynomial number of GenKey queries: $A$ produces a predicate $\overrightarrow{x}$, $C$ computes the key $k_{\overrightarrow{x},I} \xleftarrow{\text{R}} \text{GenKey}(MSK,S,\overrightarrow{x})$ associated with an index $I$, and gives it to $A$.

**Challenge.** $A$ outputs challenge attribute vectors $(\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)})$, challenge revocation lists $(L^{(0)}, L^{(1)})$, and challenge plaintexts $(M^{(0)}, M^{(1)})$, subject to one of the following restrictions for each queried key $k_{\overrightarrow{x},I}$:

1. $f_{\overrightarrow{x}}(\overrightarrow{y}^{(0)}) = f_{\overrightarrow{x}}(\overrightarrow{y}^{(1)}) = 0$
2. $f_{\overrightarrow{x}}(\overrightarrow{y}^{(0)}) = f_{\overrightarrow{x}}(\overrightarrow{y}^{(1)}) = 1$ and $(I \in L^{(0)} \land I \in L^{(1)})$
3. $(f_{\overrightarrow{x}}(\overrightarrow{y}^{(0)}) = 1 \land f_{\overrightarrow{x}}(\overrightarrow{y}^{(1)}) = 0)$ and $I \in L^{(0)}$
4. $(f_{\overrightarrow{x}}(\overrightarrow{y}^{(0)}) = 0 \land f_{\overrightarrow{x}}(\overrightarrow{y}^{(1)}) = 1)$ and $I \in L^{(1)}$.

$C$ flips a random coin $b$. If $b = 0$ then $A$ is given $C = \text{Encrypt}(PK,L^{(0)}, \overrightarrow{y}^{(0)}, M^{(0)})$. If $b = 1$ then $A$ is given $C = \text{Encrypt}(PK,L^{(1)}, \overrightarrow{y}^{(1)}, M^{(1)})$.

**Query phase 2.** Repeat the Query phase 1 subject to the restrictions as in the challenge phase.

**Guess.** $A$ outputs a guess $b'$ of $b$, and succeeds if $b' = b$.

The advantage of $A$ is defined to be $\text{Adv}_{A,RPE}^{\text{FH}}(\lambda) = |\text{Pr}[b = b'] - 1/2|$.

**Remark 5.1.** Definition 5.1 can be easily extended to capture chosen-ciphertext attacks (CCA) by allowing decryption queries (for all but the challenge ciphertext). The advantage of $A$ in such CCA game is defined to be $\text{Adv}_{A,RPE}^{\text{FH,CCA}}(\lambda) = |\text{Pr}[b = b'] - 1/2|$. Our CPA-secure FH-RPE from Section 5.3 can be strengthened to resist CCA by applying the well-known CHK transformation from [18] that uses one-time signatures to authenticate the ciphertext.

5.3 Our Scheme

We present our RPE scheme, which achieves the property of full hiding. The scheme is based on Okamoto and Takashima’s FE [43] and the Subset-Cover Framework due to Naor et al. [40].

The intuition behind the new construction is as follows. In addition to having vectors representing predicates and attributes, we have index vectors and revocation vectors. The scheme can be seen as the composition of two encryption steps, one using the attributes and
predicates vectors, and the other one using the index and revocation vectors. Attributes and predicates vectors use a different basis from index and revocation list vectors. By using separate bases, we avoid the quadratic length growth that would otherwise occur if we concatenated the vectors using the same basis. We also use a secret sharing scheme over bilinear pairing groups to combine the components in the secret key, so that it is hard to modify the secret key to make valid a key that has otherwise been revoked.

Each key has an associated index vector $\overrightarrow{x}_I$, which encodes the key index $I$ by assigning a random value to the vector component at the corresponding index position. To revoke a set of keys, the encryptor sets random values on positions in the revocation vector $\overrightarrow{y}_L$ that correspond to the indexes of revoked keys. We see that if $\overrightarrow{y}_L$ has a random value in the $I^{th}$ component, then $\overrightarrow{x}_I \cdot \overrightarrow{y}_L \neq 0$ and results in a random group element on decryption. This indicates that the key with index $I$ is revoked. We assume that both index vector and revocation vector are initially set to $\overrightarrow{0}$. If we denote the number of keys in the system as $N$, then the size of both index vectors and revocation vectors is $O(N)$. If the predicate/attribute vector is of size $n$, then the size of ciphertexts and keys is $O(n + N)$.

The major drawback of such approach is, however, the space cost. To alleviate this limitation, our scheme takes advantage of the complete-subtree data structure from [40]. Informally, in a binary tree with $N$ leaves, the index $I$ of a key will be associated with a leaf node. Each node in the tree will be assigned a unique identity. To compute a key with an index, we compute on identities of all the nodes on the path from the leaf node associated with $I$ to the root node. To encrypt, the sender first finds a minimal set of nodes which contains an anchor (or, the node itself) of all the non-revoked indexes. It then computes ciphertext on the attribute and the identities of all the nodes in that set. To retain the full-hiding property we apply the binary structure in an anonymous setting. Decryption works if there exists one common node (identity) between the key and the ciphertext, which is given for unrevoked keys.

Consider the following illustrative example: in Figure 5.1 (left), indexes 2 and 6 are associated with different keys, say $k_{x_2,2}$ and $k_{x_6,6}$ respectively. The key $k_{x_2,2}$ is computed on the predicate vector $\overrightarrow{x}_2$ and tree nodes $\{ID_1, ID_2, ID_3, ID_4\}$, whereas $k_{x_6,6}$ is computed on predicate vector $\overrightarrow{x}_6$ and nodes $\{ID_1, ID_5, ID_6, ID_7\}$. Assume in Figure 5.1 (right) that $k_{x_2,2}$ is revoked. The minimal subset of nodes covering all other indexes is $\{ID_5, ID_8, ID_9\}$. The ciphertext $c_x$ will thus be computed on nodes $\{ID_5, ID_8, ID_9\}$. Since $c_x$ doesn’t have any common node with revoked key $k_{x_2,2}$, decryption with this key will fail but it will succeed with $k_{x_6,6}$ due to the common node $ID_5$.

A detailed specification of our FH-RPE scheme follows: (note that $G_{ob}$ is defined in Section 4.3)

**Setup**$\langle 1^\lambda, \Delta = (\overrightarrow{n} = (2; n_1, n_2 = 2), N) \rangle$: Perform the following computations:

(param $\overrightarrow{y}, \overrightarrow{b}^{(0)}, \overrightarrow{b}^{(1)}), \overrightarrow{b}^{(2)}, \overrightarrow{b}^{(3)} \rangle \xrightarrow{R} G_{ob}(1^\lambda, \overrightarrow{n})$,

$\overrightarrow{b}^{(0)} = (b_1^{(0)}, b_3^{(0)}, b_9^{(0)}), \overrightarrow{b}^{(1)} = (b_1^{(1)}, \ldots, b_{n_1}^{(1)}, b_{3n_1 + 1}^{(1)}), \overrightarrow{b}^{(2)} = (b_1^{(2)}, b_2^{(2)}, b_7^{(2)}),$
\[\overline{B}^{(0)}(x) = (b_1^{(0)}(x), b_2^{(0)}(x), b_4^{(0)}(x)), \quad \overline{B}^{(1)}(x) = (b_1^{(1)}(x), b_2^{(1)}(x), b_3^{(1)}(x), b_5^{(1)}(x)).\]

Let \( Tree \) be a complete binary tree structure with at least \( N \) leaf nodes, which corresponds to the number of keys in the system. Each node \( x \) in \( Tree \) has unique identity \( ID_x \). Let state information \( S \), which records the assigned indexes \( I \) so far, be an initially empty set.

The output of the algorithm is given by the public key \( PK = (1^k, \text{param}_{\overline{B}}, \{\overline{B}^{(k)}\}_{k=0,1,2}, Tree) \), the master secret key \( MSK = \{\overline{B}^{(k)}\}_{k=0,1,2} \), and the state information \( S \).

\[
\text{GenKey}(MSK, S, \overline{y} = (x_1, \ldots, x_{n_1}) \in \mathbb{F}_q^{n_1} \setminus \{0\}): \text{Choose } \alpha, \eta, \eta_1^{(1)}, \eta_2^{(1)}, \beta^{(1)} \text{ uniformly random in } \mathbb{F}_q, \text{ and choose } \alpha^{(1)}, \alpha^{(2)} \text{ uniformly random in } \mathbb{F}_q \text{ such that } \alpha = \alpha^{(1)} + \alpha^{(2)}; \text{ choose index } I \leftarrow \Gamma \text{ such that } I \notin S, \text{ and set } S = S \cup \{I\}; \text{ then compute:}
\]

\[
k_0 = (-\alpha, 0, 1, \eta, 0)_{\overline{B}^{(0)}},
\]

\[
k_1 = \left( \alpha^{(1)}, \eta^{(1)}, \eta^{(2)}, 0 \right)_{\overline{B}^{(1)}},
\]

\[
\forall x \in \mathbb{P}(I): \quad k_x = \left( \alpha^{(2)}, \beta_x^{(2)}(x), 0, \right)_{\overline{B}^{(2)}},
\]

where \( \beta_x^{(2)}, \eta_{1,x}^{(2)}, \eta_{2,x}^{(2)} \) uniformly random in \( \mathbb{F}_q \). The output is given by \( S \) and the secret key \( \mathbf{k}^{x}_{\overline{B}, I} = (I, k_0, k_1, \{k_x\}_{x \in \mathbb{P}(I)}) \).

(Note that \( I \) is associated with the \( I_{th} \) leaf node in the binary tree. \( \mathbb{P}(I) \) denotes all the nodes on the path from the leaf node \( I \) up to the root node (leaf and root nodes inclusive). The secret key \( \mathbf{k}^{x}_{\overline{B}, I} \) thus contains secrets for all nodes \( ID_x \) on the mentioned path from \( I \) to the root.)

\[
\text{Encrypt}(PK, L, \overline{y} = (y_1, \ldots, y_{n_1}) \in \mathbb{F}_q^{n_1} \setminus \{0\}, M \in \mathbb{Z}_q): \text{Choose } \delta, \zeta, \varphi \text{ uniformly random in } \mathbb{F}_q \text{ and compute:}
\]

\[
c_0 = (\delta, 0, \zeta, \varphi)_{\overline{B}^{(0)}},
\]

\[
c_1 = (\delta \overline{y}, \eta_1^{(1)}, \eta_2^{(2)}, \varphi^{(1)})_{\overline{B}^{(1)}},
\]
\[ \forall x \in \text{RevokeNodes}(\text{Tree}, L) : \quad c_x = (\delta, \delta(-1D_x), 0^\nu, 0^\nu, \varphi^{(2)}_x)^{e(2)} \]
\[ c_M = g_T^M. \]

where \( \varphi^{(2)}_x \leftarrow \mathbb{F}_q. \) Output the ciphertext \( C = (c_0, c_1, \{c_x\}_{x \in \text{RevokeNodes}(\text{Tree}, L)}, c_M). \) (Note, \( \text{RevokeNodes}(\text{Tree}, L) \) outputs a minimal set of nodes which contains an ancestor (or, the node itself) of all the non-revoked indexes. \( c_x \) is then computed on all the identities of the nodes in the set.)

\[ \text{Decrypt}(C, k_{\mathcal{P}, I}^\ast) : \text{Given a ciphertext } C = (c_0, c_1, \{c_x\}_{x \in \text{RevokeNodes}(\text{Tree}, L)}, c_M) \text{ and a secret key } k_{\mathcal{P}, I}^\ast = (I, k_0, k_1, \{k_{x'}\}_{x' \in \mathcal{D}(I)}) \text{ compute} \]
\[ \forall x, x' : \quad M_{x,x'} = g_T^{c_M / e(c_0, k_0)c(e_1, k_1)c(c_x, k_{x'})}. \]

If there exists a pair \((x, x')\) corresponding to the same node in \( \text{Tree} \) and \( \mathcal{P} \cdot \mathcal{F} = 0 \), the decrypted message is \( M = M_{x,x'} \). Otherwise, obtained messages are random with all but negligible probability. Note that it is not necessary to test all pairs of \((x, x')\), i.e., it is possible to mark the level of each node encrypted in the ciphertext (without revealing the node’s identifier) and compute using pairs \((x, x')\) that are located on the same level in the tree. In this way decryption costs can be decreased from \( O((\log N)^2) \) to \( O(\log N) \).

**Correctness.** Let \( C \) and \( k_{\mathcal{P}, I}^\ast \) be as above. If \( \mathcal{P} \cdot \mathcal{F} = 0 \) and \( I \notin L \) then \( M \) can be recovered by computing \( c_M / (e(c_0, k_0)c(e_1, k_1)c(c_x, k_{x'})) \), since
\[ e(c_0, k_0)c(e_1, k_1)c(c_x, k_{x'}) = g_T^{\alpha\delta + \zeta} g_T^{\alpha^{(1)}\delta + \beta^{(1)}\delta \mathcal{P} \mathcal{F} \varphi^{(2)}_x} g_T^{\alpha^{(2)}\delta + \beta^{(2)}\delta (-1D_{x'})} = g_T^{\alpha\delta + \zeta} g_T^{\mathcal{P}} = g_T^{\zeta}. \]

### 5.4 Proof of Security

**Theorem 5.1.** \( \text{FH-RPE} \) is adaptively full hiding against chosen plaintext attacks under the DLIN assumption (provided the restriction in Remark 5.2 holds). For any adversary \( \mathcal{A} \), there exists a probabilistic polynomial time machine \( \mathcal{D} \) such that for any security parameter \( \lambda \),
\[ \text{Adv}_{\mathcal{A}, \text{FH-RPE}}^{\mathcal{D}}(\lambda) \leq (2\nu + 1)\text{Adv}_{\mathcal{D}}^{\text{DLIN}}(\lambda) + \psi \]
where \( \nu \) is the maximum number of \( \mathcal{A} \)'s key queries and \( \psi = (2\log N\nu + 18\nu + \log N + 10)/q. \)

**Remark 5.2.** In the proof of Theorem 5.1 we assume that \( |X^{(0)}| = |X^{(1)}| \), where \( X^{(0)} = \{x | x \in \text{RevokeNodes}(\text{Tree}, L^{(0)}) \} \) and \( X^{(1)} = \{x' | x' \in \text{RevokeNodes}(\text{Tree}, L^{(1)}) \} \). The revocation lists \( L^{(0)} \) and \( L^{(1)} \) are defined in the challenge phase of Definition 5.1. This restriction is necessary to prevent the adversary from trivially distinguishing based on the length of the challenge ciphertext.

**Outline of the Proof of Theorem 5.1** In the proof, the concepts of normal, semi-functional and nominal semi-functional forms are defined similarly as in the proof of Theorem 4.1. A semi-functional secret key \( k^\ast_{\mathcal{P}, I} \) and a semi-functional ciphertext \( C^\ast \) are expressed by Eq. (5.7)
and Eqs. (5.1)–(5.4) respectively. Meanwhile, a nominal semi-functional secret key $k_{\pi_{\gamma}^{-1}}^{\text{nom-semi}}$ and a nominal semi-functional ciphertext $C_{\text{nom-semi}}$ are expressed by Eq. (5.5) and Eq. (5.6) respectively. The theorem is then proved through a sequence from Game 0 (original game) to Game 3 using similar techniques as for Theorem 4.1. First, we prove that the probability difference between Games 0 and 1 is equivalent to the advantage of Problem 1. The difference between Games 2-$m'$ and 2-$m$ is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption). We also show that the difference between Games 2-$m'$ and 2-$(m + 1)$ is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption). In the final step, we show that Game 2-$\nu$ can be conceptually changed to Game 3 where the adversary has zero advantage.

**Definition 5.2 (Problem 1).** Problem 1 is to find bit $\beta$ given $(\text{param}_{\pi_{\gamma}}, \{B^{(k)}, \tilde{B}^{(k)}\})_{k=0,1,2}$, $\{\gamma, \{\gamma_{1,1,1,0,1,0,1,0,0,0,0\}}\}$, $\{\gamma_{1,1,1,0,1,0,1,0,0,0,0\}}$ for $\beta \in \{0, 1\}$ with probability non-negligibly better than by a random guess, where

$\mathcal{G}_{\beta}^{\text{P}1}(1^\lambda, \gamma_{1,1,1,0,1,0,1,0,0,0,0}) = (2; n_1, n_2 = 2), d : (\text{param}_{\pi_{\gamma}}, \mathbb{B}^{(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{(3)}, \mathbb{B}^{(4)}, \mathbb{B}^{(5)}) \xrightarrow{R} \mathcal{G}_{\text{ob}}(1^\lambda, \gamma_{1,1,1,0,1,0,1,0,0,0,0})$

\[ \tilde{\mathbb{B}}^{*}(0) = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}), \]

\[ \tilde{\mathbb{B}}^{*(1)} = (b_7^{(1)}, \ldots, b_n^{(1)}, b_{n_{1}+1}^{(1)}, \ldots, b_{n_{1}+1}^{(1)}), \]

\[ \tilde{\mathbb{B}}^{*(2)} = (b_7^{(2)}, b_8^{(2)}, b_9^{(2)}, b_{10}^{(2)}, b_{11}^{(2)}), \]

\[ \check{\delta}, u, \rho \overset{R}{\leftarrow} F_q, \ sp(0), \ sp(1) = (\check{\delta}, u, 0, 0, \rho)_{B^{(0)}}, \ sp(0) = (\check{\delta}, u, 0, 0, \rho)_{B^{(1)}}, \]

\[ \rho(1) \overset{R}{\leftarrow} F_q, \ sp(1) = (\check{\delta}, u, 0, 0, \rho)_{B^{(2)}}, \]

\[ t_{i,1,1}^{(1)} = \frac{n_1}{\check{\delta}, e^{(1)}, \rho(1)}_{B^{(1)}}, \]

\[ t_{i,1,1}^{(1)} = \frac{n_1}{\check{\delta}, e^{(1)}, \rho(1)}_{B^{(2)}}, \]

For $i = 2, \ldots, n_1$:

\[ t_{i,1,1}^{(1)} = \delta b_{i}^{(1)}; \]

For $j = 1, \ldots, d$:

\[ t_{i,1,1}^{(2)} = \frac{2}{\check{\delta}, e^{(1)}, \rho(2)}_{B^{(2)}}, \]

\[ t_{i,1,1}^{(2)} = \frac{2}{\check{\delta}, e^{(2)}, \rho(2)}_{B^{(2)}}, \]

\[ t_{i,1,1}^{(2)} = \delta b_{i}^{(2)}. \]

The corresponding advantage of PPT algorithm $B$ in solving Problem 1 is defined as follows:

\[ \text{Adv}_{B}^{\text{P}1}(\lambda) = \left| \Pr[B(1^\lambda, \gamma_{1,1,1,0,1,0,1,0,0,0,0}) \rightarrow 1] - \Pr[B(1^\lambda, \gamma_{1,1,1,0,1,0,1,0,0,0,0}) \rightarrow 1, \gamma_{1,1,1,0,1,0,1,0,0,0,0,0,0} \leftarrow \mathcal{G}_{\beta}^{\text{P}1}(1^\lambda, \gamma_{1,1,1,0,1,0,1,0,0,0,0,0,0})] \right|. \]

Problem 1 is similar as that defined in Chapter 4, except that there is an extra $d$ in the input parameter in the problem.
Lemma 5.1. For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter $\lambda$, $\text{Adv}^{P_1}(\lambda) \leq \text{Adv}^{\text{DLIN}}(\lambda) + 8/q$.

Definition 5.3 (Problem 2). Problem 2 is to find bit $\beta$ given $(\text{param}_2^{(0)}, \mathbb{B}^{(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{(2)})$, $t^{(0)}$, $\{\hat{\mathbb{B}}^{(k)}, \mathbb{B}^{(k)}, \{h_{\beta,i}^{(k)}, t_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2}$ for $\beta \in \{0, 1\}$ with probability non-negligibly better than by a random guess, where

$$G^{P_2}_2(1^\lambda, \eta) = (2; n_1, n_2 = 2) : (\text{param}_2^{(0)}, \mathbb{B}^{(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{(2)}) \xleftarrow{R} G_\text{ob}(1^\lambda, \eta),$$

$$\hat{\mathbb{B}}^{(0)} = (b_1^{(0)}, b_2^{(0)}, b_3^{(0)}),$$

$$\hat{\mathbb{B}}^{(1)} = (b_1^{(1)}, \ldots, b_{n_1}^{(1)}, b_{3n_1+1}^{(1)}, \ldots, b_{3n_k+1}^{(1)}),$$

$$\hat{\mathbb{B}}^{(2)} = (b_1^{(2)}, b_2^{(2)}, b_3^{(2)}),$$

$$\omega, \xi, \delta \leftarrow \mathbb{F}_q, \ z, \pi \leftarrow \mathbb{F}_q^{\times}, \ u = z^{-1},$$

$$h_0^{(0)} = (\omega, 0, 0, 0)_{\mathbb{B}^{(0)}},$$

$$h_0^{(0)} = (\omega, z, 0, \xi, 0)_{\mathbb{B}^{(1)}},$$

$$t^{(0)} = (\delta, \pi u, 0, 0, 0)_{\mathbb{B}^{(2)}},$$

For $k = 1, 2$:

For $i = 1, \ldots, n_k$ and $j = 1, \ldots, n_k$:

$$(u_{i,j}^{(k)}) \leftarrow GL(\mathbb{F}_q, n_k), \ (z_{i,j}^{(k)}) = ((u_{i,j}^{(k)})^{-1})^T,$$

For $i = 1, \ldots, n_k$:

$$\omega_i^{(k)} \leftarrow \mathbb{F}_q^{n_k},$$

$$h_{0,i}^{(k)} = (\omega_i^{(k)}, 0^{n_k}, \omega_i^{(k)}, 0)_{\mathbb{B}^{(k)}},$$

$$h_{1,i}^{(k)} = (\omega_i^{(k)}, z_{i,1}^{(k)}, \ldots, z_{i,n_k}^{(k)}, \omega_i^{(k)}, 0)_{\mathbb{B}^{(k)}},$$

$$t_i^{(k)} = (\delta, \pi u_i^{(k)}, \ldots, \pi u_i^{(k)}, 0^{n_k}, 0)_{\mathbb{B}^{(k)}},$$

return $(\text{param}_2^{(0)}, \mathbb{B}^{(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{(2)}, \{h_{\beta,i}^{(k)}, t_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2}$.

Let $\mathcal{B}$ be a probabilistic machine, we define the advantage of $\mathcal{B}$ for Problem 2 as follows:

$$\text{Adv}^{P_2}_{\mathcal{B}}(\lambda) = \left| \Pr_{\mathbb{B}(1^\lambda, \tau) \rightarrow 1} \left[ \omega \xleftarrow{R} G_0^{P_2}(1^\lambda, \eta) \right] - \Pr_{\mathbb{B}(1^\lambda, \tau) \rightarrow 1} \left[ \omega \xleftarrow{R} G_0^{P_2}(1^\lambda, \eta) \right] \right|. $$

Note that Definition 5.3 is identical to the definition of Problem 2 in Chapter 4. We repeat it here for completeness.

Lemma 5.2. For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter $\lambda$, $\text{Adv}^{P_2}_{\mathcal{B}}(\lambda) \leq \text{Adv}^{\text{DLIN}}(\lambda) + 5/q$.

Proof of Lemma 5.1 and 5.2 In order to reduce the DLIN problem to Problems 1 and 2 from Definitions 5.2 and 5.3 respectively, we further introduce three “basic problems” that will serve in intermediate steps of the reduction:

- Basic Problem 0 in Definition 5.4
5.4. Proof of Security

- Basic Problem 1 in Definition 5.3
- Basic Problem 2 in Definition 5.6

In order to prove Lemmas 5.1 and 5.2 we use two intermediate lemmas (Lemmas 5.3 and 5.4) that are common lemmas used in the proofs of Lemmas 5.1 and 5.2.

Lemma 5.3. Let \((q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)\) be dual pairing vector spaces by direct product of symmetric pairing groups. Using \{\phi_{i,j}\}, we can efficiently sample a random linear transformation \(W = \sum_{i=1,j=1}^{N,N} r_{i,j} \phi_{i,j}\) of \(\mathbb{V}\) with random coefficients \(\{r_{i,j}\}_{i,j \in \{1,\ldots,N\}} \subseteq \mathbb{G}(N, \mathbb{F}_q)\). The matrix \((r_{i,j}^*) = (r_{i,j}^{-1})^T\) defines the adjoint action on \(\mathbb{V}\) for pairing \(e\), i.e., \(e(W(x), (W^{-1})^T(y)) = e(x, y)\) for any \(x, y \in \mathbb{V}\), where \((W^{-1})^T = \sum_{i=1,j=1}^{N,N} r_{i,j}^* \phi_{i,j}\).

The proof of Lemma 5.3 was given in [43].

Definition 5.4. Basic Problem 0 is to find bit \(\beta\) given \((\text{param}_{\text{BP}0}, \hat{B}, \mathbb{B}^*, y^*_\beta, f, bG, aG, \delta aG) \leftarrow_{R} \mathcal{G}_{\beta}^{\text{BP}0}(1^\lambda)\) for \(\beta \leftarrow_{U} \{0, 1\}\) with probability non-negligibly greater than by a random guess, where

\[
\mathcal{G}_{\beta}^{\text{BP}0}(1^\lambda) : \\
\text{param}_G = (q, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow_{R} \mathcal{G}_{\text{gpg}}(1^\lambda), \\
\text{param}_V = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \leftarrow_{R} \mathcal{G}_{\text{dpps}}(1^\lambda, 3, \text{param}_G), \\
\Lambda = (\lambda_{i,j}) \leftarrow_{U} \mathbb{G}(3, \mathbb{F}_q), (\mu_{i,j}) = (\Lambda^T)^{-1}, \ b, a \leftarrow_{U} \mathbb{F}_q, \\
b_i = b \sum_{j=1}^{3} \lambda_{i,j} a_j, i = 1, 3, \ \hat{B} = (b_1, b_3), \\
b'_i = a \sum_{j=1}^{3} \mu_{i,j} a_j, i = 1, 2, 3, \ B^* = (b'_2, b'_3), \\
g_T = e(G, G)^{ab}, \ \text{param}_{\text{BP}0} = (\text{param}_V, g_T), \\
\delta, \sigma, \omega \leftarrow_{U} \mathbb{F}_q, \ \rho, \tau \leftarrow_{U} \mathbb{F}_q, \\
y^*_\delta = (\delta, 0, \sigma)_{\mathbb{B}^*}, \ y^*_\rho = (\delta, \rho, \sigma)_{\mathbb{B}^*}, \ f = (\omega, \tau)_{\mathbb{B}}, \\
\text{Output} (\text{param}_{\text{BP}0}, \hat{B}, B^*, y^*_\beta, f, bG, aG, \delta aG).
\]

Let \(\text{Adv}_{\mathcal{F}}^{\text{BP}0}(\lambda)\) denote the advantage of a PPT algorithm \(\mathcal{F}\) for the Basic Problem 0.

Lemma 5.4. For any adversary \(\mathcal{F}\), there exists a probabilistic machine \(\mathcal{D}\), whose running time is essentially the same as that of \(\mathcal{D}\), such that for any security parameter \(\lambda\), \(\text{Adv}_{\mathcal{F}}^{\text{BP}0}(\lambda) \leq \text{Adv}_{\mathcal{D}}^{\text{DLIN}}(\lambda) + 5/q\).

The proof of Lemma 5.4 was given in [43].

Proof of Lemma 5.1. Combining Lemmas 5.3, 5.4, 5.5 and 5.6 we obtain Lemma 5.1.

Definition 5.5 (Basic Problem 1). Basic Problem 1 is to find bit \(\beta\) given \((\text{param}_V, \{\mathbb{B}(k)\}, \hat{B}^{(k)}, \{f_0^{(0)}, f_1^{(1)}, f_2^{(2)}\}_{i=2, \ldots, n_1}, f_2^{(1)} \leftarrow_{R} \mathcal{G}_{\beta}^{\text{BP}1}(1^\lambda, \vec{n} = (2; n_1, n_2 = 2))\) for \(\beta \leftarrow_{U} \{0, 1\}\).
\{0, 1\} with probability non-negligibly greater than by a random guess, where

\[
G_{BP}^{B^1} \subseteq (\lambda, \vec{n} = (2; n_1, n_2 = 2)):
\]

\((\text{param}_{\vec{\eta}}, B(0), B^*(0), B(1), B^*(1), B(2), B^*(2)) \overset{R}{\leftarrow} G_{\text{ob}}(1^\lambda, \vec{n})),\)

\[
\tilde{B}^*(0) = (b_1^*(0), b_3^*(0), b_4^*(0), b_5^*(0)),
\]

\[
\tilde{B}^*(1) = (b_1^*(1), \ldots, b_{n_1}^*(1), b_{n_1+1}^*(1), \ldots, b_{3n_1+1}^*(1)),
\]

\[
\tilde{B}^*(2) = (b_1^*(2), b_3^*(2), b_4^*(2), b_5^*(2), b_6^*(2), b_7^*(2)),
\]

\[
\omega, \gamma \overset{U}{\leftarrow} F_q, \quad \tau \overset{U}{\leftarrow} F_q^k, \quad f_0^{(0)} = (\omega, 0, 0, 0, \gamma) \in \mathbb{B}^{(0)}, \quad f_1^{(0)} = (\omega, \tau, 0, 0, \gamma) \in \mathbb{B}^{(1)}, \quad f_1^{(1)} = \omega b_1^{(1)},
\]

\[
\forall k = 1, 2 : \tilde{n}^{(k)} \overset{U}{\leftarrow} F_q^k,
\]

\[
f_{0,1}^{(1)} = \begin{pmatrix} 1 & n_1 \\ \omega e_1^{(1)} & 0 \end{pmatrix} \in \mathbb{B}^{(1)},
\]

\[
f_{1,1}^{(1)} = \begin{pmatrix} 1 & n_1 \\ \omega e_1^{(1)} & 0 \end{pmatrix} \in \mathbb{B}^{(1)},
\]

\[
\forall i = 1, \ldots, n_1 : f_{i}^{(1)} = \omega b_{i}^{(1)},
\]

\[
f_{0,1}^{(2)} = \begin{pmatrix} 2 & 2 \\ \omega, 0 & 0 \end{pmatrix} \in \mathbb{B}^{(1)},
\]

\[
f_{1,1}^{(2)} = \begin{pmatrix} 2 & 2 \\ \omega, 0 & 0 \end{pmatrix} \in \mathbb{B}^{(1)},
\]

\[
f_{2,1}^{(2)} = \begin{pmatrix} 2 & 2 \\ \omega, 0 & 0 \end{pmatrix} \in \mathbb{B}^{(1)},
\]

\[
\text{Output: } \langle \text{param}_{\vec{\eta}}, \{B^{(k)}, \tilde{B}^{(k)}\} \rangle_{k=0,1,2}, f_0^{(0)}, f_{0,1}^{(1)}, f_{1,1}^{(2)}; \{f_i^{(1)}\}_{i=2,\ldots,n_1}, f_2^{(2)} \rangle.
\]

Let \(\text{Adv}_{BP}^B(\lambda)\) denote the advantage of a PPT algorithm \(C\) for the Basic Problem 1.

**Lemma 5.5.** For any adversary \(C\), there exists a probabilistic machine \(F\), whose running time is essentially the same as that of \(C\), such that for any security parameter \(\lambda\), \(\text{Adv}_{BP}^B(\lambda) \leq \text{Adv}_{BP0}(\lambda)\).

**Proof.** \(F\) is given a Basic Problem 0 instance \((\text{param}_{BP0}, \tilde{B}, B^*, y_0, f, bG, aG, acG)\).

With \(\text{param}_C = (q, G, GT, G, c)\) contained in \(\text{param}_{BP0}\), \(F\) computes

\[
\text{param}_{\vec{\eta}_0} = (q, \tilde{V}_0, GT_0, A_0, c) \overset{R}{\leftarrow} G_{\text{ob}}(1^\lambda, 5, \text{param}_C),
\]

\[
\text{param}_{\vec{\tau}_l} = (q, \tilde{V}_l, GT_l, A_l, c) \overset{R}{\leftarrow} G_{\text{ob}}(1^\lambda, 3n_l + 1, \text{param}_C), \quad l = 1, 2,
\]

\[
\text{param}_{\vec{\eta}} = \langle \{\text{param}_{\vec{\tau}_l}\}_{l=0,1,2}, GT \rangle,
\]

where \(GT\) is contained in \(\text{param}_{BP0}\). \(F\) generates random linear transformation \(W_l \in V_l(l = 0, 1, 2)\) given in Lemma 5.3 and sets

\[
d_0^{(0)} = W_0(b_0, 0, 0), \quad l = 1, 2; \quad d_0^{(0)} = W_0(0, 0, 0, 0, aG),
\]

\[
d_2^{(0)} = W_0(b_0, 0, 0, 0, aG, 0), \quad d_2^{(0)} = W_0(b_0^*, 0, 0),
\]

\[
d_2^{(0)} = (W_0)^T(b, 0, 0), \quad l = 1, 2; \quad d_3^{(0)} = (W_0^{-1})^T(0, 0, 0, 0, bG),
\]
\[ d_4^{(0)} = (W_0^{-1})^T(0,0,0,bG,0), \quad d_5^{(0)} = (W_0^{-1})^T(b_3,0,0), \]
\[ g_0^{(0)} = W_0(g_3,0,0), \]
\[ d_4^{(1)} = W_1(b_1,0^{N_1-3}), \quad d_5^{(1)} = W_1(b_2,0^{N_1-3}), \quad d_6^{(1)} = W_1(b_3,0^{N_1-3}), \]
\[ d_7^{(1)} = W_1(0^m,aG,0^{N_1-m-1}) \text{ where } \begin{cases} 
  m = l + 1 \text{ if } l \in \{2,\ldots,n_1\}, \\
  m = l \text{ if } l \in \{n_1 + 2,\ldots,N_1 - 1\}, 
\end{cases} \]
\[ d_4^{(2)} = W_2(b_1,0^4), \quad d_5^{(2)} = W_2(b_2,0^4), \quad d_6^{(2)} = W_2(b_3,0^4), \]
\[ d_7^{(2)} = W_2(0^m,aG,0^{7-m-1}) \text{ where } \begin{cases} 
  m = 3 \text{ if } l = 2, \\
  m = l \text{ if } l \in \{4,\ldots,6\}, 
\end{cases} \]
\[ d_4^{(3)} = (W_2^{-1})^T(b_1,0^4), \quad d_5^{(3)} = (W_2^{-1})^T(b_2,0^4), \quad d_6^{(3)} = (W_2^{-1})^T(b_3,0^4), \]
\[ d_7^{(3)} = (W_2^{-1})^T(0^m,bG,0^{7-m-1}) \text{ where } \begin{cases} 
  m = 3 \text{ if } l = 2, \\
  m = l \text{ if } l \in \{4,\ldots,6\}, 
\end{cases} \]
\[ g_0^{(1)} = W_1(g_3,0^{N_1-3}), \quad g_l^{(1)} = W_1(0^l+1,aG,0^{N_1-l-2}), \quad l = 2,\ldots,n_1; \]
\[ g_0^{(2)} = W_2(a_3,0^4), \quad g_l^{(2)} = W_2(0^3,acG,0^3), \]
where \((v,0^{N_1-3}) = (G',G'',G''',0^{N_1-3})\) for any \(v = (G',G'',G''') \in V = G^3\). This implies that \(D^{(0)} = (d_l^{(0)})_{l=1,\ldots,5}\) and \(D^{(0)} = (d_l^{(0)})_{l=1,\ldots,5}\) and \(D^{(j)} = (d_l^{(j)})_{l=1,\ldots,3n_j+1}\) and \(D^{(j)} = (d_l^{(j)})_{l=1,\ldots,3n_j+1}\), \(j = 1,2\) are dual orthonormal bases. \(F\) can compute \(D^{(j)}, j = 0,1,2; \)
\( \hat{D}^{(0)} = (d_1^{(0)},d_2^{(0)},d_3^{(0)},d_4^{(0)},d_5^{(0)}), \quad \hat{D}^{(j)} = (d_1^{(j)},\ldots,d_{n_j}^{(j)},d_{n_j+2}^{(j)},\ldots,d_{3n_j+1}^{(j)})\), \(j = 1,2\) using \(\hat{b} = (b_1,b_2), B^*, B^*, aG\), and \(aG\). \(F\) then gives \(\text{param}_{\hat{b}}(\{\hat{D}^{(k)},\hat{D}^{(k)}\}_{k=0,1,2},g_0^{(0)},g_0^{(1)},g_0^{(2)}\} + \{g_i^{(1)}\}_{i=2,\ldots,n_1},g_2^{(2)}\) to \(C\), and outputs bit \(\beta'\) if \(C\) outputs \(\beta'\).

We observe that
\[ g_0^{(0)} = (\omega',0,0,0)^T, \quad g_0^{(0)} = (\omega',0,0,0)^T, \]
\[ g_1^{(0)} = (\omega',0,0,0)^T, \quad g_1^{(0)} = (\omega',0,0,0)^T, \]
\[ g_{0,1}^{(1)} = \left( \omega \beta^{1(1)} e_1, 0^{n_1}, 0^{n_1}, 1 \gamma \right)_{D(1)}, \quad g_{0,1}^{(2)} = \left( \omega, 0, 2, 0, 1 \gamma \right)_{D(0)}, \]

\[ g_{1,1}^{(1)} = \left( \omega \beta^{1(1)} e_1, 0^{n_1}, 0^{n_1}, 1 \gamma \right)_{D(1)}, \quad g_{1,1}^{(2)} = \left( \omega, 0, 2, 0, 1 \gamma \right)_{D(0)}, \]

\[ g_i^{(1)} = \omega b_i(1) \quad i = 2, \ldots, n_1, \]

\[ g_2^{(2)} = \omega b_2^{(2)} \]

where \( \omega' = \delta, \tau' = \rho, \gamma' = \sigma \) are distributed uniformly in \( \mathbb{F}_q \). Therefore, the distribution of \( \left( \text{param}_{\mathbb{F}_q}, \{ D(k), \hat{b}^{(1)} \}_{k=0,1,2}, g_\beta, \{ g_i \}_{i=2,\ldots,n_1}, g_2 \right) \) is exactly the same as in the instance of the Basic Problem 1.

**Lemma 5.6.** For any adversary \( B \), there exists a probabilistic machine \( C \), whose running time is essentially the same as that of \( B \), such that for any security parameter \( \lambda \), \( \text{Adv}^{\text{BP}1}_{\text{G}}(\lambda) \leq \text{Adv}^{\text{BP}1}_{\text{B}}(\lambda) + 3/q \) for \( (n_1, n_2, 2) \).

**Proof.** \( C \) is given an instance of the Basic Problem 1, i.e. \( \left( \text{param}_{\mathbb{F}_q}, \{ D(k), \hat{b}^{(1)} \}_{k=0,1,2}, f_\beta \right) \). It computes \( r_j \) \( \left. \frac{U}{\beta} \right| \) span \( b_i^{(1)} \) \( \beta^{1(1)} \) \( \left. \frac{U}{\beta} \right| \) span \( b_i^{(2)} \), \( j = 1, \ldots, d \) and sets \( t_{j,1}^{(1)} = \beta^{1(1)} + r_j, t_{j,1}^{(2)} = f_{j,1}^{(2)} + r_j, j = 1, \ldots, d \).

Then, \( C \) chooses \( u_0 \in \mathbb{F}_q \), \( (\nu_{i,j}^{(k)})_i \leftarrow \mathcal{G}(\mathbb{F}_q, n_k), (z_{i,j}^{(k)}) = \left( (\nu_{i,j}^{(k)})^{-1} \right)^T, i = 1, \ldots, n_k, j = 1, \ldots, k, l = 1, 2 \), and computes:

\[ d_2^{(0)} = (0, u_0, 0, 0, 0)_{\mathbb{F}_q}, \]

\[ d_{nk+i}^{(k)} = \left( 0^{nk}, u_i^{(k)}, \ldots, u_i^{(k)}_{i=nk}, 0^{nk}, 1 \right)_{\mathbb{F}_q}, i = 1, \ldots, n_k, k = 1, 2; \]

\[ C \] then sets dual orthonormal basis vectors

\[ d_2^{(0)} = (0, u_0^{-1}, 0, 0, 0)_{\mathbb{F}_q}, \]

\[ d_{nk+i}^{(k)} = \left( 0^{nk}, z_i^{(k)}, \ldots, z_i^{(k)}_{i=nk}, 0^{nk}, 1 \right)_{\mathbb{F}_q}, i = 1, \ldots, n_k, k = 1, 2. \]

Note that \( C \) cannot compute \( d_2^{(0)} \) and \( d_{nk+i}^{(k)}, i = 1, \ldots, n_k, k = 1, 2 \) due to the lack of \( b_2^{(0)} \) and \( b_{nk+i}^{(k)} \). Then, \( C \) computes \( D(0) = (b_1^{(0)}, d_2^{(0)}, b_2^{(0)}, b_3^{(0)}, b_4^{(0)}, \hat{b}_1^{(0)}, \hat{b}_2^{(0)}, \hat{b}_3^{(0)}, \hat{b}_4^{(0)}), \hat{D}^{(0)} = (b_1^{(0)}, b_2^{(0)}, \hat{b}_1^{(0)}, \hat{b}_2^{(0)}, b_3^{(0)}, b_4^{(0)}, b_3^{(0)}, b_4^{(0)}), \]

\( D(k) = (b_1^{(k)}, \ldots, b_{nk}^{(k)}, d_{nk+i}^{(k)}, \ldots, d_{2nk+i}^{(k)}, b_{2nk+i}^{(k)}, \ldots, b_{3nk+i}^{(k)}, k = 1, 2 \).

Finally, \( C \) hands \( \left( \text{param}_{\mathbb{F}_q}, \{ D(k), \hat{b}^{(1)} \}_{k=0,1,2}, f_\beta \right), \{ t^{(1)}_{j,1} \}_{j=1, \ldots, d}, \{ t^{(2)}_{j,1,j} \}_{j=1, \ldots, d}, \{ f_i^{(1)} \}_{i=2, \ldots, n_k}, f_2^{(2)} \) over to \( B \) and outputs \( \beta' \in \{ 0, 1 \} \) if \( B \) outputs \( \beta' \).

Observe that with respect to \( D(k), \hat{b}^{(1)}, k = 0, 1, 2 \), the input to \( B \) has the same distribution as the instance of Problem 1 unless the following events occur: \( u = 0, \hat{u}^{(0)} = 0, \) or \( \hat{u}^{(2)} = 0 \). Those events occur with probability \( 3/q \) when \( \beta = 1 \).

**Proof of Lemma 5.2** Combining Lemmas 5.3, 5.4, 5.7 and 5.8 we obtain Lemma 5.2

**Definition 5.6** (Basic Problem 2). Basic Problem 2 is to find bit \( \beta \) given \( \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), f_\beta, \{ \hat{b}^{(k)}, \hat{b}^{(1)} \}_{k=1,2}, \left( y_{\beta, i}^{(1)} \right)_{i=1, \ldots, n_k} \) \( \left( \frac{R}{\beta} \right) \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), \hat{b}^{(1)} \{ y_{\beta, i}^{(1)} \}_{i=1, \ldots, n_k} \) \( \left( \frac{R}{\beta} \right) \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), \hat{b}^{(1)} \{ y_{\beta, i}^{(1)} \}_{i=1, \ldots, n_k} \) \( \left( \frac{R}{\beta} \right) \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), \hat{b}^{(1)} \{ y_{\beta, i}^{(1)} \}_{i=1, \ldots, n_k} \) \( \left( \frac{R}{\beta} \right) \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), \hat{b}^{(1)} \{ y_{\beta, i}^{(1)} \}_{i=1, \ldots, n_k} \) \( \left( \frac{R}{\beta} \right) \left( \text{param}_{\mathbb{F}_q}, \hat{b}^{(0)} \right), \hat{b}^{(1)} \{ y_{\beta, i}^{(1)} \}_{i=1, \ldots, n_k} \)
{0,1} with probability non-negligibly greater than by a random guess, where

\[
G_{\beta}^{\text{BP2}} \left( (1^\lambda, \tilde{n}) = (2; n_1, n_2 = 2) \right) : \\
(\text{param}_{\beta}, B^0, B^*(0), B^X(1), B^*(1), B^*(2)) \xleftarrow{\text{R}} G_{\text{ob}}(1^\lambda, \tilde{n}), \\
\hat{B}^0 = (b_1^0, b_2^0, b_3^0, b_4^0, b_5^0), \\
\hat{B}^X = (b_1^X, b_2^X, b_3^X, b_4^X, b_5^X), \\
\omega, \xi, \delta \in \mathbb{F}_q, z, \pi \in \mathbb{F}_q^\times, \\
y_0^{(0)} = (\omega, 0, 0, \xi, 0)_{\beta^0(0)}, y_1^{(0)} = (\omega, z, 0, \xi, 0)_{\beta^0(0)}, f^{(0)} = (\delta, \pi, 0, 0, 0)_{\beta^0(0)}, \\
\text{for } k = 1, 2 \text{ and } i = 1, \ldots, n_k : \\
y_{0,i}^{(k)} = (\omega e_i^{(k)}, 0_{n_k}, \xi e_i^{(k)}, 0_{n_k}, 1)_{\beta^0(0)}, \\
y_{1,i}^{(k)} = (\omega e_i^{(k)}, z e_i^{(k)}, 0_{n_k}, 1)_{\beta^0(0)}, \\
f_{i}^{(k)} = (\sigma e_i^{(k)}, \pi e_i^{(k)}, 0_{n_k}, 1)_{\beta^0(0)}, \\
\text{Output } (\text{param}_{\beta}, \hat{B}^0, B^*(0), y_{\beta}^{*}, f^{(0)}, \{\hat{B}^{(k)}, B^*(k), \{y_{\beta,i}^{(k)}, f_{i}^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2}).
\]

Let AdvC^{BP2}(\lambda) denote the advantage of a PPT algorithm C for the Basic Problem 2.

**Lemma 5.7.** For any adversary C, there exists a probabilistic machine F, whose running time is essentially the same as that of C, such that for any security parameter \(\lambda\), AdvC^{BP2}(\lambda) = Adv\beta^{BP0}(\lambda) for \(\tilde{n} = (2; n_1, n_2 = 2)\).

**Proof.** F is given a Basic Problem 0 instance (param_{BP0}, \hat{B}, B^*, y_{\beta}^*, f, bG, aG, acG).

With param_{G} = (q, \hat{G}, G_T, G, e) contained in param_{BP0}, C computes

\[
\begin{align*}
\text{param}_{V_0} &= (q, V_0, G_T, A_0, e) \xleftarrow{\text{R}} G_{\text{ob}}(1^\lambda, 5, \text{param}_{G}), \\
\text{param}_{V_1} &= (q, V_1, G_T, A_1, e) \xleftarrow{\text{R}} G_{\text{ob}}(1^\lambda, 3n_1 + 1, \text{param}_{G}), \quad l = 1, 2, \\
\text{param}_{\tilde{n}} &= (\{\text{param}_{V_i}\}_{i=0,1,2}, g_T),
\end{align*}
\]

where g_T is contained in param_{BP0}. F generates random linear transformation W_l on V_l (\(l = 0, 1, 2\)) given in Lemma 5.3 then sets

\[
\begin{align*}
\hat{d}_l^{(0)} &= W_0(b_l, 0, 0), \quad l = 1, 2, \\
\hat{d}_3^{(0)} &= W_0(0, 0, 0, 0, bG), \\
\hat{d}_4^{(0)} &= W_0(b_3, 0, 0), \\
\hat{d}_5^{(0)} &= W_0(0, 0, 0, 0, bG, 0), \\
\hat{d}_l^{(0)} &= (W_0^{-1})^T(b_l, 0, 0), \quad l = 1, 2, \\
\hat{d}_3^{(0)} &= (W_0^{-1})^T(0, 0, 0, 0, 0, aG), \\
\hat{d}_4^{(0)} &= (W_0^{-1})^T(b_3, 0, 0), \\
\hat{d}_5^{(0)} &= (W_0^{-1})^T(0, 0, 0, 0, aG, 0), \\
\hat{g}_{\beta}^{(0)} &= (W_0^{-1})^T(y_{\beta}^*, 0, 0), \\
g^{(0)} &= W_0(f, 0, 0),
\end{align*}
\]

For \(k = 1, 2\) :
For \( l = 1, 2, 3 \) and \( i = 1, \ldots, n_k \):
\[
d^{(k)}_{l(1)n_k + i} = W_k(y^{3(i-1)}, b_i, y^{3(n_k-i)}, 0);
\]
\[
d^{(k)}_{3n_k + 1} = W_k(0^{3n_k}, bG),
\]
For \( l = 1, 2, 3 \) and \( i = 1, \ldots, n_k \):
\[
d^{(k)}_{(l-1)n_k + i} = (W^{-1}_k)T(y^{3(i-1)}, b_i^*, y^{3(n_k-i)}, 0);
\]
\[
d^{(k)}_{3n_k + 1} = (W^{-1}_k)T(0^{3n_k}, aG),
\]
For \( i = 1, \ldots, n_k \):
\[
p_{β,i}^{(k)} = (W^{-1}_k)T(y^{3(i-1)}, y_b^*, y^{3(n_k-i)}, 0),
\]
\[
g_i^{(k)} = W_1(0^{3(i-1)}, f, 0^{3(n_k-i)}, 0).
\]
Hence, we have that \( D^{(0)} = (d_i^{(0)})_{l=1, \ldots, 5} \) and \( D^{*(0)} = (d_i^{*(0)})_{l=1, \ldots, 5} \) as well as \( D^{(j)} = (d_i^{(j)})_{l=1, \ldots, 3n_k+1} \) and \( D^{*(j)} = (d_i^{*(j)})_{l=1, \ldots, 3n_k+1} \). \( j = 1, 2 \) are dual orthonormal bases. \( \mathcal{F} \) can compute \( D^{*(j)}, j = 0, 1, 2; \mathcal{B}^{(0)} = (d_i^{(0)}, d_i^{(d)}, d_i^{(0)}, d_i^{(0)}), \mathcal{B}^{(j)} = (d_i^{(j)}, \ldots, d_i^{(j)}, d_i^{(j)}, d_i^{(j)}, d_i^{(j)}, d_i^{(j)}), j = 1, 2, \) using \( \hat{\mathcal{B}} = (b_1, b_3, \mathfrak{B}, bG, \mathfrak{a}G) \).

Then, \( \mathcal{F} \) hands \((\text{param}^{(k)}, \mathcal{B}^{(0)}, \mathcal{B}^{*(0)}, p_{β,i}^{(k)}, g_i^{(k)}, (\mathcal{B}^{(k)}, D^{*(k)}, (p_{β,i}^{(k)}, g_i^{(k)}))_{i=1, \ldots, n_k})_{k=1, 2}\) over to \( \mathcal{C} \) and outputs bit \( \beta' \) if \( \mathcal{C} \) outputs \( \beta' \).

We observe that
\[
p_0^{(0)} = (\omega, 0, 0, \xi, 0)_{D^{(0)}}, \quad p_i^{(0)} = (\omega, z, 0, \xi, 0)_{D^{*(0)}}, \quad g_i^{(0)} = (\delta, \pi, 0, 0, 0)_{D^{(0)}},
\]
For \( k = 1, 2 \) and \( i = 1, \ldots, n_k \):
\[
p_{β,i}^{(k)} = \begin{pmatrix} \omega_{\beta,i}^{(k)} & n_k & n_k & 0 & n_k & 1 \end{pmatrix}_{D^{*(k)}},
\]
\[
p_i^{(k)} = \begin{pmatrix} \omega_{i}^{(k)} & n_k & n_k & 0 & n_k & 1 \end{pmatrix}_{D^{*(k)}},
\]
\[
g_i^{(k)} = \begin{pmatrix} \delta_{i}^{(k)} & \pi \epsilon_{i}^{(k)} & 0 \end{pmatrix}_{D^{(k)}}.
\]

Therefore, the distribution of \((\text{param}^{(k)}, \mathcal{B}^{(0)}, \mathcal{B}^{*(0)}, p_{β,i}^{(k)}, g_i^{(k)}, (\mathcal{B}^{(k)}, D^{*(k)}, (p_{β,i}^{(k)}, g_i^{(k)}))_{i=1, \ldots, n_k})_{k=1, 2}\) is exactly the same as in the instance of the Basic Problem 2.

**Lemma 5.8.** For any adversary \( \mathcal{B} \), there exists a probabilistic machine \( \mathcal{C} \), whose running time is essentially the same as that of \( \mathcal{B} \), such that for any security parameter \( \lambda \), \( \text{Adv}_{\mathcal{B}^{2\mathcal{P}}}^{\mathcal{P}}(\lambda) = \text{Adv}_{\mathcal{B}^{2\mathcal{P}}}^{\mathcal{B}}(\lambda) \).

**Proof.** Given an instance of the Basic Problem 2, i.e. \((\text{param}^{(k)}, \mathcal{B}^{(0)}, \mathcal{B}^{*(0)}, y_b^{(0)}, f^{(0)}, (\mathcal{B}^{(k)}, \mathcal{B}^{*(k)}, \{y_{β,i}^{(k)}, f_i^{(k)}\}_{i=1, \ldots, n_k})_{k=1, 2}\) the algorithm \( \mathcal{C} \) computes \( r_i^{(k)} \leftarrow \text{span} < b_{2n_k+1}^{(k)}, \ldots, b_{3n_k}^{(k)} > \) and sets \( h_{β,i}^{(k)} = y_{β,i}^{(k)} + r_i^{(k)}, i = 1, \ldots, n_k, k = 1, 2, \).

Then, \( \mathcal{C} \) chooses \( z_0^{(k)} \leftarrow \mathcal{F}^{*\mathcal{F}}_{\mathcal{Q}}, (z_{i,j}^{(k)} \leftarrow \text{GL}(\mathbb{F}_q, n_k), i = 1, \ldots, n_k, j = 1, \ldots, n_k, k = 1, 2, \) and computes:
\[
d^{(0)}_2 = (0, z_0', 0, 0, 0)_{\mathcal{B}^{*(0)}},
\]
Then, $C$ sets $z_0 = z^{-1}z_0$, $u_0 = z^{-1}(z_{i,j}^{(k)}) = z^{-1}(z_{i,j}^{(k)})$, and $(u_{i,j}^{(k)}) = ((z_{i,j}^{(k)})^{-1})^T$, where $z$ is defined as in the Basic Problem 2. Then,

$$d_{n_k+i}^{(k)} = (0, z_{i,j}^{(k)}, z_{i,j}^{(k)}, 0^{n_k}, 1_{B^{(k)}}, i = 1, \ldots, n_k, \ k = 1, 2).$$

Finally, $C$ hands $(\text{param}, \widehat{D}^{(0)}, D^{(0)}, y^{(0)}, f^{(0)}, \{\widehat{D}^{(k)}, D^{(k)}, y^{(k)}, f^{(k)}\}_{i=1, \ldots, n_k}^{k=1, 2})$ over to $B$ and if $B$ outputs $\beta'$ forwards this bit as its own output. For $\pi$ in the Basic Problem 2 let $\pi' = z\pi$. Then, with respect to $\pi'$, $D^{(k)}, D^{(k)}, k = 0, 1, 2$, the above input to $B$ has the same distribution as the instance of Problem 2.

Next, we will prove our scheme FH-RPE using a sequence of games under Problem 1 in Definition 5.11 and Problem 2 in Definition 5.12

**Lemma 5.9.** For $p \in \mathbb{F}_q$, let $C_p = \{(\overrightarrow{\varphi}, \overrightarrow{\psi})|\overrightarrow{\varphi} \cdot \overrightarrow{\psi} = p\} \subset V \times V^*$ where $V$ is an $n$-dimensional vector space $\mathbb{F}_q^n$, and $V^*$ its dual. For all $(\overrightarrow{\varphi}, \overrightarrow{\psi}) \in C_p$, for all $(\overrightarrow{\varphi'}, \overrightarrow{\psi'}) \in C_p$, $Pr[\overrightarrow{\varphi} U = \overrightarrow{\varphi'} \land \overrightarrow{\psi} Z = \overrightarrow{\psi'}] = Pr[\overrightarrow{\varphi} Z = \overrightarrow{\varphi'} \land \overrightarrow{\psi} U = \overrightarrow{\psi'}] = 1/2C_p$, where $Z \overset{\text{iid}}{\sim} GL(n, \mathbb{F}_q), U = (Z^{-1})^T$, and $\sharp C_p$ denotes the number of elements in $C_p$.

The proof of Lemma 5.9 was given in [43]. Lemma 5.9 will be used in the proof of Lemma 5.11 and Lemma 5.12 shown later.

We then consider the following games:

**Game 0.** This is the real security game from Definition 5.1

**Game 1.** Game 1 is almost identical to Game 0, except that the ciphertext for challenge attribute vectors ($\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)}$), challenge revocation lists ($L^{(0)}, L^{(1)}$), and challenge plain-
Chapter 5. Revocable Predicate Encryption with Full Hiding

texts \((M'(0), M'(1))\) is

\[ c_0 = (\delta, w, \zeta, 0, \varphi)^{B(0)}, \]
\[ c_1 = (\delta, \varphi, \psi, 0, \varphi)^{B(1)}, \]
\[ \forall x \in \text{RevokeNodes}(Tree, M') : c_x = (\delta, (1, -ID_x), (1, -ID_x) \cdot U^{(2)}, 0^{(2)}, \varphi_x^{B(2)}), \]
\[ c_M = g_T^{c}M^{(b)}, \]

where \(\delta, w, \zeta, \psi, \varphi, \psi_x \in \mathbb{F}_q\), \(\varphi \in \{0, 1\}\), \(\varphi^{(b)} = (y_1^{(b)}, \ldots, y_n^{(b)})\), and \((w_1^{(1)}, \ldots, w_n^{(1)}) \leftarrow \mathbb{F}_q \setminus \{0\}\).

**Game 2-0** \((m = 0, \ldots, \nu - 1)\). Game 2-0 is Game 1. Game 2-0 is almost identical to Game 2-m, except the reply to the \((m + 1)\)-th GenKey query for \(x = (x_1, \ldots, x_n)\), and the challenge ciphertexts are computed as follows:

\[ k_0 = (-\alpha, -\gamma, \eta, \delta, 0)^{B(0)}, \]
\[ k_1 = (\alpha^{(1)} \psi^{(1)}, \beta^{(1)} \varphi, \gamma^{(1)} \psi, \sigma^{(1)} \varphi, (1)^{B(1)}), \]
\[ \forall x \in P(I) : k_x = (\alpha^{(2)} + \beta^{(2)} x, \beta^{(2)} \varphi_x, \gamma^{(2)} \psi, \sigma^{(2)} \varphi, (2)^{B(2)}), \]
\[ c_0 = (\delta, w, \zeta, 0, \varphi)^{B(0)}, \]
\[ c_1 = (\delta, \varphi, \psi, 0, \varphi)^{B(1)}, \]
\[ \forall x \in \text{RevokeNodes}(Tree, M') : c_x = (\delta, (1, -ID_x), (1, -ID_x) \cdot U^{(2)}, 0^{(2)}, \varphi_x^{B(2)}), \]
\[ c_M = g_T^{c}M^{(b)}, \]

where \(\epsilon, \gamma^{(1)}, \psi^{(2)} \in \mathbb{F}_q\), \(\psi^{(2)} \leftarrow \Gamma\), \(Z^{(k)} \leftarrow GL(\mathbb{F}_q, n_k)\), \(U^{(k)} = (Z^{(k)})^{-1} \cdot T, k = 1, 2\), and all the other variables are generated as in Game 2-m.

**Game 2-(m + 1)** \((m = 0, \ldots, \nu - 1)\). Game 2-(m + 1) is almost identical to Game 2-m',
except the reply to the \((m + 1)\)-th \text{GenKey} query for \(\vec{x} = (x_1, \ldots, x_n)\) is

\[
 k_0 = (-\alpha, \epsilon, 1, \eta, 0)_{\mathbb{B}(0)},
\]

\[
 k_1 = (\alpha^{(1)} \rightarrow 1 + \beta^{(1)} \rightarrow x_1^n, \eta^{(1)}_{1, n_1}, \ldots, \eta^{(1)}_{n_1}), \quad 0)_{\mathbb{B}(1)},
\]

\[
 k_2 = (\alpha^{(2)} + \beta^{(2)} x_{1,x_1}, \eta^{(2)}_{1,x_1}, \ldots, \eta^{(2)}_{x_1,n_1}, \eta, 0)_{\mathbb{B}(2)},
\]

\[
 \forall x \in \mathbb{P}(I) : \quad k_x = (\alpha^{(2)} + \beta^{(2)} x_{1,x_1}, \eta^{(2)}_{1,x_1}, \ldots, \eta^{(2)}_{x_1,n_1}, \eta, 0)_{\mathbb{B}(2)},
\]

the challenge ciphertext is the same as Eqs. (5.1)-(5.4), where \(v^{(2)}_{1,x_1}, v^{(2)}_{2,x_2} \leftarrow \mathbb{F}_q, (v^{(1)}_1, \ldots, v^{(1)}_{b_0}) \leftarrow \mathbb{F}_{q^n} \setminus \{0\}\), and all the other variables are generated as in Game 2-\(m\).

**Game 3.** Game 3 is almost identical to Game 2-\(\nu\), except that the target ciphertext \(C\) for challenge attribute vectors \((\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)})\), challenge revocation lists \((L^{(0)}, L^{(1)})\), and challenge plaintexts \((M^{(0)}, M^{(1)})\) are computed as follows:

\[
 c_0 = (\delta, w, \zeta', 0, \varphi)_{\mathbb{B}(0)},
\]

\[
 c_1 = (\overrightarrow{y}^{(0)}, w^{(1)}, \ldots, w^{(1)}_m, 0^{m_1}, \ldots, 0^{m_1}, \varphi)_{\mathbb{B}(1)},
\]

\[
 \forall x \in \text{RevokeNodes}(Tree, L^{(0)}) : \quad c_x = (ID'_x, ID'_x, w^{(2)}_{1,x}, w^{(2)}_{1,x}, 0^{2}, \varphi)_{\mathbb{B}(2)},
\]

\[
 c_M = g_\zeta M^{(b)}.
\]

where \(\zeta' \leftarrow \mathbb{F}_q, \overrightarrow{y}^{(0)} = (y^{(0)}_1, \ldots, y^{(0)}_m) \leftarrow \mathbb{F}_{q^n}^m\), and \(ID'_x, ID'_x \leftarrow \mathbb{F}_q^2\). We note that \(\zeta', (y^{(0)}_1, \ldots, y^{(0)}_m)\) and \(ID'_x, ID'_x\) are chosen uniformly and independently from \(\zeta, (\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)})\) and \(ID_x\), respectively.

Let \(\text{Adv}_{\lambda}^{(0)}(\lambda)\) be \(\text{Adv}_{\lambda}^{\text{FH-RPE}}(\lambda)\) in Game 0, and \(\text{Adv}_{\lambda}^{(j)}(\lambda), \text{Adv}_{\lambda}^{(2m)}(\lambda), \text{Adv}_{\lambda}^{(2m-\nu)}(\lambda), \text{Adv}_{\lambda}^{(3)}(\lambda)\) be the advantage of \(\mathcal{A}\) in Game 1, 2-\(m\), 2-\(m\) and 3 respectively. We can show Lemmas 5.10-5.14 which evaluate the gaps between pairs of \(\text{Adv}_{\lambda}^{(b)}(\lambda), \text{Adv}_{\lambda}^{(1)}(\lambda), \text{Adv}_{\lambda}^{(2m)}(\lambda), \text{Adv}_{\lambda}^{(2m-\nu)}(\lambda), \text{Adv}_{\lambda}^{(3)}(\lambda)\) for \(m = 0, \ldots, \nu - 1\), and \(\text{Adv}_{\lambda}^{(3)}(\lambda)\). From these lemmas and Lemma 5.1 and 5.2 we obtain:

\[
 \text{Adv}_{\lambda}^{\text{FH-RPE}}(\lambda) = \text{Adv}_{\lambda}^{(0)}(\lambda)
\]

\[
 \leq |\text{Adv}_{\lambda}^{(0)}(\lambda) - \text{Adv}_{\lambda}^{(1)}(\lambda)| + \sum_{m=0}^{\nu-1} |\text{Adv}_{\lambda}^{(2m)}(\lambda) - \text{Adv}_{\lambda}^{(2m-\nu)}(\lambda)| +
\]

\[
 \sum_{m=0}^{\nu-1} |\text{Adv}_{\lambda}^{(2m-\nu)}(\lambda) - \text{Adv}_{\lambda}^{(2-(m+1))}(\lambda)| + |\text{Adv}_{\lambda}^{(2-\nu)}(\lambda) - \text{Adv}_{\lambda}^{(3)}(\lambda)| +
\]

\[
 \text{Adv}_{\lambda}^{(3)}(\lambda)
\]

\[
 \leq \text{Adv}_{\lambda}^{(b_1)}(\lambda) + \sum_{m=0}^{\nu-1} \text{Adv}_{\lambda}^{b_2}(\lambda) + \sum_{m=0}^{\nu-1} \text{Adv}_{\lambda}^{b_2}(\lambda) + (2 \log N + 8\nu + \log N + 2)/q
\]
Chapter 5. Revocable Predicate Encryption with Full Hiding

$$\leq (2\nu + 1)\text{Adv}^{\text{Plin}}_2(\lambda) + (2 \log N\nu + 18\nu + \log N + 10)/q.$$ 

This completes the proof of Theorem 5.1.

**Lemma 5.10.** For any adversary $A$, there exists a probabilistic machine $B_1$, whose running time is essentially the same as that of $A$, such that for any security parameter $\lambda$, $|\text{Adv}^{(0)}_A(\lambda) − \text{Adv}^{(1)}_A(\lambda)| = \text{Adv}^{(1)}_B(\lambda)$.

**Proof.** Suppose a polynomial time adversary $A$ can successfully distinguish between Game 0 and Game 1. We construct a simulator $B_1$ that leverages $A$ as a black box to solve Problem 1. The procedure is shown as follows:

1. $B_1$ is given an instance of Problem 1, i.e. $(\text{param}_p, \{\mathbb{B}^{(k)}, \mathbb{B}^{*+(k)}\}_{k = 0, 1, 2}, \{t^{(0)}_\beta, \{t^{(1)}_{\beta,j}, t^{(2)}_{\beta,j}\}_{j = 1, \ldots, d}\}$, and play the role of the challenger in the security game against adversary $A$. Let $d$ denote the total number of the nodes in the binary tree such that each node is associated with one element in $\{t^{(2)}_{\beta,j}\}_{j = 1, \ldots, d}$.

2. At the beginning of the game, $B_1$ gives $A$ the public key $PK = (1^\lambda, \text{param}_p, (b^{(0)}_1, b^{(0)}_3, b^{(0)}_5, b^{(1)}_1, \ldots, b^{(1)}_{n_1}, b^{(1)}_{n_1 + 1}, b^{(2)}_1, b^{(2)}_2, b^{(2)}_7)$, which is obtained from the Problem 1 instance.

3. When a $\text{GenKey}$ query is issued, $B_1$ computes a normal secret key using $(\mathbb{B}^{+(0)}, \mathbb{B}^{+(1)}, \mathbb{B}^{*+(2)})$, which is obtained from the Problem 1 instance.

4. When $B_1$ receives challenge attribute vectors $(\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)})$, challenge revocation lists $(L^{(0)}, L^{(1)})$, and challenge plaintexts $(M^{(0)}, M^{(1)})$ from $A$, $B_1$ computes and returns

$$C = (c_0, c_1, \{c_x\}_{x \in \text{RevokeNodes}(\text{Tree}, L^{(b)})}, c_M),$$

where $c_0 = t^{(0)}_0 + \zeta b^{(0)}_0$, $c_1 = y^{(b)}_1 t^{(1)}_{\beta,j} + \sum_{i = 2}^{n_1} y^{(b)}_i t^{(1)}_i$, $\forall x \in \text{RevokeNodes}(\text{Tree}, L^{(b)}) : c_x = t^{(2)}_{\beta,j,x} + (-ID_x) \cdot t^{(2)}_2$, and $c_M = g^{\sum_{i = 1}^{n_1} y^{(b)}_i}$ using $(t^{(0)}_b, t^{(1)}_b, \{t^{(2)}_{\beta,j}\}_{j = 1, \ldots, d}, \{t^{(1)}_i\}_{i = 2, \ldots, n_1}, t^{(2)}_2, b^{(0)}_0)$ from the instance of Problem 1 and $\overrightarrow{y}^{(b)}, L^{(b)}, M^{(b)}$ where $\zeta \overset{\$}{\leftarrow} \mathbb{F}_q$, $b \overset{\$}{\leftarrow} \{0, 1\}$.

5. After the challenge phase, $\text{GenKey}$ oracle simulation for a key query is executed in the same manner as step 3.

6. $A$ outputs a bit $b'$. If $b = b'$, $B_1$ outputs 1. Otherwise, $B_1$ outputs 0.

**Claim 5.1.** For $\beta = 0$ the challenge ciphertext $C = (c_0, c_1, \{c_x\}_{x \in \text{RevokeNodes}(\text{Tree}, L^{(b)})}, c_M)$ generated in step 4 is distributed exactly as in Game 0, whereas if $\beta = 1$, the challenge ciphertext $C = (c_0, c_1, \{c_x\}_{x \in \text{RevokeNodes}(\text{Tree}, L^{(b)})}, c_M)$ generated in step 4 is identically distributed to Game 1.

**Proof.** First recall that $y^{(b)}_1 = 1$. If $\beta = 0$ then the ciphertext given by

$$c_0 = (\delta, 0, \zeta, 0, p)_{\mathbb{B}^{(0)}},$$

$$c_1 = (\overrightarrow{y}^{(b)}, 0, 0^r, 0, 0^r, 0^r, 0^r, 0^r, 1)_{\mathbb{B}^{(1)}},$$

where $\delta \overset{\$}{\leftarrow} \mathbb{F}_q$, $\zeta \overset{\$}{\leftarrow} \mathbb{F}_q$, and $\nu, \rho \overset{\$}{\leftarrow} \{0, 1\}$.
\[ \forall x \in \text{RevokeNodes}(\text{Tree}, L(b)) : c_x = (\delta, \delta(-ID_x), 0^2, 0^2, 0^1, \rho_x^{(2)})_{\beta(x)}, \]

is the challenge ciphertext from Game 0. In contrast, if \( \beta = 1 \) then the following components of the ciphertext have a different form

\[ c_0 = (\delta, u, \zeta, 0, \rho)_{\beta(0)}, \quad c_1 = (\delta, \hat{y}(b), u^{(1)}, u^{(2)}, 0^{n_1}, 0^{n_2}, \rho^{(1)})_{\beta(1)}, \]

\[ \forall x \in \text{RevokeNodes}(\text{Tree}, L(b)) : c_x = (\delta, \delta(-ID_x), 0^2, 0^2, 0^1, \rho_x^{(2)})_{\beta(2)}, \]

where \( \hat{y}(1) = (u_1^{(1)}, \ldots, u_{n_1}^{(1)}), \hat{y}(2) = (u_1^{(2)}, u_{2,2}^{(2)}). \) Since \( \hat{y}(1) \in \mathbb{F}_q^{n_1}, \hat{y}(2) \in \mathbb{F}_q^{n_2}, \rho^{(1)}, \rho^{(2)} \in \mathbb{F}_q \) are independently uniform this ciphertext corresponds to the challenge ciphertext from Game 1.

From the above claim, if \( \beta = 0 \) then simulated ciphertexts are distributed exactly as in Game 0, whereas for \( \beta = 1 \) their distribution is identical to Game 1. Therefore,

\[ |\text{Adv}_{\beta}^{(0)}(\lambda) - \text{Adv}_{\beta}^{(1)}(\lambda) | = \left| \Pr \left[ B_1(1^\lambda, x) \rightarrow 1 \bigg| x \leftarrow \mathbb{F}_q \right] - \Pr \left[ B_1(1^\lambda, x) \rightarrow 1 \bigg| x \leftarrow \mathbb{F}_q \right] \right| = \text{Adv}_{\beta}^{(1)}(\lambda). \]

This completes the proof of Lemma 5.10.

**Lemma 5.11.** For any adversary \( A \), there exists a probabilistic machine \( \mathcal{B}_{2m}^\prime \), whose running time is essentially the same as that of \( A \), such that for any security parameter \( \lambda \), \( |\text{Adv}_{\beta}^{(2-m)}(\lambda) - \text{Adv}_{\beta}^{(2+m)}(\lambda) | \leq \text{Adv}_{\beta}^{(2)}(\lambda) + (4 + \log N)/q. \)

**Proof.** Suppose a polynomial time adversary \( A \) can successfully distinguish between Game 2-\( m \) and Game 2-\( m' \). We construct a simulator \( \mathcal{B}_{2m}^\prime \) that leverages \( A \) as a black box to solve Problem 2. The procedure is shown as follows:

1. \( \mathcal{B}_{2m}^\prime \) is given an instance of Problem 2, i.e., \( (\text{param}_{\beta}, \mathcal{B}_{2m}(0), b^*(0), h^*(0), t^{(0)}, \hat{y}^{(0)}, \mathcal{B}^{(k)}, h^{(k)}, t^{(k)}) \), and acts as a challenger in the security game against adversary \( A \).

2. At the beginning of the game, \( \mathcal{B}_{2m}^\prime \) gives \( A \) the public key \( PK = (1^\lambda, \text{param}_{\beta}, (b_0^{0(0)}, b_3^{0(0)}, b_5^{0(0)}, b_4^{0(1)}, \ldots, b_5^{1(1)}, b_6^{0(1)}, b_5^{1(2)}, b_6^{2(2)}), (b_2^{(2)}, b_7^{(2)})) \), which is obtained from the Problem 2 instance.

3. When the \( s \)-th \( \text{GenKey} \) query is issued for a predicate \( \mathcal{P} = (x_1, \ldots, x_{n_1}) \), \( \mathcal{B}_{2m}^\prime \) answers as follows:

   a) For \( 1 \leq s \leq m \) the algorithm \( \mathcal{B}_{2m}^\prime \) computes a semi-functional key using \( \{b^{(k)}_{\beta,i,j}\}_{k=0,1,2} \) of the Problem 2 instance.

   b) For \( s = m + 1 \) it computes \( k_{\beta,i,j} = (I, k_0, k_1, \{k_x\}_{x \in \mathcal{P}(I)}) \) using \( \{h^{*(0), b_{\beta,i,j}^{0(0)}, b_{\beta,j}^{0(0)}, \{b_{\beta,i,j}^{*}(i), b_{\beta,i,j}^{*}(j)\}_{i=1,2,j=1,\ldots,n_1}\} \) of the Problem 2 instance as follows:

\[
\text{For } i = 1, 2 : \quad g_{i}, v_{i}, v'_i, \theta, u \leftarrow \mathbb{F}_q;
\]
Proof. 

4. When $\mathcal{B}'_{2m}$ receives challenge attribute vectors $(\overrightarrow{y}^{(0)}, \overrightarrow{y}^{(1)})$, challenge revocation lists $(L^{(0)}, L^{(1)})$, and challenge plaintexts $(M^{(0)}, M^{(1)})$ from $\mathcal{A}$, $\mathcal{B}'_{2m}$ computes and returns the ciphertext $C = (c_0, c_1, \{c_x \in \text{RevokeNodes}(\text{Trex}, L^{(0)}), c_M \})$ where $c_0 = t^{(0)} + \zeta b^{(0)}_i + \varphi b^{(0)}_i$, $c_1 = \sum_{j=1}^{n_i} \gamma_j^{(0)} (t_1^{(0)} + \varphi^{(1)} b^{(1)}_{3n+1+i})$, $\forall x \in \text{RevokeNodes}(\text{Trex}, L^{(b)}) : c_x = t_1^{(2)} + (-ID_x)k_2 + \varphi (b^{(2)}_1)$ and $c_M = \hat{g}_{2}^{(0)} M^{(0)}$, using $(t^{(0)}, \{t_1^{(0)} \}_{1=1, \ldots, n_i}, t_2^{(1)}, \gamma, \varphi)_{i=1, \ldots, n_i}$ from the instance of Problem 2 and $\overrightarrow{y}^{(b)}, L^{(b)}, M^{(b)}$, where $\zeta, \varphi, \varphi^{(1)}, \varphi^{(2)} \in \mathbb{F}_q, b \in \{0, 1\}$.

5. After the challenge phase, GenKey oracle simulation for a key query is executed in the same manner as step 3.

6. $\mathcal{A}$ outputs a bit $b'$. If $b = b'$, $\mathcal{B}'_{2m}$ outputs 1. Otherwise, $\mathcal{B}'_{2m}$ outputs 0.

Claim 5.2. The distribution of the view of adversary $\mathcal{A}$ in the above-mentioned game simulated by $\mathcal{B}'_{2m}$ given a Problem 2 instance with $\beta \in \{0, 1\}$ is the same as that in Game (2-m) (resp. Game (2-m')) if $\beta = 0$ (resp. $\beta = 1$) except with probability $(3 + \log N)/q$ (resp. 1/q).

Proof. It is clear that $\mathcal{B}'_{2m}$'s simulation of the public key generation (step 2) and the answers to the i-th GenKey query where $i \neq m + 1$ (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and the GenKey oracles in Game 2-m and Game 2-m'.

Next we analyze the distribution of the i-th GenKey query where $i = m + 1$ (case (b) of steps (3) and (5)). In this case values $s_{i-1}^{(0)}, s_{i}^{(0)}, s_{i-1}^{(1)}, s_{i}^{(1)}$, $i = 1, 2, j = 1, \ldots, n_i$ can be expressed as follows. Let $\beta^{(i)} = \theta_i \omega + \nu_i^{(i)}$, $\alpha^{(i)} = \varphi_i \omega + \nu_i$, $\alpha = \alpha^{(1)} + \alpha^{(2)}$, $\gamma = \varphi_1 + \varphi_2$, $\epsilon = \gamma z$. Then,

$$s_{0}^{(0)} = (\alpha, 0, 0, \gamma \xi, 0)_{\mathbb{G}_t}, \quad s_{1}^{(0)} = (\alpha, \epsilon, 0, \gamma \xi, 0)_{\mathbb{G}_t},$$

$$s_{i}^{(0)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right), \quad s_{i}^{(1)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right), \quad s_{i}^{(1)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right),$$

$$s_{i}^{(1)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right), \quad s_{i}^{(1)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right), \quad s_{i}^{(1)} = \left(\begin{array}{c}
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i} \\
\frac{n_i}{n_i}
\end{array}
\right),$$

where $\overrightarrow{z}^{(i)} = z^{(i)}_1, \ldots, z^{(i)}_n$, $\omega, \zeta, \xi, \{\omega_j^{(i)}, z^{(i)}_j\}_{i=1, j=1, \ldots, n_i}$ are defined as in Problem 2. If $\beta = 1$ in the instance of Problem 2 then the secret key $K_{\mathbb{F}_q}^{(i)} = (I, k_0, k_1, \{k_x\}_{x \in F_q})$ has the same distribution as in Eq. 5.5 except that $\epsilon w = \gamma$, where $\gamma = \varphi_1 + \varphi_2$ and $w = u \in \mathbb{F}_q$ of $c_0$. 
in Eq. [5.6]

Next, we show that the joint distribution of the response to $(m + 1)$-th GenKey query and of the challenge ciphertext in the simulation by $B'_{2m}$ for the given instance of Problem 2 is equivalent to the distribution in Game 2- $m$ if $\beta = 0$ and to the distribution in Game 2- $m'$ if $\beta = 1$.

If $\beta = 0$ then this equivalence follows easily, unless one of the following conditions holds:

1. $\omega$ defined in Problem 2 is zero, (2) $w = 0$, (3) $(w^{(1)}, \ldots, w^{(1)}_{n}) = \emptyset$, (4) $(w^{(2)}_{1}, w^{(2)}_{2}) = \emptyset$,

where $x \in \text{RevokeNodes}(\text{Tree}, L^{(b)})$, and $w, (w^{(1)}, \ldots, w^{(1)}_{n})$ and $(w^{(2)}, (2))$ are defined in Eqs. [5.1] [5.2] and [5.3] respectively. However, those events occur with probability $(3 + \log N)/q$.

If $\beta = 1$, then $B'_{2m}$’s simulation for the key is the same as that expressed in Eq. 5.5 and $B'_{2m}$’s simulation for the challenge ciphertext is the same as that expressed in Eq. 5.6 except that $cw = \gamma$, where $\gamma = \varphi_1 + \varphi_2$, and $w \leftarrow \mathbb{F}_q$ of $c_0$ in Eq. 5.6.

Therefore, we will show that $\gamma$ is uniformly distributed and is independent from the other variables used in the simulation by $B'_{2m}$. Since $\gamma$ is related to $\overrightarrow{A}_1, \overrightarrow{A}_{2,x}, \overrightarrow{B}_1^{(b)}$, and $\overrightarrow{B}_{2,x}$, where $\overrightarrow{A}_1 = (\varphi_1 \overrightarrow{A}_1^{(1)} + \varphi_2 \overrightarrow{A}_1^{(2)}) \cdot Z^{(1)}, \overrightarrow{A}_{2,x} = (\varphi_1 \overrightarrow{A}_1^{(2)} + \varphi_2 (1D_{x}, 1)) \cdot Z^{(2)} x \in \mathbb{F}(1), \text{and } \overrightarrow{B}_1^{(b)} = \overrightarrow{B}_{1,x} \cdot Z^{(1)}, \overrightarrow{B}_{2,x} = (1, -ID_{x}^{(1)}) \cdot U(x) \in \text{RevokeNodes}(\text{Tree}, L^{(b)})$, where $b \in \{0, 1\}$. We analyze joint distribution of these variables for the four distinct cases that appear in Definition 5.1.

1. When $f_{\overrightarrow{A}}(\overrightarrow{y}^{(0)}) = f_{\overrightarrow{A}}(\overrightarrow{y}^{(1)}) = 0$, due to Lemma 5.9 The pair $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)}) (b \in \{0, 1\})$ is uniformly and independently distributed over $C_{\varphi_1}(\overrightarrow{A}_1^{(1)} + \varphi_2), (b \in \{0, 1\})$ (over $Z^{(1)} \mathbb{F}_q$ $GL(\mathbb{F}_q, n_1)$). Since $\varphi_1 \overrightarrow{y} \leftarrow \mathbb{F}_q$, the pair $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)}) (b \in \{0, 1\})$ is thus uniformly and independently distributed over $\mathbb{F}_q^{n_1}$.

2. When $f_{\overrightarrow{A}}(\overrightarrow{y}^{(0)}) = f_{\overrightarrow{A}}(\overrightarrow{y}^{(1)}) = 1$ and $(I \in L^{(0)} \lor I \in L^{(1)})$, the pair $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)}) (b \in \{0, 1\})$ is uniformly and independently distributed over $C_{\varphi_1}$ over $Z^{(1)} \mathbb{F}_q GL(\mathbb{F}_q, n_1)$. The pair $(\overrightarrow{A}_{2,x}, \overrightarrow{B}_{2,x}) (b \in \{0, 1\})$ is uniformly and independently distributed over $\mathbb{F}_q^{4}$.

3. When $(f_{\overrightarrow{A}}(\overrightarrow{y}^{(0)}) = 1 \land f_{\overrightarrow{A}}(\overrightarrow{y}^{(1)}) = 0$ and $I \in L^{(0)}$, the pair $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)})$ (resp. $(A_1, B_1^{(b)})$) is uniformly and independently distributed over $C_{\varphi_1}$ (resp. $F_q^{n_1}$). The pair $(\overrightarrow{A}_{2,x}, \overrightarrow{B}_{2,x})$ is uniformly and independently distributed over $\mathbb{F}_q^{4}$.

4. When $(f_{\overrightarrow{A}}(\overrightarrow{y}^{(0)}) = 0 \land f_{\overrightarrow{A}}(\overrightarrow{y}^{(1)}) = 1$ and $I \in L^{(1)}$, the pair $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)})$ (resp. $(A_1, B_1^{(b)})$) is uniformly and independently distributed over $\mathbb{F}_q^{4}$ (resp. $C_{\varphi_1}$). The pair $(\overrightarrow{A}_{2,x}, \overrightarrow{B}_{2,x})$ is uniformly and independently distributed over $\mathbb{F}_q^{4}$.

Considering the adversary $A$’s restriction on key queries from Definition 5.1 in each of the above four cases at least one of $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)})$ and $(\overrightarrow{A}_{2,x}, \overrightarrow{B}_{2,x})$ is uniformly and independently distributed over $\mathbb{F}_q^{2n_t}$ for $k = 1, 2$. Therefore, $\gamma = \varphi_1 + \varphi_2$ is independent from the distribution of $\varphi_1$ (resp. $\varphi_2$), which can be given by $(\overrightarrow{A}_1, \overrightarrow{B}_1^{(b)})$ (resp. $(\overrightarrow{A}_{2,x}, \overrightarrow{B}_{2,x})$). Thus, $\gamma$ is uniformly and independently distributed from the other variables in the simulation of $B'_{2m}$.

Therefore, the view of $A$ in the game simulated by $B'_{2m}$ on input an instance of Problem 2 with $\beta = 1$ is the same as in Game 2- $m'$ unless $\omega = 0$ occurs. This event happens with probability $1/q$.

This completes the proof of Lemma 5.11
Lemma 5.12. For any adversary $A$, there exists a probabilistic machine $B_{2(m+1)}$, whose running time is essentially the same as that of $A$, such that for any security parameter $\lambda$, $|\text{Adv}_A^{(2m+1)}(\lambda) - \text{Adv}_A^{(2m+1)}(\lambda)| \leq \text{Adv}_B^{(2m+1)}(\lambda) + (4 + \log N)/q$.

Proof. Suppose a polynomial time adversary $A$ can successfully distinguish between Game 2-$m'$ and Game 2-$(m+1)$. We construct a simulator $B_{2(m+1)}$ that leverages $A$ as a black box to solve Problem 2. The procedure is same as that in the proof of Lemma 5.11 except that in case (b) of step 3 $k^{(j)}_{x,y}$ is computed as follows:

$$
k_0 = -s_{\beta}^{(0)} + \epsilon' b_2^{(0)} + b_3^{(0)},
$$

$$
k_1 = \sum_{j=1}^{n_1} x_j s_{B,j}^{(1)} + s_{\beta,1}^{(1)} + \sum_{j=1}^{n_1} v_j^{(1)} b_{n_1+j}^{(1)},
$$

$$\forall x \in \mathbb{P}(I) : k_x = ID_x s_{\beta,1}^{(2)} + s_{\beta,2}^{(2)} + s_{/2,1}^{(2)} + \sum_{j=1}^{2} v_j^{(2)} b_{/2}^{(2)},$$

where $\epsilon' \leftarrow \mathbb{F}_q$, $v_{/1}^{(2)}$, $v_{/2}^{(2)} \leftarrow \mathbb{F}_q$, $(v_1^{(1)}, \ldots, v_{n_1}^{(1)}) \leftarrow \mathbb{F}_q^n \setminus \{0\}$. In the last step, if $b = b'$, $B_{2(m+1)}$ outputs 0. Otherwise, $B_{2(m+1)}$ outputs 1.

The view of the adversary $A$ in the game simulated by $B_{2(m+1)}$ given an instance of Problem 2 with $\beta = 0$ is the same as in Game 2-$(m+1)$ unless one of following events occur: (1) $\omega = 0$ in Problem 2 instance, (2) $w = 0$, (3) $(w_{1,1}^{(1)}, \ldots, w_{n_1}^{(1)}) = 0$, (4) $(w_{1,2}^{(2)}, w_{2,2}^{(2)}) = 0$, where $x \in \text{RevokeNodes}(Tree, L^{(b)})$, and $w$, $(w_{1,1}^{(1)}, \ldots, w_{n_1}^{(1)})$ and $(w_{1,2}^{(2)}, w_{2,2}^{(2)})$ are defined in Eqs. 5.1, 5.2, and 5.3 respectively. Those events occur with probability $(3 + \log N)/q$. In case $\beta = 1$ the argument is similar to that in the proof of Lemma 5.11, i.e., each variable has uniform distribution and is independent from other variables occurring in the simulation by $B_{2(m+1)}$. The view of the adversary $A$ is the same as its view in Game 2-$m'$ unless $\omega = 0$ occurs in the instance of Problem 2. The event $\omega = 0$ occurs with probability $1/q$.

Lemma 5.13. For any adversary $A$, $\text{Adv}_A^{(3)}(\lambda) \leq \text{Adv}_A^{(2+\nu)}(\lambda) + (2 + \log N)/q$.

Proof. First we show the distribution $(\text{param}_T, \{b^{(j)}_x\}_{k=0,1,2}, \{k^{(j)}_{x,y}\}_{j=1,\ldots,C})$ of Game 3 is same as that of Game 2-$\nu$, where $k^{(j)}_{x,y}$ is the answer to the $j$-th key query, and $C$ is the challenge ciphertext. We will define new bases $B^{(k)}_x$ of $\forall_k$ and $D^{(k)}_x$ of $\forall_k$, $k = 0, 1, 2$.

For $k = 0$, we set $d_0^{(0)} = b_0^{(0)} - \lambda b_3^{(0)}$, and $d_3^{(0)} = b_3^{(0)} + \lambda b_2^{(0)}$, where $\lambda \leftarrow \mathbb{F}_q$. The new bases are $D^{(0)} = (b_1^{(0)}, d_2^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)})$ and $D^{*(0)} = (b_1^{*(0)}, b_2^{*(0)}, d_2^{*(0)}, b_3^{*(0)}, b_4^{(0)})$. We can easily verify that $D^{(0)}$ and $D^{*(0)}$ are dual orthonormal, and are distributed the same as the original bases $B^{(0)}$ and $B^{*(0)}$ respectively.

For $i = 1, \ldots, n_k$, $j = 1, \ldots, n_k$, $k = 1, 2$, choose $Q^{(k)} = (\mu^{(k)}_{i,j}) \leftarrow \mathbb{F}_q^{n_k \times n_k}$, and compute $d^{(k)}_{n_k+i} = b^{(k)}_{n_k+i} + \sum_{j=1}^{n_k} \mu^{(k)}_{i,j} b^{(k)}_{n_k+j}$, and $d^{*(k)}_{i} = b^{*(k)}_{i} - \sum_{j=1}^{n_k} \mu^{(k)}_{i,j} b^{(k)}_{n_k+j}$, which are equivalent to the following matrix computations:

$$
\begin{pmatrix}
& \overrightarrow{B_1^{(k)}} \\
\overrightarrow{B_2^{(k)}} & Q^{(k)} & I_{n_k}
\end{pmatrix} =
\begin{pmatrix}
I_{n_k} & 0_{n_k} \\
0_{n_k} & I_{n_k}
\end{pmatrix}
\begin{pmatrix}
\overrightarrow{B_1^{(k)}} \\
\overrightarrow{B_2^{(k)}} & \overrightarrow{B_1^{*(k)}} & \overrightarrow{B_2^{*(k)}}
\end{pmatrix} =
\begin{pmatrix}
I_{n_k} & -Q^{(k)}T^{(k)} \\
0_{n_k} & I_{n_k}
\end{pmatrix}
\begin{pmatrix}
\overrightarrow{B_1^{(k)}} \\
\overrightarrow{B_2^{(k)}} & \overrightarrow{B_1^{*(k)}} & \overrightarrow{B_2^{*(k)}}
\end{pmatrix},
$$
where $\overline{B}_1^{(k)} = (b_1^{(k)}, \ldots, b_{n_k}^{(k)})^T$, $\overline{B}_2^{(k)} = (b_{n_k+1}^{(k)}, \ldots, b_{2n_k}^{(k)})^T$, $\overline{D}_1^{(k)} = (b_1^{(k)}, \ldots, b_{n_k}^{(k)})^T$, $\overline{D}_2^{(k)} = (b_{n_k+1}^{(k)}, \ldots, b_{2n_k}^{(k)})^T$, $\overline{D}_1^{(k)} = (d_1^{(k)}, \ldots, d_{n_k}^{(k)})^T$, $\overline{D}_2^{(k)} = (d_{n_k+1}^{(k)}, \ldots, d_{2n_k}^{(k)})^T$.

For $k = 1, 2$, the new bases are $D^{(k)} = (b_1^{(k)}, \ldots, b_{n_k}^{(k)}, b_{n_k+1}^{(k)}, \ldots, b_{2n_k}^{(k)})$ and $D^{*(k)} = (d_1^{(k)}, \ldots, d_{n_k}^{(k)}, b_{n_k+1}^{(k)}, \ldots, b_{2n_k}^{(k)})$. It is clear that $D^{(k)}$ and $D^{*(k)}$ are dual orthonormal, and are distributed the same as the original bases $B^{(k)}$ and $B^{*(k)}$ respectively.

The secret keys and challenge ciphertext $\{(k^{*\nu}\theta_j^{(k)})_{j=1,\ldots,\nu}, C\}$ in Game 2-$\nu$ are expressed over the bases $B^{(k)}$ and $B^{*(k)}$, $k = 0, 1, 2$ as follows:

$$k_{0,j} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{*(0)}},$$
$$k_{1,j} = (\alpha_j^{(1)}c_1^{(1)} + \beta_j^{(1)}x_j, \gamma_j^{(1)}1, \ldots, \gamma_j^{(1)}n_{1,j}, \eta_j^{(1)}1, \eta_j^{(1)}n_{1,j}, 1, 0)_{B^{*(1)}},$$
$$\forall x \in \mathbb{P}(I) : k_{x,j} = (\alpha_j^{(2)} + \beta_j^{(2)}1x_j, \gamma_j^{(2)}1, \ldots, \gamma_j^{(2)}n_{2,j}, \eta_j^{(2)}1, \eta_j^{(2)}n_{2,j}, 1, 0)_{B^{*(2)}},$$
$$c_0 = (\delta, w_1, \zeta, 0, \varphi)_{B^{*(0)}},$$
$$c_1 = \left(\delta y_1^{(b)}, w_1^{(2)}, 1\right)_{B^{*(1)}},$$
$$\forall x \in \text{RevokeNodes}(\text{Tree}, L^{(b)}) : c_x = \left(\delta(1-1\mu), w_1^{(2)}, 1\right)_{B^{*(2)}},$$
$$c_M = \left(\check{\gamma} M^{(b)}\right).$$

The above keys and challenge ciphertext can also be expressed over bases $D^{(k)}$ and $D^{*(k)}$, $k = 0, 1, 2$ as specified in the following. The first components of secret keys can be expressed as $k_{0,j} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{*(0)}} = (-\alpha_j, \theta_j, 1, \eta_j, 0)_{B^{*(0)}}$, where $\theta_j = \epsilon_j - \lambda$ are uniform and independent since $\epsilon_j \sim \mathbb{F}_q$. Similarly, other key components can be represented as

$$k_{1,j} = \left(\alpha_j^{(1)}c_1^{(1)} + \beta_j^{(1)}x_j, \gamma_j^{(1)}1, \ldots, \gamma_j^{(1)}n_{1,j}, \eta_j^{(1)}1, \eta_j^{(1)}n_{1,j}, 1, 0\right)_{D^{*(1)}},$$
$$\forall x \in \mathbb{P}(I) : k_{x,j} = \left(\alpha_j^{(2)} + \beta_j^{(2)}1x_j, \gamma_j^{(2)}1, \ldots, \gamma_j^{(2)}n_{2,j}, \eta_j^{(2)}1, \eta_j^{(2)}n_{2,j}, 1, 0\right)_{D^{*(2)}}.$$
For any adversary $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{(3)}(\lambda) = 0$. 

**Lemma 5.14.**
Proof. The value of $b$ is independent from the adversary’s view in Game 3. Hence, $\text{Adv}_A^{(3)}(\lambda) = 0$.

5.5 Conclusion

In this chapter, we proposed a revocable predicate encryption with full hiding. In the scheme, no information about revoked users is leaked from a given ciphertext and is a natural extension in the context of PE that cares about privacy. The proposed scheme is proven secure under standard model under a well established assumption. In the next chapter, we will study another important issue in PE, i.e., forward security, which offers protection for the private keys when they get compromised.
Chapter 6

A Forward-Secure Hierarchical Predicate Encryption Scheme

This chapter focuses on the problem of forward security in PE. We propose the first forward-secure hierarchical predicate encryption scheme, which also implies the first forward-secure predicate encryption. We provide syntax and security definitions. We also analyze that our scheme is adaptively attribute-hiding against chosen plaintext attacks under the DLIN assumption in the standard model.

6.1 Introduction

Forward Security. Forward Security (FS) offers meaningful protection in cryptographic applications with long-term (i.e. static) private keys in the unfortunate case when these keys become compromised. Being a standard requirement in authenticated key exchange protocols, where it also takes its origin \[23,29\], forward security has further been explored in digital signatures \[4,31\] and in public key encryption (PKE) \[17\]; see the work of Itkis and Reyzin \[31\] for a nice survey and strong motivation of forward security. The concept of time evolution is central to forward security since from the moment the private key is exposed the intended security goals can no longer be guaranteed and the key must be changed. FS aims to tame potential damage by offering protection with respect to earlier time periods. For example, in forward secure digital signatures signing keys that are exposed in one time period cannot be used to forge signatures related to prior time periods. Similarly, in the case of forward secure encryption, decryption keys used in one time period cannot be used to decrypt ciphertexts generated in the past.

The first forward-secure PKE scheme, due to Canetti, Halevi, and Katz \[17\], was built from the technical tool called binary tree encryption \[33\], which in turn is implied by Hierarchical IBE (HIBE) \[26,30\] by considering identities as nodes of the tree and restricting the intermediate nodes to have exactly two descendants: a parent node with identity string \(id \in \{0,1\}^l\) is split.
into two child nodes with identities $id_0, id_1 \in \{0, 1\}^{\ell+1}$. For each node $id$ there exists a secret key $SK_{id}$, which can be used to derive secret keys $SK_{id_0}$ and $SK_{id_1}$ in a one-way fashion. The intuition behind FS-PKE is to split the entire lifetime of the scheme into $N$ time periods and construct a binary tree with depth $\log N$, where each node corresponds to a unique time period. In order to encrypt a message for some time period $i \in [1, N]$ one uses the master public key of HIBE and the identity string $id_i$ of the node $i$. At any period $i \in [1, N]$ the private decryption key of the user contains the secret key $SK_{id_i}$ as well as secret keys for all right siblings of the nodes on the path from the root to node $i$. The latter keys can be used to derive secret keys $SK_{id_j}$ for all subsequent periods $j \in [i, N]$. The actual FS property is obtained by erasing $SK_{id_i}$ (and all secret keys that can be used to derive it) from the private key upon transition to period $i + 1$.

These ideas were extended by Yao et al. [62] to obtain FS in the identity-based setting. More precisely, they came up with a forward-secure HIBE (FS-HIBE) constructed via a “cross-product” combination of two HIBE schemes, in the random oracle model. Boneh, Boyen, and Goh [9] offered more efficient FS-HIBE constructions, with selective security in the standard model and with adaptive security in the random oracle model. The first adaptively secure FS-HIBE scheme in the standard model is due to Lewko and Waters [37]. As mentioned by Boyen and Waters [16] and also explored in [19, 21, 51, 52, 56] FS is also achievable for anonymous HIBE systems, whose ciphertexts hide the (hierarchy of) identities for which messages were encrypted. Since HIBE generalizes IBE, (anonymous) FS-HIBE covers (anonymous) FS-IBE.

**Forward Security in ABE/PE.** A message encrypted with an ABE/PE scheme can potentially be decrypted by many users. Exposure of some user’s private key in these schemes is likely to cause more damage in comparison to PKE or IBE schemes since the adversary could obtain messages that were encrypted for more than one user. Adding forward security to ABE/PE schemes is thus desirable to alleviate this problem. A naïve approach, i.e., to change all keys (incl. public ones) for each new time period, has already been ruled out as being impractical in PKE and IBE schemes, and it seems even more complicated in the ABE/PE setting. In this work we formalize and construct the first forward-secure hierarchical predicate encryption (FS-HPE). Since HPE includes PE/ABE [35,42], our FS-HPE scheme also implies constructions of first forward secure ABE/PE schemes.

Although forward-secure HIBE constructions exist, formalizing and designing FS-HPE is challenging due to a number of advanced properties that must be considered. In HPE schemes predicates (and by this indirectly private keys) are organized in a hierarchy — any ciphertext that can be decrypted by a low-level predicate must also be decryptable by a high-level predicate but the converse may not be true. In contrast to HIBE, where delegation is performed by extending the parent identity with a substring, predicates in HPE have more complex structures and their delegation requires different techniques. Moreover, predicates should be delegatable at any period in time, irrespective of time evolution for FS. Another aspect is that encryption of messages in forward-secure HPE must be possible only using the master public key, the set of attributes, and the current time period, without having a priori knowledge of predicates at any level of the hierarchy, whereas in FS-HIBE schemes encryption is performed with respect to a given identity at one of the hierarchy levels. As we will discuss in Section 6.1.2 obtaining
forward security in HPE schemes by applying techniques from existing FS-PKE [17] and FS-HIBE [62] results in a number of obstacles. For example, a “cross-product” combination of two HPE schemes [35, 42], akin to the case of two HIBE schemes for FS-HIBE in [62], seems not feasible due to the unique delegation and randomization mechanisms used in those HPE schemes. Finally, an FS-HPE scheme should still provide attribute hiding, which could be threatened if (public) time periods for FS are mixed up with attributes during the encryption.

6.1.1 Contributions of this chapter

FS-HPE: Model and Scheme. We formalize and design the first forward-secure hierarchical predicate encryption (FS-HPE) scheme, for zero-inner-product predicates [34]. Our scheme is secure (adaptively attribute-hiding) in the standard model under the well-known Decision Linear (DLIN) assumption [10] in bilinear groups of prime order. We first present a new syntax and security definitions that are specific to FS-HPE, in particular definition of attribute hiding had to be extended in order to account for FS, in a more complex way than in existing FS-HIBE definitions [62], as explained in Section 6.2.3. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates (to support dynamic behavior for delegation of decrypting rights to new users), local update for users’ private keys (i.e., no master authority needs to be contacted), forward security, and the scheme’s encryption process does not require knowledge of predicates at any level including when those predicates join the hierarchy. Considering the relationships amongst the encryption flavors, we can restrict our scheme to one-level hierarchy and obtain the first adaptively-secure FS-PE/ABE construction, or we can set the inner-product predicate to perform the equality test, in which case we would obtain the first adaptively-secure anonymous FS-HIBE scheme under the basic DLIN assumption (as an alternative to De Caro et al. [19] that works in bilinear groups of composite order and requires new hardness assumptions).

Techniques. Our FS-HPE scheme is built based on the dual system encryption approach introduced by Waters [61] and uses the concept of dual pairing vector spaces (DPVS) of Okamoto and Takashima [42]. Techniques underlying forward security of the scheme can be seen as reminiscent of binary tree encryption [17] that was invented for FS-PKE and does not apply immediately to the more complex HPE setting. We had to resort to those techniques and modify them for integration with HPE since obtaining FS-HPE in a more direct way, e.g. by adopting the “cross-product” idea from [62], seems not feasible with existing HPE constructions [35, 42]. On a high level, we modify the existing HPE scheme from Lewko et al. [35] and combine two of its instances in a non-trivial way to achieve a FS-HPE scheme. One of the HPE schemes handles predicate/attribute hierarchy while another one is used for maintaining time periods using the concept behind binary tree encryption [17]. The modification of the scheme [35] is necessary to prove security involving FS. The combination of the two schemes is non-trivial due to the delegation and randomization components inherited from HPE. Our scheme perfectly synchronizes all private key components (decryption, delegation and randomization) from both HPE instances. These components are updated at each new time period and they are also used for time-independent delegation of predicates. We apply game-hopping
proofs, following the general proof strategy from Okamoto and Takashima [43], i.e. we first define several hard problems and prove that security of our scheme relies on them, then we prove that those hard problems can individually be used to solve the DLIN problem.

6.1.2 Initial Attempts, or Why FS-HPE is Challenging?

We first discuss several initial attempts to construct FS-HPE and illustrate why combining HPE [35][12] with FS techniques from Canetti et al. [17] is far from being trivial. Informally, in HPE [35][12] for inner-product relation, hierarchical attributes are represented by \((\overrightarrow{y_1}, \ldots, \overrightarrow{y_h})\) and hierarchical predicate vectors are defined as \((\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})\) such that \(f(\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})(\overrightarrow{y_1}, \ldots, \overrightarrow{y_h}) = 1\) iff \(l \leq h\) and \(\overrightarrow{x_i} \cdot \overrightarrow{y_i} = 0\) for \(1 \leq i \leq l\).

**First Attempt.** Consider the following intuitive modification of the HPE scheme, e.g. [35]. The user with predicate vectors \((\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})\) maintains two subtrees with the same root labeled as \((\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})\): the time subtree that evolves over time for forward security (using the concept from [17]), and the predicate subtree to which, in case of delegation, further children can be added to expand the hierarchy. The time subtree can be viewed as a special predicate subtree where each time node \(ID_t\) is denoted by a two dimensional predicate vector \([34]\), for example \(\overrightarrow{x}_{ID_t} = (-ID_t, 1)\). To encrypt a message at time \(t\), the sender uses attribute vectors \((\overrightarrow{y_1}, \ldots, \overrightarrow{y_h}, \overrightarrow{y}_{ID_t})\), where \(ID_t\) denotes the node for time \(t\) and \(\overrightarrow{y}_{ID_t} = (1, ID_t)\). This message can be decrypted, provided that the receiver knows the secret key for time \(t\) and the ciphertext’s attributes \((\overrightarrow{y_1}, \ldots, \overrightarrow{y_h})\) satisfy his private key predicates \((\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})\).

The limitation of this approach is that key delegation is not independent of the time period. To guarantee forward security the user must erase the secret key corresponding to the root \((\overrightarrow{x_1}, \ldots, \overrightarrow{x_l})\), as its exposure would compromise the secrecy of derived keys for earlier time.
periods. However, if this secret key is erased then the predicate \((\bar{x}_1, \ldots, \bar{x}_l)\) can no longer be expanded, i.e. the scheme will not be able to support hierarchical key delegation at the level of that predicate. This attempts indicates that all secret keys must evolve together.

**Third Attempt.** Consider a more direct FS-HPE construction where all private keys evolve over time. Assume that attribute vectors (resp. predicate vectors) consist of alternating attributes (resp. predicates) and time identifiers, which are referred to as an attribute-time-tuple (resp. predicate-time-tuple). The private key at each node serves three purposes: decryption, hierarchical delegation, and derivation of private keys for the next time period. Predicate vectors associated with the private key of a newly added node are defined by the predicate-time-tuple of its parent extended with the node-specific predicate vector. This private key can in turn be used to derive further keys for its descendants. For example, if \(\bar{x}_2\) extends the root \(\bar{x}_1\) at time \(t_1\) and \(\bar{x}_3\) extends \(\bar{x}_2\) at time \(t_2\), the hierarchical predicate for the third node at time \(t_2\) is given by \((\bar{x}_1, \bar{x}_{ID_1}, \bar{x}_2, \bar{x}_{ID_2}, \bar{x}_3)\), where \(\bar{x}_{ID_1} = (-ID_1, 1)\) and \(\bar{x}_{ID_2} = (-ID_2, 1)\) (as in the previous attempt).

In order to decrypt a message using this hierarchical predicate the ciphertext must contain attributes for the time periods \(t_1\) and \(t_2\). That is, the sender must know all time periods at which different nodes on the path joined the hierarchy. Clearly, this is a limitation in terms of both scalability and privacy. That is, different private keys within the predicate hierarchy should ideally evolve in time that remains transparent to the encryption algorithm.

Our exposition above shows that designing FS-HPE is not straightforward even though the forward security concept has been known in other encryption flavors. Our scheme, presented in Section 6.3, that is based on a variant of the HPE scheme from Lewko et al. [35] and a reminiscent of the binary tree encryption from Canetti et al. [17], includes additional tricks to overcome the problems demonstrated above.

### 6.2 Security Model

In this section we present our model for forward secure hierarchical predicate encryption (FS-HPE). First, we highlight the idea behind the FS-HPE concept and introduce some notation. In FS-HPE private keys are associated with predicate vectors and evolve over the time. At any time period \(i\) a user may join the hierarchy and receive delegated private keys. These keys are computed by the parent user for time period \(i\) and together with further secret information that is necessary to derive private keys for later time periods is handed over to the joined user. Once the user receives this secret information, at the end of each period the user updates his private key locally and erases secrets that are no longer needed. Additionally, at any time \(j \geq i\) the user may delegate its private key down the hierarchy without contacting its parent. In any time period \(i\) a message can be encrypted using public parameters, the attribute vectors, and \(i\). In order to decrypt for time period \(i\) users must possess private keys satisfying attributes from the ciphertext for that time.
6.2.1 Notation

Time Period Let the total number of time periods \( N = 2^\kappa \), where \( \kappa \in \mathbb{N} \).

Hierarchical Inner-Product Predicate Encryption We borrow some notation from Lewko et al. [35] to describe our HPE with inner-product predicates. Let \( \vec{\mu} = (n; d, \mu_1, \ldots, \mu_d) \) be a tuple of positive integers such that \( \mu_0 = 0 < \mu_1 < \mu_2 < \cdots < \mu_d = n \). We call \( \vec{\mu} \) a format of hierarchy of depth \( d \) attribute spaces. With \( \Sigma_i, l = 1, \ldots, d \) we denote attribute sets and each \( \Sigma_i = F_q^{\mu_i-\mu_{i-1}} \setminus \{0\} \). A hierarchical attribute \( \Sigma = \cup_{i=1}^d (\Sigma_1 \times \ldots \times \Sigma_i) \) is defined using the disjoint union. For \( \vec{x}_i \in F_q^{\mu_i-\mu_{i-1}} \setminus \{0\} \), a hierarchical attribute \( (\vec{y}_1, \ldots, \vec{y}_h) \in \Sigma \) is said to satisfy a hierarchical predicate \( f(\vec{x}_1, \ldots, \vec{x}_i) \) iff \( l \leq h \) and \( \vec{y}_1 \cdot \vec{x}_i = 0 \) for \( 1 \leq i \leq l \), which we denote as \( f(\vec{x}_1, \ldots, \vec{x}_i)(\vec{y}_1, \ldots, \vec{y}_h) = 1 \). The space of hierarchical predicates is \( \mathcal{F} = \{ f(\vec{x}_1, \ldots, \vec{x}_i) | \vec{x}_i \in F_q^{\mu_i-\mu_{i-1}} \setminus \{0\} \} \). We call \( h \) (resp. \( l \)) the level of \( (\vec{y}_1, \ldots, \vec{y}_h) \) (resp. \( (\vec{x}_1, \ldots, \vec{x}_i) \)). Throughout the chapter we will assume that an attribute vector \( \vec{y}_1 = (y_1, \ldots, y_{\mu_1}) \) is normalized such that \( y_1 = 1 \) (note that \( \vec{y}_1 \) can be normalized via \( (1/y_1) \cdot \vec{y}_1 \), assuming that \( y_1 \) is non-zero). By \( \vec{x}_i^{(k)} \) we denote the canonical basis vector \( (0, \ldots, 0, 1, 0, \ldots, 0) \) \( k = 1, 2 \) and \( i = 1, \ldots, n_k \), where \( n_1 \) and \( n_2 \) are for hierarchical predicates and time tree respectively.

Keys We will work with two notations for secret keys: \( sk_{w, l}(\vec{x}_1, \ldots, \vec{x}_i) \) is the key associated with some prefix \( w \) of the bit representation of a time period \( l \) and a hierarchical predicate \( (\vec{x}_1, \ldots, \vec{x}_i) \), whereas \( SK_{i,l}(\vec{x}_1, \ldots, \vec{x}_i) \) denotes the key associated with time \( i \) and a hierarchical predicate \( (\vec{x}_1, \ldots, \vec{x}_i) \). That is, \( SK_{i,l}(\vec{x}_1, \ldots, \vec{x}_i) = \{ sk_{w,l}(\vec{x}_1, \ldots, \vec{x}_i), sk_{w,1}(\vec{x}_1, \ldots, \vec{x}_i) : w \text{ is a prefix of } i \} \) to simplify the presentation, we denote the keys as \( sk_{w,l} \) and \( SK_{i,l} \) respectively.

6.2.2 Syntax

Definition 6.1. A Forward Secure Hierarchical Predicate Encryption Scheme (FS-HPE) is a tuple of five algorithms \((\text{RootSetup}, \text{Delegate}, \text{Update}, \text{Encrypt}, \text{Decrypt})\) described in the following:

RootSetup\((1^\lambda, N, \vec{\mu})\) This algorithm takes as input a security parameter \( 1^\lambda \), the total number of time periods \( N \) and the format of hierarchy \( \vec{\mu} \). It outputs public parameters of the system, including public key \( PK \), and a root secret key \( SK_{0,1} \), which is assumed to be known only to the master authority of the hierarchy.

Delegate\((SK_{i,l}, i, \vec{x}_{i+l})\) This algorithm takes as input a secret key \( SK_{i,l} \) associated with time \( i \) on hierarchy level \( l \) and an \((l+1)\)-th level predicate vector \( \vec{x}_{i+l} \). It outputs the delegated secret key \( SK_{i,l+1} \). This key is intended for the direct descendant at level \( l+1 \). It is assumed that predicate vector \( \vec{x}_{i+l} \) is added to the predicate hierarchy during the time period \( i \).

Update\((SK_{i,l}, i)\) This algorithm takes as input a secret key \( SK_{i,l} \) and the current time period \( i \). It outputs an updated secret key \( SK_{i+1,l} \) for the following time period \( i+1 \) and erases \( SK_{i,l} \).
6.2. Security Model

Definition 6.2. A

6.2.3 Security Definition

Correctness. For all correctly generated PK and SK_{i,l} associated with predicate vectors \((\mathcal{P}_1, \ldots, \mathcal{P}_l)\) and a time period \(i\), let \(C \leftarrow \text{Encrypt}(PK, (y_1^{(0)}, \ldots, y_h^{(0)}), i, M)\) and \(M' = \text{Decrypt}(C, SK_{i,l})\). Then, if \(f(\mathcal{P}_1, \ldots, \mathcal{P}_l)(y_1^{(0)}, \ldots, y_h^{(0)}) = 1\) then \(M = M'\); otherwise, \(M \neq M'\) with all but negligible probability.

6.2.3 Security Definition

Definition 6.2. A FS-HPE scheme is adaptively attribute hiding against chosen plaintext attacks if for all PPT adversaries \(A\), the advantage of \(A\) in the following game is negligible in the security parameter:

Setup. RootSetup algorithm is run by the challenger \(C\) to generate public key \(PK\) and root secret key \(SK_{0,1}\). \(PK\) is given to \(A\).

Query phase 1. \(A\) may adaptively make a polynomial number of delegation queries by asking \(C\) to create a secret key for any given time period \(i\) and hierarchical predicate vectors \((\mathcal{P}_1, \ldots, \mathcal{P}_l)\). In response, \(C\) computes the secret key \(SK_{i,l}\) and reveals it to \(A\).
(Note that \(C\) computes \(SK_{i,l}\) with the help of algorithms Delegate and Update that it may need to execute several times, i.e. depending on the input time period \(i\) and hierarchy level \(l\).)

Challenge. \(A\) outputs its challenge, containing two attribute vectors \((Y^{(0)}, Y^{(1)}) = ((y_1^{(0)}, \ldots, y_h^{(0)}), (y_1^{(1)}, \ldots, y_h^{(1)}))\), two plaintexts \((M^{(0)}, M^{(1)})\), and a time period \(I\), such that

- either \(i > I\), or
- \(i \leq I\) and \(f(\mathcal{P}_1, \ldots, \mathcal{P}_l)(y_1^{(0)}, \ldots, y_h^{(0)}) = f(\mathcal{P}_1, \ldots, \mathcal{P}_l)(y_1^{(1)}, \ldots, y_h^{(1)}) = 0\)

for each revealed key for \(f(\mathcal{P}_1, \ldots, \mathcal{P}_l)\) at time period \(i\).

\(C\) then flips a random coin \(b\). If \(b = 0\) then \(A\) is given \(C = \text{Encrypt}(PK, Y^{(0)}, I, M^{(0)})\) and if \(b = 1\) then \(A\) is given \(C = \text{Encrypt}(PK, Y^{(1)}, I, M^{(1)})\).

Query phase 2. Repeat the Query phase 1 subject to the restrictions as in the challenge phase.

Guess. \(A\) outputs a bit \(b'\), and succeeds if \(b' = b\).

We define the advantage of \(A\) as the quantity \(\text{Adv}^{\text{FS-HPE}}_A(\lambda) = |\Pr[b = b'] - 1/2|\).
Remark 6.1. In Definition 6.2, adversary $A$ is not allowed to ask a key query for time period $i$ and hierarchical predicate vectors $(\vec{x}_1, \ldots, \vec{x}_l)$ such that $i \leq I$ and $f_1(\vec{x}_1, \ldots, \vec{x}_l)(\vec{y}_{b_1}^{(b)}, \ldots, \vec{y}_{b_h}^{(b)}) = 1$ for some $b \in \{0, 1\}$, i.e., the queried key is not allowed to decrypt the challenge ciphertext. Recently, Okamoto and Takashima [46] proposed a PE (HPE) which allow such key query, provided that $M^{(0)} = M^{(1)}$. The technique of Okamoto and Takashima [46] can be applied in our scheme to achieve strong security.

Remark 6.2. In Definition 6.2, $A$ may ask delegation queries and obtain the resulting keys. This contrasts slightly with the HPE security definition in [35], where $A$ may ask the challenger to create and delegate private keys but will not be given any of them, unless it explicitly asks a separate reveal query. This is because HPE in [35] has two algorithms for computing secret keys, either directly (using the master secret key) or through delegation (using secret key of the parent node). In our FS-HPE syntax we compute secret keys through delegation only which is the similar to that of [62] and in the security definition we are mainly concerned with maintaining time evolution for delegated keys.

Remark 6.3. Definition 6.3 can be easily extended to address chosen-ciphertext attacks (CCA) by allowing decryption queries. The usual restriction is that decryption queries cannot be used for the challenge ciphertext. Our CPA-secure FS-HPE scheme from Section 6.3 can be strengthened to resist CCA by applying the well-known CHK transformation from [18] that uses one-time signatures to authenticate the ciphertext.

6.3 Our Scheme

High-Level Description. For simplicity of presentation, our FS-HPE makes use of a version of FS-PKE scheme by Katz [32]. In Katz’s scheme, time periods are associated with the leaf nodes of a binary tree while in Canetti et al. scheme [17], time periods correspond to all nodes of the tree. Our scheme can also be realized based on the FS-PKE scheme by Canetti et al., which will give faster key update time. We utilize a full binary tree of height $\kappa$, whose root is labeled $\epsilon$ and all other nodes are labeled recursively: if the label of a node is $w$, then its left child is $w0$, and its right child is $w1$. Each time period $i \in \{0, \ldots, N - 1\}$ corresponds to a leaf identified via the binary representation of $i$. We denote the $k$-bit prefix of a $d$-length word $w = w_1w_2\ldots w_d$ by $w|_k$, i.e. $w|_k = w_1w_2\ldots w_k$ for $k \leq d$. Let $w|_0 = \epsilon$ and $w = w|_d$.

We use two HPE schemes in parallel. Private keys in each scheme contain three components: decryption, delegation and randomness. Private key of a user contains private keys from both schemes that are linked together using secret sharing. One HPE scheme is used to handle predicate/attribute hierarchy, while the other one is used to handle time evolution. Each of the two HPE schemes is a modification of the scheme in [35], in a way that allows us to prove attribute-hiding property under more sophisticated conditions involving time evolution. The efficiency of the modified scheme is still comparable to the one in [35], i.e. it increases the ciphertext by an additional component (master component) that is used to combine both HPE schemes and is crucial for the security proof. This change implies that the length of the orthonormal bases grows from $(2n + 3) \cdot |G|$ in [35] to $(3n + 1) \cdot |G|$ in our scheme, where $n$ is the dimension of the attribute vectors, and $|G|$ is the length of a group element from $G$. 
At time period $i$, the entity at level $l$ with a hierarchical predicate ($\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_l$) holds a secret key $SK_i(\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_l)$, denoted for simplicity as $SK_{i,l}$. It contains secret keys $sk_i,l$ and \{sk$_{w,l}$\} for each label $w$ corresponding to a right sibling node (if one exists) on the path from $i$ to the root. We view $sk_{i,l}$ as a decryption key, which is associated with current time $i$ and the predicate ($\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_l$). The secret keys in \{sk$_{w,l}$\} contain auxiliary information used to update $SK_{i,l}$ for future time periods and to derive its lower-level predicates. The initial keys $sk_{0,1}$ and $sk_{1,1}$ are computed in the RootSetup algorithm and are associated with the predicate $\overrightarrow{\mathcal{P}}_1$. In general, each $sk_{w,l}$ contains three secret components: the decryption component $(k_{w,l,dec}^{(0)}, k_{w,l,dec}^{(1)}, k_{w,l,dec}^{(2)})$, the randomness component $(k_{w,l,ran,1}, k_{w,l,ran,1+l}, k_{w,l,ran,1+2l}, \ldots)$ and the delegation component $(k_{w,l,del,1}, k_{w,l,del,1+l}, k_{w,l,del,1+2l}, \ldots)$, where $L = 2\kappa$. All above components are constructed using orthonormal bases $\mathbb{B}$ generated by $\mathcal{G}_{ob}$. There are three different bases in the system, i.e., $\mathbb{B}^{(0)}, \mathbb{B}^{(1)}$ and $\mathbb{B}^{(2)}$. The superscript of each key component denotes its base. $k_{w,l,dec}$ is the mentioned master component that links $k_{w,l,dec}^{(1)}$ and $k_{w,l,dec}^{(2)}$ using the secret sharing techniques. In turn, $k_{w,l,dec}^{(1)}$ and $k_{w,l,dec}^{(2)}$ are used in respective HPE schemes. If $w$ represents a leaf of the binary tree then the decryption component $(k_{w,l,dec}^{(0)}, k_{w,l,dec}^{(1)}, k_{w,l,dec}^{(2)})$ is used for decryption at time represented by $w$.

Delegation and randomization of private keys are processed similarly as in [35], except that upon derivation of keys for lower level predicates, we also delegate and randomize their time-dependent part. In particular, the delegation component of the $l$-th level key is essential to compute the $(l+1)$-th level child key, and the randomness component of the $l$-th level key is used to re-randomize the latter’s coefficients. To handle time hierarchy, we deploy “dummy” nodes. Similarly, we will compute the dummy child for predicate hierarchy when time evolves. In this way, all derived keys are re-randomized.

We define a helper algorithm ComputeNext that will be called from RootSetup and Update. Given a secret key $sk_{w,l}$ for node $w$ and a hierarchical predicate ($\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_l$) it outputs $sk_{w(b),l}$, $b \in \{0, 1\}$ for the nodes $w0$ and $w1$ by updating the three components of $sk_{w,l}$. The algorithm Update computes secret keys for the next time period through the internal call to ComputeNext and erases all secret information that was used to derive the key for the current time period. The update procedure involves all three components of the secret key. For example, for a given secret key $SK_{i,l} = (sk_{i,l}, \{sk_{w,l}\})$, forward security is achieved by deleting $sk_{i,l}$ and using all three components of \{sk$_{w,l}$\} to derive $SK_{i+1,l}$ for the following time period with the help of ComputeNext.

In algorithm Delegate, a secret key $sk_{w,l}$ for a string $w$ is used to derive $sk_{w,u}$ for a lower hierarchy level $u > l$ and a hierarchical predicate ($\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_u$) that has restricted capabilities in comparison to ($\overrightarrow{\mathcal{P}}_1, \ldots, \overrightarrow{\mathcal{P}}_l$). As mentioned, the delegation component for hierarchical predicates of $sk_{w,l}$ is essential for the derivation of $sk_{w,u}$, whose coefficients are re-randomized with the randomization component.

The algorithm Encrypt requires only a time period $t$ and a hierarchical attribute ($\overrightarrow{\mathcal{Y}}_1, \ldots, \overrightarrow{\mathcal{Y}}_h$) to encrypt the message. We note that during encryption attributes ($\overrightarrow{\mathcal{Y}}_1, \ldots, \overrightarrow{\mathcal{Y}}_h$) are extended with random elements from level $h + 1$ down to the leaf, i.e., the scheme encrypts attribute vectors on all levels in the hierarchy instead of encrypting only the input vectors. In this way, parent keys can directly decrypt ciphertexts produced for their children without taking effort to derive child keys first.
The algorithm Decrypt uses the decryption key $sk_{i,1}$, which is associated with time period $i$ and hierarchical predicate ($\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l$). The message is decrypted iff the attributes in the ciphertext satisfy the predicates in the decryption component of the key and the ciphertext is created at time $i$.

**Detailed Description.** The five algorithms of our FS-HPE scheme are detailed in the following: (note that $G_{\text{ob}}$ is defined in Section 4.3)

**RootSetup**($1^\lambda, N = 2^n, \overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)$):

Let $\overrightarrow{x}_1$ be the root predicate and let $L = 2\kappa$ and $\overrightarrow{\nu} = (2; n, L)$. Compute

$$(\text{param}_{\overrightarrow{\mu}}, \overrightarrow{B}^{(0)}, \overrightarrow{B}^{(1)}, \overrightarrow{B}^{(2)}, \overrightarrow{B}^{(3)} ) \xleftarrow{\$} G_{\text{ob}}(1^\lambda, \overrightarrow{\nu}),$$

\[ \overrightarrow{B}^{(0)} = (b_1^{(0)}, b_2^{(0)}, b_3^{(0)}), \overrightarrow{B}^{(1)} = (b_1^{(1)}, \ldots, b_n^{(1)}, b_{n+1}^{(1)}), \overrightarrow{B}^{(2)} = (b_1^{(2)}, \ldots, b_L^{(2)}, b_{3L+1}^{(2)}), \]

\[ \overrightarrow{B}^{(3)} = (b_1^{(3)}, \ldots, b_{3n+1}^{(3)}), \overrightarrow{B}^{(4)} = (b_n^{(4)}, \ldots, b_{3L+1}^{(4)}). \]

The master authority needs to generate not only the secret key associated with the current time period 0 but also secret keys corresponding to the internal nodes on the binary tree whose bit representations are all 0 except for the last bit. The secret key for time 0 and predicate $\overrightarrow{x}_1$ is denoted as $sk_{0,1}$. Secret keys that will be used to derive keys for future time periods are denoted as \{sk$_{1,1}, sk_{0(1)}, \ldots, sk_{k_{n-1},1}\}$. These values are generated recursively as follows, starting with $sk_{0,1}$ and $sk_{1,1}$.

**Computing $sk_{0,1}$**: Pick $\psi, \psi', \alpha_{\text{dec}}, \alpha_{\text{dec}}^{(1)}, \alpha_{\text{dec}}^{(2)}, \psi^{(1)} \in \mathbb{F}_q$ such that $\alpha_{\text{dec}} = \alpha_{\text{dec}}^{(1)} + \alpha_{\text{dec}}^{(2)}$. Pick $\eta^{(0)}, \beta^{(1)}_{\text{ran},j,1}(j = 1, 2), \beta^{(2)}_{\text{ran},j,1}(j = 1, 2)$, $\beta^{(1)}_{\text{del},j,1}(j = 1, \ldots, n), \beta^{(2)}_{\text{del},j,1}(j = 1, \ldots, L)$ \[ \mathbb{F}_q, \overrightarrow{\mu} \xleftarrow{\$} G^{(1)}, \overrightarrow{\mu} \xleftarrow{\$} G^{(2)} \] Compute

\[ \begin{array}{ll}
    k_{0,1,\text{dec}}^{(0)} &= (-\alpha_{\text{dec}}, 0, \eta^{(0)}) \in \mathbb{B}_{(0)}, \\
    k_{0,1,\text{dec}}^{(1)} &= \left( \alpha_{\text{dec}}^{(2)}, \mu^{(1)}_{\text{dec}}, 0, \eta^{(1)}_{\text{dec}} \right) \in \mathbb{B}_{(1)}, \\
    k_{0,1,\text{dec}}^{(2)} &= \left( \alpha_{\text{dec}}^{(2)}, \beta^{(1)}_{\text{dec},1}, 0, \eta^{(2)}_{\text{dec}} \right) \in \mathbb{B}_{(2)}, \\
    k_{0,1,\text{ran},j}^{(1)} &= \left( \beta^{(1)}_{\text{ran},j,1}, 0, \eta^{(1)}_{\text{ran},j,1} \right) \in \mathbb{B}_{(1)}, & \text{for } j = 1, 2, \\
    k_{0,1,\text{ran},j}^{(2)} &= \left( \beta^{(2)}_{\text{ran},j,1}, 0, \eta^{(2)}_{\text{ran},j,1} \right) \in \mathbb{B}_{(2)}, & \text{for } j = 1, 2, \\
    k_{0,1,\text{del},j}^{(1)} &= \left( \beta^{(1)}_{\text{del},j,1}, 0, \eta^{(1)}_{\text{del},j,1} \right) \in \mathbb{B}_{(1)}, & \text{for } j = \mu_1 + 1, \ldots, n, \\
    k_{0,1,\text{del},j}^{(2)} &= \left( \beta^{(2)}_{\text{del},j,1}, 0, \eta^{(2)}_{\text{del},j,1} \right) \in \mathbb{B}_{(2)}, & \text{for } j = 3, \ldots, L.
\end{array} \]

Let $sk_{0,1} = (k_{0,1,\text{dec}}^{(0)}, k_{0,1,\text{dec}}^{(1)}, k_{0,1,\text{dec}}^{(2)}, k_{0,1,\text{ran},1}^{(2)}, k_{0,1,\text{ran},2}^{(2)}, k_{0,1,\text{del},1}^{(2)}, \ldots, k_{0,1,\text{del},1}^{(2)}, k_{0,1,\text{del},n}^{(2)}, k_{0,1,\text{del},3}^{(2)}, \ldots, k_{0,1,\text{del},L}^{(2)}).$
6.3. Our Scheme

Let $\sk_{1,1} = (k_{1,1,\dec}, k_{1,1,\ran}, k_{1,2,\dec}, k_{1,2,\ran})^T$.

Recursion: Use $\sk_{0,1}$ to recursively invoke algorithm $\text{ComputeNext}$, i.e. compute

$$(sk_{w,0,1}, sk_{w,0,1,1}) = \text{ComputeNext}(PK, sk_{w,1}, w)$$

Output: Finally, the algorithm outputs public key $PK = (1^\lambda, \param_{\pi}, \{\vec{\sk}^{(k)}\})_{k=0,1,2}, \vec{\sk}^{(2)}, \vec{b}_4^{(0)}$ and the root secret key $SK_0,1 = (sk_{w,1,1}, sk_{w,1,0,1}, \ldots, sk_{w,1,0,1,1})$.

$\text{ComputeNext}(PK, sk_{w,1}, w)$: This is a helper method and is called by the $\text{Root Setup}$ and $\text{Update}$ algorithms. It takes a public key $PK$, a secret key $sk_{w,1}$, a node $w$, and outputs keys $sk_{w,0,1,1}$, $sk_{w,1,1}$ for time nodes $w_0$ and $w_1$ of predicate vectors ($\vec{\pi}_0, \vec{\pi}_1$). Parse $w$ as $w_0, \ldots, w_r$, where $|w| = r$. Parse $sk_{w,1}$ as $k_{w,0,1,\dec}, k_{w,0,1,\ran}, k_{w,1,1,\dec}, k_{w,1,1,\ran}, \ldots$, $k_{w,1,1,\dec}, k_{w,1,1,\ran}$.

Computing $sk_{w,0,1,1}$: Pick $\psi, \psi'$, $c_d, c_{\dec, \ran}, c_{w,1,\dec}, c_{w,1,\ran}$ for $t = 1, \ldots, l + 1$. Pick $c_{\dec, \ran}(j; \sigma_{\dec, \ran}, \sigma_{\ran, j}(j = 1, \ldots, r + 2), \sigma_{\ran, j}(j = 1, \ldots, r + 1), r_{\dec, \ran}(j = 1, \ldots, l + 1), r_{\dec, \ran}(j = 1, \ldots, r + 2), \sigma_{\dec, \ran}(j = 1, \ldots, l))$. Compute

$$k_{w,0,1,\dec}^{(0)} = k_{w,1,1,\dec}^{(0)} + c_{\dec, \ran}(j = 1, \ldots, n) \vec{b}_4^{(0)}$$

$$k_{w,0,1,\dec}^{(1)} = k_{w,1,1,\dec}^{(1)} + \sum_{t=1}^{l+1} c_{\dec, \ran} k_{w,1,\dec, \ran, t} + r_{\dec}^{(0)}$$

$$k_{w,0,1,\dec}^{(2)} = k_{w,1,1,\dec}^{(2)} + \sum_{t=1}^{l+1} c_{\dec, \ran} k_{w,1,\dec, \ran, t} + \sigma_{\dec, \ran} r_{\dec, \ran}(j = 1, \ldots, l + 1) + r_{\dec, \ran}(j = 1, \ldots, r + 2)$$
\[ k_{w,l,ran,j}^{(1)} = \sum_{t=1}^{l+1} \epsilon_{ran,j,t}^{(1)} k_{w,l,ran,t}^{(1)} + \epsilon_{ran,j}^{(1)}, \quad \text{for } j = 1, \ldots, l + 1, \]
\[ k_{w,l,ran,j}^{(2)} = \sum_{t=1}^{r+1} \epsilon_{ran,j,t}^{(2)} k_{w,l,ran,t}^{(2)} + \sigma_{ran,j}^{(2)} k_{w,l,del,2(r+1)} + r_{ran,j}^{(2)}, \quad \text{for } j = 1, \ldots, r + 2, \]
\[ k_{w,l,del,j}^{(1)} = \sum_{t=1}^{l+1} \epsilon_{del,j,t}^{(1)} k_{w,l,del,t}^{(1)} + \psi_{w,l,del,j}^{(1)} + r_{del,j}^{(1)}, \quad \text{for } j = \mu + 1, \ldots, n, \]
\[ k_{w,l,del,j}^{(2)} = \sum_{t=1}^{r+1} \epsilon_{del,j,t}^{(2)} k_{w,l,del,t}^{(2)} + \sigma_{del,j}^{(2)} k_{w,l,del,2(r+1)} + \psi_{w,l,del,j}^{(2)} + r_{del,j}^{(2)}, \quad \text{for } j = 2(r + 1) + 1, \ldots, L. \]

Let \( sk_{w,l} = (k_{w,l,dec}^{(0)} \cdots k_{w,l,del,n}^{(2)}) \).

Computing \( sk_{w,l,j} \): Pick \( \tau, \tau', \epsilon_{dec,t}^{(0)}, \epsilon_{ran,j,t}^{(1)}, \epsilon_{del,t}^{(1)}, \epsilon_{dec,t}^{(2)}, \epsilon_{ran,j,t}^{(2)}, \epsilon_{del,t}^{(2)}, \sum_{t=1}^{l+1} \epsilon_{dec,t}^{(0)} \epsilon_{dec,t}^{(2)} k_{w,l,ran,t}^{(0)} k_{w,l,ran,t}^{(2)} \in \mathbb{F}_q \) for \( t = 1, \ldots, l + 1 \). Pick \( \epsilon_{dec,t}^{(1)}, \epsilon_{dec,t}^{(2)}, \epsilon_{ran,j,t}^{(1)}, \epsilon_{ran,j,t}^{(2)}, \epsilon_{del,t}^{(1)}, \epsilon_{del,t}^{(2)}, \sum_{j=1}^{l+1} \epsilon_{dec,t}^{(1)} \epsilon_{dec,t}^{(2)} k_{w,l,ran,t}^{(1)} k_{w,l,ran,t}^{(2)} \in \mathbb{F}_q \) for \( t = 1, \ldots, r + 1 \), \( \sum_{j=1}^{l+1} \epsilon_{dec,t}^{(1)} \epsilon_{dec,t}^{(2)} k_{w,l,del,t}^{(1)} k_{w,l,del,t}^{(2)} \in \mathbb{F}_q \) for \( t = \mu + 1, \ldots, n \).

\[ k_{w,l,dec}^{(0)} = k_{w,dec}^{(0)} + \epsilon_{dec}^{(0)} t_{dec}^{(0)}, \]
\[ k_{w,l,dec}^{(1)} = k_{w,l,dec}^{(1)} + \sum_{t=1}^{l+1} \epsilon_{dec,t}^{(1)} k_{w,l,ran,t}^{(1)} + t_{dec}^{(1)}, \]
\[ k_{w,l,dec}^{(2)} = k_{w,l,dec}^{(2)} + \sum_{t=1}^{r+1} \epsilon_{dec,t}^{(2)} k_{w,l,ran,t}^{(2)} + \sum_{i=2r+1}^{2r+2} k_{w,l,del,i}^{(2)} + t_{dec}^{(2)}, \]
\[ k_{w,l,ran,j}^{(1)} = \sum_{t=1}^{l+1} \epsilon_{ran,j,t}^{(1)} k_{w,l,ran,t}^{(1)} + t_{ran,j}^{(1)}, \quad \text{for } j = 1, \ldots, l + 1, \]
\[ k_{w,l,ran,j}^{(2)} = \sum_{t=1}^{r+1} \epsilon_{ran,j,t}^{(2)} k_{w,l,ran,t}^{(2)} + \sum_{i=2r+1}^{2r+2} k_{w,l,del,i}^{(2)} + t_{ran,j}^{(2)}, \quad \text{for } j = 1, \ldots, r + 2, \]
\[ k_{w,l,del,j}^{(1)} = \sum_{t=1}^{l+1} \epsilon_{del,j,t}^{(1)} k_{w,l,del,t}^{(1)} + \sum_{i=2r+1}^{2r+2} k_{w,l,del,i}^{(1)} + t_{del,j}^{(1)}, \quad \text{for } j = \mu + 1, \ldots, n, \]
\[ k_{w,l,del,j}^{(2)} = \sum_{t=1}^{r+1} \epsilon_{del,j,t}^{(2)} k_{w,l,del,t}^{(2)} + \sum_{i=2r+1}^{2r+2} k_{w,l,del,i}^{(2)} + t_{del,j}^{(2)}, \quad \text{for } j = 2(r + 1) + 1, \ldots, L. \]

Let \( sk_{w,l,j} = (k_{w,l,j,dec}^{(0)} \cdots k_{w,l,j,del,n}^{(2)}) \).
6.3. Our Scheme

Output: Output \((sk_{w,0,t}, sk_{w,1,t})\).

Delegate\((SK_{i,t}, i, \gamma, t+1 = (x_{i+1}, \ldots, x_{i+\kappa}))\): Parse \(i\) as \(i_1, \ldots, i_\kappa\) where \(\kappa = \log_2 N\). Parse \(SK_{i,t}\) as \((sk_{i,t}, \{sk_{\gamma_{i-1},t}\}_{i_1=0})\). For each \(sk_{w,t}\) in \(SK_{i,t}\) compute \(sk_{w,t+1}\) as follows:

Parse \(w\) as \(w_1, \ldots, w_r\), where \(|w| = r\). Pick \(\psi', \psi'' \in \mathbb{F}_q\) for \(t = 1, \ldots, l + 1\). Pick \(\gamma_{dec, t, ran, j}, \gamma_{ran, t, j} (j = 1, \ldots, r + 1), \sigma_{ran, j} (j = 1, \ldots, l + 2)\) for each \(\gamma_{ran, t, j} (j = 1, \ldots, r + 1), \sigma_{ran, j} (j = 1, \ldots, l + 2)\)

\(\psi, \psi' = \text{span}(b_{2n+1}^{(1)}, \ldots, b_{3n}^{(2)}), r_{dec, t, ran, j} (j = 1, \ldots, r + 1), r_{del, j} (j = 1, \ldots, L)\).

Compute

\[
\begin{align*}
ks_{w,t+1,dec}^{(0)} &= k_{w,t,dec}^{(0)} + \gamma_{dec, t}^{(0)} b_{k}^{(0)}, \\
ks_{w,t+1,dec}^{(1)} &= k_{w,t,dec}^{(1)} + \sum_{t=1}^{l+1} \gamma_{dec, t}^{(1)} k_{w,t,ran, t}^{(1)} + \sigma_{del, j}^{(1)} \sum_{i=\mu_{t+1}}^{\mu_{t+1}} x_{i} k_{w,1,dec}\rangle^{(1)} + r_{dec}^{(1)}, \\
ks_{w,t+1,dec}^{(2)} &= k_{w,t,dec}^{(2)} + \sum_{t=1}^{l+1} \gamma_{dec, t}^{(2)} k_{w,t,ran, t}^{(2)} + r_{dec}^{(2)}, \\
ks_{w,t+1,ran, j}^{(1)} &= \sum_{t=1}^{r+1} \gamma_{ran, j}^{(1)} k_{w,t,ran, t}^{(1)} + \sigma_{ran, j}^{(1)} \sum_{i=\mu_{t+1}}^{\mu_{t+1}} x_{i} k_{w,1,dec}^{(1)} + r_{ran, j}^{(1)} \text{ for } j = 1, \ldots, r + 1, \\
ks_{w,t+1,ran, j}^{(2)} &= \sum_{t=1}^{r+1} \gamma_{ran, j}^{(2)} k_{w,t,ran, t}^{(2)} + r_{ran, j}^{(2)}, \text{ for } j = 1, \ldots, r + 1, \\
ks_{w,t+1,del, j}^{(1)} &= \sum_{t=1}^{r+1} \gamma_{del, j}^{(1)} k_{w,t,ran, t}^{(1)} + \sigma_{del, j}^{(1)} \sum_{i=\mu_{t+1}}^{\mu_{t+1}} x_{i} k_{w,1,dec}^{(1)} + r_{del, j}^{(1)}, \\
ks_{w,t+1,del, j}^{(2)} &= \sum_{t=1}^{r+1} \gamma_{del, j}^{(2)} k_{w,t,ran, t}^{(2)} + r_{del, j}^{(2)}, \text{ for } j = 2r + 1, \ldots, L.
\end{align*}
\]

Let \(sk_{w,t+1} = (k_{w,t+1,dec}^{(0)} k_{w,t+1,dec}^{(1)} k_{w,t+1,dec}^{(2)} k_{w,t+1,ran, 1}^{(1)} k_{w,t+1,ran, 1}^{(2)} \ldots k_{w,t+1,ran, r+1}^{(1)} k_{w,t+1,ran, r+1}^{(2)} k_{w,t+1,del, j}^{(1)} k_{w,t+1,del, j}^{(2)} \ldots k_{w,t+1,del, L}^{(1)} k_{w,t+1,del, L}^{(2)}).

Output \(SK_{i,t+1} = (sk_{i,t+1}, \{sk_{\gamma_{i-1},t}\}_{i_1=0})\) and erase all other information.

Update\((SK_{i,t}, i)\): This algorithm follows the concept from \(\text{[17][32]}\) to compute a private key for the next time period \(i + 1\). Parse \(i\) as \(i_1, \ldots, i_\kappa\) where \(\kappa = \log_2 N\). Parse \(SK_{i,t}\) as \((sk_{i,t}, \{sk_{\gamma_{i-1},t}\}_{i_1=0})\). Erase \(sk_{i_{\gamma_{i-1}},t}\). If \(\kappa = 0\), simply output the remaining keys as the key \(SK_{i+1,t}\) for the next time period. Otherwise, let \(\bar{z}\) be the largest value such that \(i_\bar{z} = 0\). Let \(i' = i_{\bar{z}+1}\). Using \(sk_{i',t}\), which is part of \(SK_{i,t}\), recursively apply algorithm ComputeNext to generate keys \(sk_{i',i_{\bar{z}+1},t}\) for \(0 \leq d \leq \bar{z} - 1\) and \(sk_{i',i_{\bar{z}+1},t}\). (The key \(sk_{i',i_{\bar{z}+1},t}\) will be used for decryption in the next time period \(i + 1\), whereas other generated secret keys will be used to compute
future keys.) Erase \( sk_{i',l} \) and output the remaining keys as \( SK_{(i+1),l} \).

**Encrypt** \( (PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)) = ((y_1, \ldots, y_{\mu_1}), \ldots, (y_{\mu_{h-1}}, \ldots, y_{\mu_h})), i, M \in \mathbb{G}_T \):

Parse \( i \) as \( i_1, \ldots, i_\kappa \). Pick \( (\overrightarrow{y}_{i+1}, \ldots, \overrightarrow{y}_d) \) \( \overset{\leftarrow}{\in} \mathbb{F}_q^{\mu_{h+1} - \mu_h} \times \cdots \times \mathbb{F}_q^{n - \mu_{d-1}}, \delta, \zeta, \varphi(1), \varphi(2) \overset{\leftarrow}{\in} \mathbb{F}_q \),

\[
\begin{align*}
e(0) &= (\delta, 0, \zeta, 0, \varphi)_{B(0)}, \\
e(1) &= (\delta(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_d), 0^{2n}, \varphi(1))_{B(1)}, \\
e(2) &= (\delta(1, -i_1), \ldots, (1, -i_\kappa)), 0^{2L}, \varphi(2))_{B(2)}, \\
e(M) &= g^y M.
\end{align*}
\]

Output ciphertext \( C = (e(0), e(1), e(2), e(M)) \).

**Decrypt** \( (C, SK_{i,l}) \): Parse ciphertext \( C \) as \( (e(0), e(1), e(2), e(M)) \) and secret key \( SK_{i,l} \) as \( (sk_{i,l}, \{sk_{i_{-1},l}\})_{i=0} \). Use \( sk_{i,l} \) to decrypt and output

\[
M = \frac{e(M)}{e(0) e(k_{(0)_{i,l,dec}}) e(1) e(k_{(1)_{i,l,dec}}) e(2) e(k_{(2)_{i,l,dec}})}.
\]

**Correctness.** To see why the scheme is correct, let \( C \) and \( SK_{i,l} \) be as above. If \( \overrightarrow{y}_i, \overrightarrow{y}_{i+1} = 0 \) for \( 1 \leq i \leq l \), and \( C \) and \( SK_{i,l} \) are encoded with the same time period \( i \) then \( M \) can be recovered by computing \( e(M) / e(0) e(k_{(0)_{i,l,dec}}) e(1) e(k_{(1)_{i,l,dec}}) e(2) e(k_{(2)_{i,l,dec}}) \), since

\[
\begin{align*}
e(0) e(k_{(0)_{i,l,dec}}) e(1) e(k_{(1)_{i,l,dec}}) e(2) e(k_{(2)_{i,l,dec}}) &= g_T^{-\alpha_{dec} + \zeta} g_T^{(1)\delta} g_T^{(2)\delta} = g_T^{-\alpha_{dec} + \zeta}.\end{align*}
\]

In the scheme, the size for the secret key \( SK_{i,l} \) is \( (3n^2 + 12\kappa^2 + 3nl + 7n - 3n\mu_l - \mu_l + 14\kappa - 6\kappa + r + 9)|\mathbb{G}| \), and the size of the ciphertext is \( (3n + 6\kappa + 8)|\mathbb{G}_T| + |\mathbb{G}_T| \), where \( n \) denotes the size of predicate vectors, \( \kappa \) is depth of the hierarchy, \( l \) represents the level of hierarchical predicate, \( \mu_l \) is the size of level \( l \)-predicate and \( r \) is the level of time node, \( |\mathbb{G}| \) is the size of group element in \( \mathbb{G} \), and \( |\mathbb{G}_T| \) is the size of group element in \( \mathbb{G}_T \).

**Remark 6.4.** Recently, Okamoto and Takashima \[45\] proposed a PE with short secret keys. We note that their scheme can be easily applied to our system to achieve better efficiency in key size. Moreover, in an updated version \[44\], Okamoto and Takashima devised a payload-hiding HIPE with compact secret keys. The technique \[44\] can also be applied in our system, specifically, for the time period subtree.

### 6.4 Proof of Security

**Theorem 6.1.** Our FS-HPE scheme is adaptively attribute-hiding against chosen plaintext attacks under the DLIN assumption. For any adversary \( A \), there exists a probabilistic polynomial time machine \( D \) such that for any security parameter \( \lambda \),

\[
\text{Adv}_A^{\text{FS-HPE}}(\lambda) \leq (2\nu(\kappa + 1)(n + L + 1)) + 1) \text{Adv}_D^{\text{DLIN}}(\lambda) + \psi
\]
where \( \nu \) is the maximum number of \( A \)'s key queries, \( \kappa \) is the depth of the time tree, and 
\[
\psi = (20\nu(\kappa + 1)(n + L + 1) + 9)/q.
\]

**Outline of the Proof of Theorem 6.1** In the proof, we have semi-functional ciphertexts and 
keys in addition to normal ciphertexts and keys. Semi-functional keys can decrypt all normal 
ciphertexts, but decryption will fail if one attempts to decrypt semi-functional ciphertexts 
with semi-functional keys. Similarly, semi-functional ciphertexts can be decrypted only by 
normal keys. A normal secret key 
\[ SK_{i,l} = (sk_{i,l}, \{sk_{i|_{i-1},l}\}_{i=0}) \]  
associated with time period 
\( i \) and hierarchical predicate \((\overrightarrow{x}_1, ... , \overrightarrow{x}_l)\), a normal ciphertext \( C \) with hierarchical predicate \((\overrightarrow{y}_1, ... , \overrightarrow{y}_h)\) and time period \( i \) are shown in the scheme. A semi-functional secret key and 
a semi-functional ciphertext are expressed in Eqs. (6.9)-(6.11) and Eq. (6.12) respectively. We 
also introduce nominal semi-functional form which is analogue to that in [37]. A nominal 
semi-functional secret key and a nominal semi-functional ciphertext are expressed in Eqs. 
(6.5)-(6.7) and Eq. (6.8) respectively. Normal ciphertexts and keys are used in the real system, 
while their nominal semi-functional or semi-functional counterparts are used in a sequence of 
security games only.

To prove the theorem, we analyze a sequence of games from Game 0 (original game) to 
Game 3. The challenge ciphertext is changed to a semi-functional one in Game 1. When 
at most \( \nu \) delegation queries are issued and let \( \kappa \) be the depth of the time tree, there are 
\( 2\nu(\kappa + 1)(n + L + 1) \) game changes from Game 1 (Game 2-(0, 0, 0)), Game 2-(0, 0, 0'), Game 
2-(0, 0, 1) through Game 2-(\( \nu - 1, \kappa, (n + L) \)) and Game 2-(\( \nu - 1, \kappa, n + L + 1 \)) (Game 2-(\( \nu, 0, 0 \))).

In Game 2-(\( m, k, j \)), the first \( m \times k \times j \) keys are semi-functional and the rest of keys are normal, 
and challenge ciphertext is semi-functional. In Game 2-(\( m, k, j' \)), the first \( m \times k \times j \) keys are 
semi-functional and the \( (m \times k \times j + 1) \)-th key is nominal semi-functional while the remaining 
keys are normal, and challenge ciphertext is nominal semi-functional. In Game 3, all keys and 
challenge ciphertext are semi-functional, where the adversary has zero advantage.

The advantage difference between Games 0 and 1 is equivalent to the advantage of Problem 
1. To prove that, we construct a simulator that uses a Problem 1 instance as an input, and 
interpolates between Game 0 and Game 1. We show that the distribution of the secret keys 
and the challenge ciphertext answered by the simulator is identical to those of Game 0 provided 
\( \beta = 0 \) and those of Game 1 provided \( \beta = 1 \). The hardness of Problem 1 is also based on the 
DLIN assumption.

The advantage difference between Game 2-(\( m, k, j \)) and Game 2-(\( m, k, j' \)), is equivalent to 
the advantage of Problem 2 (i.e., advantage of the DLIN assumption). Here, we introduce 
special forms of nominal semi-functional keys and ciphertext. They are same as their counterparts in semi-functional forms except that there is correlation between coefficients over base 
\( b_2^{(0)}, b_2^{(0)'}, b_{n+1}^{(1)} \) and \( b_{L+1}^{(2)} \). Semi-functional keys and ciphertext are simulated using Problem 
2 instance when \( \beta = 1 \). Due to their algebra structures, semi-functional keys can always de- 
crypt Semi-functional ciphertext when 
\[
J(\overrightarrow{x}_1, ... , \overrightarrow{x}_l)(\overrightarrow{y}_1, ... , \overrightarrow{y}_h) = 1 \text{ and time periods match.}
\]
Therefore, it is hard for the simulator to test for the semi-functional key by creating the 
semi-functional ciphertext by itself. On the other hand, the joint distribution of nominal semi- 
functional key and ciphertext is equivalent to that of semi-functional key and ciphertext when 
either the predicate does not hold or time for queried key is greater than the challenge time.
Hence, both of them appear identical from the adversary’s view, since from the security definition the adversary’s queries should satisfy at least one of the conditions (predicate does not holds and time for queried key is greater than the challenge time).

With the similar argument, we show that the advantage difference between Game 2-\((m, k, j')\) and Game 2-\((m, k, j + 1)\) is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption).

In the final step, we show that Game 2-\((\nu, 0, 0)\) can be conceptually changed to Game 3.

**Definition 6.3** (Problem 1). Problem 1 is to find bit \(\beta\) given (\(\text{param}_\text{P1}, \{\mathcal{B}(k), \hat{\mathcal{B}}^{(k)}\}_{k=0,1,2}, t^{(0)}_\beta, \{t^{(k)}_\beta\}_{k=1,2}, \{t^{(k)}_i\}_{i=2,\ldots,n_k;k=1,2}\)) \(\xleftarrow{\in} \mathcal{G}_\beta^{\text{P1}}(1^\lambda, \vec{n}) = (2; n_1, n_2)\) for \(\beta \xleftarrow{\in} \{0, 1\}\) with probability non-negligibly better than by a random guess, where

\[
\mathcal{G}_\beta^{\text{P1}}(1^\lambda, \vec{n}) = (2; n_1, n_2) : (\text{param}_\text{P1}, \mathcal{B}(0), \mathcal{B}^{*}(0), \mathcal{B}^{*(0)}, \mathcal{B}^{(1)}, \mathcal{B}^{*(1)}, \mathcal{B}^{(2)}, \mathcal{B}^{*(2)}) \xleftarrow{\in} \mathcal{G}_{\text{ob}}(1^\lambda, \vec{n}),
\]

\[
\mathcal{B}^{*(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}),
\]

\[
\hat{\mathcal{B}}^{(1)} = (b_1^{(1)}, \ldots, b_{n_1}^{(1)}, b_{2n_1+1}^{(1)}, \ldots, b_{3n_1+1}^{(1)}),
\]

\[
\hat{\mathcal{B}}^{(2)} = (b_1^{(2)}, b_2^{(2)}, b_{n_2}^{(2)}, b_{2n_2+1}^{(2)}, \ldots, b_{3n_2+1}^{(2)}),
\]

\[
\delta, u, \rho \xleftarrow{\in} \mathbb{F}_q, t_0^{(0)} = (\delta, 0, 0, 0, \rho)_\mathbb{B}(0), t_1^{(0)} = (\delta, u, 0, 0, \rho)_\mathbb{B}(0),
\]

For \(k = 1, 2\):

\[
\rho^{(k)} \xleftarrow{\in} \mathbb{F}_q, \vec{n}^{(k)} \xleftarrow{\in} \mathbb{F}_q^{n_k},
\]

\[
t^{(k)}_0 = (\vec{n}^{(k)}_1, \vec{n}^{(k)}_2, \vec{n}^{(k)}_3, \vec{n}^{(k)}_4, \vec{n}^{(k)}_5),
\]

\[
t^{(k)}_1 = (\vec{n}^{(k)}_1, \vec{n}^{(k)}_2, \vec{n}^{(k)}_3, \vec{n}^{(k)}_4, \vec{n}^{(k)}_5),
\]

\[
\text{For } i = 2, \ldots, n_k: t_i^{(k)} = \delta b_i^{(k)},
\]

return \(\text{param}_\text{P1}, \{\mathcal{B}(k), \hat{\mathcal{B}}^{(k)}\}_{k=0,1,2}, t^{(0)}_\beta, \{t^{(k)}_\beta\}_{k=1,2}, \{t^{(k)}_i\}_{i=2,\ldots,n_k;k=1,2}\).

The corresponding advantage of PPT algorithm \(\mathcal{B}\) in solving Problem 1 is defined as follows:

\[
\text{Adv}_\mathcal{B}^{\text{P1}}(\lambda) = \left| \Pr\left[\mathcal{B}(1^\lambda, \infty) \rightarrow 1 \mid \in \mathcal{G}_0^{\text{P1}}(1^\lambda, \vec{n}) \right] - \Pr\left[\mathcal{B}(1^\lambda, \infty) \rightarrow 1 \mid \in \mathcal{G}_\lambda^{\text{P1}}(1^\lambda, \vec{n}) \right] \right|.
\]

Problem 1 is similar as that defined in Chapter 4, except that \(n_2\) in the parameter is a variable instead of a constant 2.

**Lemma 6.1.** For any adversary \(\mathcal{B}\), there exists a probabilistic machine \(\mathcal{D}\), whose running time is essentially the same as that of \(\mathcal{B}\), such that for any security parameter \(\lambda\), \(\text{Adv}_\mathcal{B}^{\text{P1}}(\lambda) \leq \text{Adv}_\mathcal{D}^{\text{DLIN}}(\lambda) + 8/q\).

**Definition 6.4** (Problem 2). Problem 2 is to decide on bit \(\beta \in \{0, 1\}\) given (\(\text{param}_\text{P1}, \mathcal{B}(0), \mathcal{B}^{*}(0), h^{*(0)}, t^{(0)}_\beta, \{\mathcal{B}(k), \hat{\mathcal{B}}^{(k)}\}, \{h^{(k)}_{\beta,i}, t^{(k)}_i\}_{i=1,\ldots,n_k;k=1,2}\)) \(\xleftarrow{\in} \mathcal{G}_\beta^{\text{P2}}(1^\lambda, \vec{n}) = (2; n_1, n_2)\), where

\[
\mathcal{G}_\beta^{\text{P2}}(1^\lambda, \vec{n}) = (2; n_1, n_2) : (\text{param}_\text{P1}, \mathcal{B}(0), \mathcal{B}^{*}(0), \mathcal{B}^{*(0)}, \mathcal{B}^{(1)}, \mathcal{B}^{*(1)}, \mathcal{B}^{(2)}, \mathcal{B}^{*(2)}) \xleftarrow{\in} \mathcal{G}_{\text{ob}}(1^\lambda, \vec{n}),
\]

\[
\mathcal{B}^{*(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}),
\]

\[
\hat{\mathcal{B}}^{(1)} = (b_1^{(1)}, \ldots, b_{n_1}^{(1)}, b_{2n_1+1}^{(1)}, \ldots, b_{3n_1+1}^{(1)}),
\]
6.4. Proof of Security

$$\widehat{\mathbb{G}}^{(2)} = (b^{(2)}_1, \ldots, b^{(2)}_{n_2}, b^{(2)}_{n_2+1}, \ldots, b^{(2)}_{3n_2+1}),$$
$$\omega, \xi, \delta \leftarrow \mathbb{F}_q, \; z, \pi \leftarrow \mathbb{F}_q^*, \; u = z^{-1},$$
$$h_0^{(0)} = (\omega, 0, 0, \xi, 0)_{\mathbb{G}^{*}(0)}, \; h_1^{(0)} = (\omega, z, 0, \xi, 0)_{\mathbb{G}^{*}(0)}, \; t^{(0)} = (\delta, \pi u, 0, 0, 0)_{\mathbb{G}^{(0)}},$$

For $$k = 1, 2$$:

For $$i = 1, \ldots, n_k$$ and $$j = 1, \ldots, n_k$$:

$$\left(\begin{array}{l}
u^{(k)}_{i,j} \end{array}\right) \leftarrow GL(\mathbb{F}_q, n_k), \; \left(\begin{array}{l}z^{(k)}_{i,j} \end{array}\right) = \left(\begin{array}{l}(\nu^{(k)}_{i,j})^{-1} \end{array}\right)^T ;$$

For $$i = 1, \ldots, n_k$$:

$$\bar{\omega}^{(k)}_i \leftarrow \mathbb{F}^{n_k} ,$$

$$h_{0,i}^{(k)} = \left(\begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}1 \end{array}, \begin{array}{c}0 \end{array} \right)_{\mathbb{G}^{(k)}},$$

$$h_{1,i}^{(k)} = \left(\begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}1 \end{array}, \begin{array}{c}0 \end{array} \right)_{\mathbb{G}^{(k)}},$$

$$t_{i}^{(k)} = \left(\begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}n_k \omega^{(k)}_i \end{array}, \begin{array}{c}1 \end{array}, \begin{array}{c}0 \end{array} \right)_{\mathbb{G}^{(k)}};$$

Output $$(\text{param}_B, \widehat{\mathbb{G}}^{(0)}, \mathbb{G}^{*}(0), h_{0,i}^{(0)}, h_{1,i}^{(0)}, \mathbb{G}^{*}(0), h_{1,i}^{(0)}, t_{i}^{(0)} = \left(\begin{array}{l}h_{1,i}^{(k)} \end{array}, \begin{array}{l}h_{0,i}^{(k)} \end{array}, \begin{array}{l}t_{i}^{(k)} \end{array} \right)_{i=1, \ldots, n_k})_{k=1,2}.$$ 

Let $$B$$ be a probabilistic machine, we define the advantage of $$B$$ for Problem 2 as follows:

$$\text{Adv}^{P2}_B(\lambda) = \text{Pr} \left[ B(1^\lambda, \omega) \rightarrow 1 \right] - \text{Pr} \left[ B(1^\lambda, \omega) \rightarrow 1 \right] \approx \text{Pr} \left[ B(1^\lambda, \omega) \rightarrow 1 \right] - \text{Pr} \left[ B(1^\lambda, \omega) \rightarrow 1 \right] .$$

Problem 2 is similar as that defined in Chapter 4, except that $$n_2$$ in the parameter is a variable instead of a constant 2.

**Lemma 6.2.** For any adversary $$\mathcal{B},$$ there exists a probabilistic machine $$\mathcal{D},$$ whose running time is essentially the same as that of $$\mathcal{B},$$ such that for any security parameter $$\lambda,$$ $$\text{Adv}^{P2}_B(\lambda) \leq \text{Adv}^{DLIN}_B(\lambda) + 5/\lambda.$$

**Proof of Lemmas 6.1 and 6.2** In order to reduce the DLIN problem to Problems 1 and 2 from Definitions 6.3 and 6.4 respectively, we further introduce three “basic problems” that will serve in intermediate steps of the reduction:

- Basic Problem 0 in Definition 6.5
- Basic Problem 1 in Definition 6.6
- Basic Problem 2 in Definition 6.7

In order to prove Lemmas 6.1 and 6.2 we use two intermediate lemmas (Lemmas 6.3 and 6.4) that are common lemmas in the proof of Lemmas 6.1 and 6.2.

**Lemma 6.3.** Let $$(q, \mathbb{V}, \mathbb{G}^T, \mathbb{A}, c)$$ be dual pairing vector spaces by direct product of symmetric pairing groups. Using $$\{\phi_{i,j}\},$$ we can efficiently sample a random linear transformation $$W = \sum_{i,j=1}^{N} \mathbb{F}_q r_{i,j} \phi_{i,j}$$ of $$\mathbb{V}$$ with random coefficients $$\{r_{i,j}\}_{i,j=1}^{N} \leftarrow GL(\mathbb{F}_q, \mathbb{G}^T).$$ The matrix $$\{r_{i,j}^*\} = \left(\begin{array}{c}r_{i,j} \end{array} \right)^{-T}$$ defines the adjoint action on $$\mathbb{V}$$ for pairing $$c,$$ i.e., $$c(W(x), (W^{-1})^T(y)) = c(x, y)$$ for any $$x, y \in \mathbb{V},$$ where $$(W^{-1})^T = \sum_{i=1}^{N} r_{i,j} \phi_{i,j}.$$
The proof of Lemma 6.3 can be found in [43].

**Definition 6.5 (Basic Problem 0).** Basic Problem 0 is to decide bit \( \beta \), given \((\text{param}_{BP^0}, \hat{\mathbb{B}}, \mathbb{B}^*, y^*_0, f, bG, aG, \delta aG) \) \( \xleftarrow{\$} G_{\beta}^{BP^0}(1^\lambda) \) for \( \beta \xleftarrow{\$} \{0, 1\} \) with probability non-negligibly better than by a random guess, where

\[
G_{\beta}^{BP^0}(1^\lambda): \quad \text{param}_G = (q, G, G_T, G, e) \xleftarrow{\$} G_{\text{bpg}}(1^\lambda),
\]

\[
\text{param}_U = (q, \forall, G_T, A, e) \xleftarrow{\$} G_{\text{dpvs}}(1^\lambda, 3, \text{param}_G),
\]

\[
\Lambda = (\lambda_{i,j}) \xleftarrow{\$} \text{GL}(3, \mathbb{F}_q), \quad (\mu_{i,j}) = (A^T)^{-1}, \quad b, a \xleftarrow{\$} \mathbb{F}_q^\times,
\]

\[
b_i = b \sum_{j=1}^{3} \lambda_{i,j} a_j, \quad i = 1, 3, \quad \hat{\mathbb{B}} = (b_1, b_3),
\]

\[
b_i^* = a \sum_{j=1}^{3} \mu_{i,j} a_j, \quad i = 1, 2, 3, \quad B^* = (b_1^*, b_2^*, b_3^*),
\]

\[
g_T = e(G, G)^{ab}, \quad \text{param}_{BP^0} = (\text{param}_U, g_T),
\]

\[
\delta, \sigma, \omega \xleftarrow{\$} \mathbb{F}_q, \quad \rho, \tau \xleftarrow{\$} \mathbb{F}_q^\times,
\]

\[
y_0^* = (\delta, 0, \sigma)_{\mathbb{B}^*}, \quad y_1^* = (\delta, \rho, \sigma)_{\mathbb{B}^*}, \quad f = (\omega, \tau, 0)_{\mathbb{B}},
\]

Output \((\text{param}_{BP^0}, \hat{\mathbb{B}}, \mathbb{B}^*, y_0^*, y_1^*, f, bG, aG, \delta aG)\).

Let \( \text{Adv}_{\mathcal{F}}^{BP^0}(\lambda) \) denote the corresponding advantage of a PPT algorithm \( \mathcal{F} \) for the Basic Problem 0.

**Lemma 6.4.** For any adversary \( \mathcal{F} \), there exists a probabilistic machine \( \mathcal{D} \), whose running time is essentially the same as that of \( \mathcal{D} \), such that for any security parameter \( \lambda \), \( \text{Adv}_{\mathcal{F}}^{BP^0}(\lambda) \leq \text{Adv}_{\mathcal{D}}^{\text{DLIN}}(\lambda) + 5/q \).

The proof of Lemma 6.4 can be found in [43].

**Proof of Lemma 6.1** Combining Lemmas 6.3, 6.4, 6.5 and 6.6, we obtain Lemma 6.1.

**Definition 6.6 (Basic Problem 1).** Basic Problem 1 is to decide bit \( \beta \), given \((\text{param}_T, \{\mathbb{B}(k)\}, \hat{\mathbb{B}}^*(k), \{f_{\beta}(0), f_{\beta}(1), f_{\beta}(2), \{f_i(1)\}_{i=2, \ldots, n_1}, \{f_{i}(2)\}_{i=2, \ldots, n_2}\}) \xleftarrow{\$} G_{\beta}^{BP^1}(1^\lambda, \vec{n} = (2; n_1, n_2)) \) for \( \beta \xleftarrow{\$} \{0, 1\} \) with probability non-negligibly better than by a random guess, where

\[
G_{\beta}^{BP^1}(1^\lambda, \vec{n} = (2; n_1, n_2)):
\]

\[
(\text{param}_T, \mathbb{B}(0), \mathbb{B}^*(0), \mathbb{B}^*(1), \mathbb{B}^*(2), \mathbb{B}^*(2)) \xleftarrow{\$} G_{\text{ob}}(1^\lambda, \vec{n}),
\]

\[
\hat{\mathbb{B}}^*(0) = (b_0^*(0), b_3^*(0), b_4^*(0), b_5^*(0)),
\]

\[
\hat{\mathbb{B}}^*(1) = (b_1^*(1), \ldots, b_{n_1+2}^*(1)),
\]

\[
\hat{\mathbb{B}}^*(2) = (b_1^*(2), \ldots, b_{n_2+2}^*(2)),
\]

\[
\omega, \gamma \xleftarrow{\$} \mathbb{F}_q, \quad \tau \xleftarrow{\$} \mathbb{F}_q^\times, \quad f_{0^0} = (\omega, 0, 0, 0, \gamma)_{\mathbb{B}(0)}, \quad f_{1^0} = (\omega, \tau, 0, 0, \gamma)_{\mathbb{B}(0)},
\]

\[
f_{0^1} = (\omega e_1^{(1)}, 0, 0, 0, \gamma)_{\mathbb{B}(1)},
\]

\[
f_{1^1} = (\omega e_1^{(1)}, 0, 0, 0, \gamma)_{\mathbb{B}(1)}.
\]
6.4. Proof of Security

Let $\text{Adv}^B_{\text{BP}}(\lambda)$ denote the advantage of a PPT algorithm $C$ for the Basic Problem 1.

**Lemma 6.5.** For any adversary $C$, there exists a probabilistic machine $F$, whose running time is essentially the same as that of $C$, such that for any security parameter $\lambda$, $\text{Adv}^B_{\text{BP}}(\lambda) \leq \text{Adv}^B_{\text{BP}}(\lambda)$ for $\tilde{\tau} = (2, n_1, n_2)$.

**Proof.** $F$ is given a Basic Problem 0 instance $(\text{param}_{\text{BP}0}, \tilde{B}, \tilde{B}^*, y_2, f, b G, a G)$. Using param$_G = (q, G, G_T, G, e)$ contained in param$_{\text{BP}0}$, $F$ computes:

$$\begin{align*}
\text{param}_{V_0} &= (q, V_0, G_T, A_0, e) \overset{R}{\leftarrow} G_{\text{dpa}}(1^\lambda, 5, \text{param}_G), \\
\text{param}_{V_l} &= (q, V_l, G_T, A_l, e) \overset{R}{\leftarrow} G_{\text{dpa}}(1^\lambda, 3 n_l + 1, \text{param}_G), \quad l = 1, 2, \\
\text{param}_{\tilde{\tau}} &= (\{\text{param}_{V_l}\}_{l=0,1,2}, g_T),
\end{align*}$$

where $g_T$ is contained in param$_{\text{BP}0}$. $F$ generates random linear transformation $W_l$ on $V_l (l = 0, 1, 2)$ given in Lemma 6.3, then sets

$$\begin{align*}
&d_1^{(0)} = W_0(b_1^*, 0, 0), \quad l = 1, 2; \quad d_3^{(0)} = W_0(0, 0, 0, 0, a G), \\
&d_4^{(0)} = W_0(0, 0, 0, a G, 0), \quad d_5^{(0)} = W_0(b_3^*, 0, 0), \\
&d_4^{(0)} = W_0^{-1}(b_1^*, 0, 0), \quad l = 1, 2; \quad d_4^{(0)} = W_0^{-1}(0, 0, 0, 0, b G), \\
&d_4^{(0)} = W_0^{-1}(0, 0, 0, b G, 0), \quad d_5^{(0)} = W_0^{-1}(b_3^*, 0, 0), \\
&g_{\tilde{\tau}}^{(0)} = W_0(g_{\tilde{\tau}}^*, 0, 0),
\end{align*}$$

$$\begin{align*}
&d_1^{(1)} = W_1(b_1^*, 0^{N_1 - 3}), \quad d_1^{(1)} = W_1(b_2^*, 0^{N_1 - 3}), \quad d_1^{(1)} = W_1(b_3^*, 0^{N_1 - 3}), \\
&d_1^{(1)} = W_1(0^n, a G, 0^{N_1 - m - 1}) \text{ where } \\
&\begin{cases}
m = l + 1 \text{ if } l \in \{2, \ldots, n_1\}, \\
m = l \text{ if } l \in \{n_1 + 2, \ldots, N_1 - 1\},
\end{cases} \\
&d_4^{(1)} = W_1^{-1}(b_1^*, 0^{N_1 - 3}), \quad d_4^{(1)} = W_1^{-1}(b_2^*, 0^{N_1 - 3}), \quad d_4^{(1)} = W_1^{-1}(b_3^*, 0^{N_1 - 3}), \\
&d_4^{(1)} = W_1^{-1}(0^n, b G, 0^{N_1 - m - 1}) \text{ where } \\
&\begin{cases}
m = l + 1 \text{ if } l \in \{2, \ldots, n_1\}, \\
m = l \text{ if } l \in \{n_1 + 2, \ldots, N_1 - 1\},
\end{cases}
\end{align*}$$

$$\begin{align*}
&\text{for } i = 2, \ldots, n_1: f_1^{(1)} = \omega b_i^{(1)}, \\
&f_0^{(2)} = (\omega e_1^{(2)}, 0^{n_2}, 0^{n_2}, 0^{n_2}, 1, g_{\tilde{\tau}}^{(0)}), \\
&f_1^{(2)} = (\omega e_1^{(2)}, \omega e_1^{(2)}, 0^{n_2}, 0^{n_2}, 1, g_{\tilde{\tau}}^{(0)}), \\
&\text{for } i = 2, \ldots, n_1: f_2^{(i)} = \omega b_i^{(2)};
\end{align*}$$

Output $(\text{param}_{\tilde{\tau}}^{(1)}, \{B^{(k)}, \tilde{B}^{(k)}\}_{k=0,1,2}, f_1^{(0)}, f_1^{(2)}, f_2^{(2)}, \{f_1^{(1)}\}_{i=2, \ldots, n_1}, \{f_2^{(2)}\}_{i=2, \ldots, n_2})$. 

Let $\text{Adv}^B_{\text{BP}}(\lambda)$ denote the advantage of a PPT algorithm $C$ for the Basic Problem 1.
\[ g_{b_1}^{(1)} = W_1(y_{b_1}, 0^{N_1-3}), \]
\[ g_{l}^{(1)} = W_1(0^{l+1}, acG, 0^{N_l-1-2}), \quad l = 2, \ldots, n_1; \]
\[ d_1^{(2)} = W_2(b_1, 0^{N_2-3}), \quad d_{n_2+1}^{(2)} = W_2(b_{n_2}, 0^{N_2-3}), \quad d_{N_2}^{(2)} = W_2(b_{N_2}, 0^{N_2-3}), \]
\[ d_l^{(2)} = W_2(0^m, acG, 0^{N_2-m-1}) \quad \text{where} \quad m = \{l \in \{2, \ldots, n_2\}, \]
\[ d_l^{(2)} = (W_2^{-1})^T(b_1, 0^{N_2-3}), \quad d_{n_2+1}^{(2)} = (W_2^{-1})^T(b_{n_2}, 0^{N_2-3}), \quad d_{N_2}^{(2)} = (W_2^{-1})^T(b_{N_2}, 0^{N_2-3}), \]
\[ d_l^{(2)} = (W_2^{-1})^T(0^m, acG, 0^{N_2-m-1}) \quad \text{where} \quad m = l \in \{2, \ldots, n_2\}, \]
\[ g_{b_1}^{(2)} = W_2(y_{b_1}, 0^{N_2-3}), \]
\[ g_{l}^{(2)} = W_2(0^m, acG, 0^{N_2-m-1}), \quad l = 2, \ldots, n_2; \]

where \((v, 0^{N_l-3}) = (G', G', G', 0^{N_l-3})\) for any \(v = (G', G', G', 0^{N_l-3}) \in V = G^3\). In this way bases \(D^{(0)} = (d_l^{(0)})_{l=1, \ldots, 5}\) and \(D^{(2)} = (d_l^{(2)})_{l=1, \ldots, 5}\), and \(D^{(3)} = (d_l^{(3)})_{l=1, \ldots, 3n_j+1}\), \(j = 1, 2\) are dual orthonormal bases.

Therefore, from \(\tilde{B} = (b_1, b_1), B^*, bG, \) and \(acG\) the algorithm \(F\) can compute \(D^{(j)}, j = 0, 1, 2; \)
\(\tilde{D}^{(0)} = (d_l^{(0)}, d_l^{(2)}), (d_l^{(2)}), (d_l^{(2)}), (d_l^{(2)}), (d_l^{(2)})\), and \(\tilde{D}^{(j)} = (d_l^{(j)}, \ldots, d_l^{(j)}, d_l^{(j)}, d_l^{(j)}, d_l^{(j)}, d_l^{(j)}), j = 1, 2.\)

Finally, \(F\) hands \((param_T, \{D^{(k)}, \tilde{D}^{(k)}\})_{k=1, 2, 3, 4, \{g_i^{(1)}\}_{i=2, \ldots, n_1}, \{g_i^{(2)}\}_{i=2, \ldots, n_2}}\) over to \(C\) and, if \(C\) outputs its bit \(\beta'\) then \(F\) forwards this bit as its own output.

We observe that:
\[ g_0^{(0)} = (\omega', 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_1^{(1)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_{0,1}^{(1)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_{1,1}^{(1)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_i^{(1)} = 0, \quad i = 2, \ldots, n_1; \]
\[ g_0^{(2)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_1^{(2)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_{0,1}^{(2)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_{1,1}^{(2)} = (\omega, 0, 0, 0, 0, 0, 0, 0, 0, 0), \]
\[ g_i^{(2)} = 0, \quad i = 2, \ldots, n_2; \]

where \(\omega' = \delta, \sigma = \rho, \gamma' = \sigma\) are distributed uniformly in \(\mathbb{F}_q\). Therefore, the distribution of \((param_T, \{D^{(k)}, \tilde{D}^{(k)}\})_{k=1, 2, 3, 4, \{g_i^{(1)}\}_{i=2, \ldots, n_1}, \{g_i^{(2)}\}_{i=2, \ldots, n_2}}\) is exactly the same as in the instance of Basic Problem 1.

**Lemma 6.6.** For any adversary \(B\), there exists a probabilistic machine \(C\), whose running time is essentially the same as that of \(B\), such that for any security parameter \(\lambda\), \(\text{Adv}^B_{B}(\lambda) \leq \)
$\text{Adv}^{\text{BP1}}_C(\lambda) + 3/q$ for $(\vec{\gamma} = (2; n_1, n_2))$.

**Proof.** $C$ is given an instance of the Basic Problem 1, i.e., $(\text{param}_\gamma, \{B^{(k)}, \widehat{B}^{*(k)}\})_{k=0,1,2}, f^{(0)}_\beta, f^{(1)}_{\beta,1}, f^{(2)}_{\beta,1}, \{f^{(1)}_i\}_{i=2,\ldots,n_1}, \{f^{(2)}_i\}_{i=2,\ldots,n_2}$, and computes $r \leftarrow \text{span} < b^{(1)}_{3n_1+1}, r' \leftarrow \text{span} < b^{(2)}_{3n_2+1},$ and sets $t^{(1)}_{\beta,1} = f^{(1)}_{\beta,1} + r$ and $t^{(2)}_{\beta,1} = f^{(2)}_{\beta,1} + r'$.

Then, $C$ chooses $u_0 \leftarrow \vec{\mathbb{P}}_g, (u^{(k)}_{i,j}) \leftarrow GL(F_q, n_k), (z^{(k)}_{i,j}) = ((u^{(k)}_{i,j})^*)^{-1}$ for $i = 1, \ldots, n_k, j = 1, \ldots, n_k$, and $k = 1, 2$, and computes:

$$ d^{(0)}_2 = (0, 0, 0, 0, 0)_{\mathbb{B}^{(0)}}, $$

$$ d^{(k)}_{n_k + i} = (0_k, k_{i_1, \ldots, i_{n_k}}, 0_k, 1)_{\mathbb{B}^{(k)}}, \quad i = 1, \ldots, n_k, \quad k = 1, 2. $$

$C$ then sets dual orthonormal basis vectors

$$ d^{(0)}_2 = (0, u_0^{-1}, 0, 0, 0)_{\mathbb{B}^{(0)}}, $$

$$ d^{(k)}_{n_k + i} = (0_k, z^{(k)}_{i_1, \ldots, i_{n_k}}, 0_k, 1)_{\mathbb{B}^{(k)}}, \quad i = 1, \ldots, n_k, \quad k = 1, 2. $$

Note that $C$ cannot compute $d^{(0)}_2$ and $d^{(k)}_{n_k + i}, i = 1, \ldots, n_k, k = 1, 2$ due to the lack of $b^{(0)}_2$ and $b^{(k)}_{n_k + i}$.

Then, $C$ sets bases $\mathbb{D}^{(0)} = (b^{(0)}_1, b^{(0)}_2, b^{(0)}_3, b^{(0)}_4, b^{(0)}_5), \mathbb{D}^{(1)} = (b^{(1)}_1, b^{(1)}_2, b^{(1)}_3, b^{(1)}_4, b^{(1)}_5), \mathbb{D}^{(2)} = (b^{(2)}_1, b^{(2)}_2, b^{(2)}_3, b^{(2)}_4, b^{(2)}_5), \mathbb{D}^{(k)} = (b^{(k)}_1, \ldots, b^{(k)}_{n_k}), \mathbb{D}^{(1)} = (b^{(1)}_1, \ldots, b^{(1)}_{n_k}), \mathbb{D}^{(2)} = (b^{(2)}_1, \ldots, b^{(2)}_{n_k}), k = 1, 2.$

Finally, $C$ hands $(\text{param}_\gamma, \{\mathbb{D}^{(k)}, \mathbb{D}^{*(k)}\})_{k=0,1,2}, f^{(0)}_\beta, t^{(1)}_{\beta,1}, t^{(2)}_{\beta,1}, \{f^{(1)}_i\}_{i=2,\ldots,n_1}, \{f^{(2)}_i\}_{i=2,\ldots,n_2}$ over to $B$ and, if $B$ outputs its bit $\beta'$ then $C$ forwards this bit as its own output. Note that with respect to $\mathbb{D}^{(k)}, \mathbb{D}^{*(k)}, k = 0, 1, 2$, the above input to $B$ has the same distribution as the instance of the Problem 1 unless following events occur: $u = 0, \vec{\gamma}^{(1)} = \vec{0},$ or $\vec{\gamma}^{(2)} = \vec{0}$. Those events occur with probability $3/q$ when $\beta = 1$.

**Proof of Lemma 6.2** Combining Lemmas 6.3, 6.4, 6.7 and 6.8 we obtain Lemma 6.2

**Definition 6.7 (Basic Problem 2).** Basic Problem 2 is to find bit $\beta$, given $(\text{param}_\gamma, \widehat{B}^{(0)}, \widehat{B}^{*(0)}, y^{*(0)}, f^{(0)}_\beta, \{\widehat{B}^{(k)}, B^{*(k)}\}, \{y^{*(k)}_i, f^{(k)}_i\}_{i=1,\ldots,n_k})_{k=1,2}$, $R \leftarrow \mathcal{G}^{\text{BP2}}_\beta(1^\lambda, \vec{\gamma} = (2; n_1, n_2))$ for $\beta \leftarrow \{0, 1\}$ with probability non-negligibly better than by a random guess, where

$$ \mathcal{G}^{\text{BP2}}_\beta(1^\lambda, \vec{\gamma} = (2; n_1, n_2)) : $$

$$ \left(\text{param}_\gamma, B^{(0)}, \widehat{B}^{*(0)}, B^{(1)}, \widehat{B}^{*(1)}, B^{(2)}, \widehat{B}^{*(2)}\right) \leftarrow \mathcal{G}_\text{ob}(1^\lambda, \vec{\gamma}), $$

$$ \widehat{B}^{(0)} = (b^{(0)}_1, b^{(0)}_2, b^{(0)}_3, b^{(0)}_4, b^{(0)}_5), $$

$$ \widehat{B}^{(1)} = (b^{(1)}_1, \ldots, b^{(1)}_{n_1}, b^{(1)}_{2n_1+1}, \ldots, b^{(1)}_{3n_1+1}), $$

$$ \widehat{B}^{(2)} = (b^{(2)}_1, \ldots, b^{(2)}_{n_2}, b^{(2)}_{2n_2+1}, \ldots, b^{(2)}_{3n_2+1}), $$

$$ \mathcal{B} = (\{y^{*(k)}_i, f^{(k)}_i\}_{i=1,\ldots,n_k})_{k=1,2}. $$
For any adversary $\mathcal{C}$, let $\delta \stackrel{U}{\leftarrow} \mathbb{F}_q$, $\omega, \pi \leftarrow \mathbb{F}_q^\times$, $\mathbf{y}_0^{(0)} = (\omega, 0, 0, \xi, 0)_{\mathbb{B}^0}$, $\mathbf{y}_1^{(0)} = (\omega, 0, \xi, 0)_{\mathbb{B}^0}$, $\mathbf{f}^{(0)} = (\delta, \pi, 0, 0)_{\mathbb{B}^0}$.

For $k = 1, 2$ and $i = 1, \ldots, n_k$:

$$\mathbf{y}_{0,i}^{(k)} = (\omega \mathbf{v}^{(k)}_i, 0, \xi \mathbf{v}^{(k)}_i, 0, 1)_{\mathbb{B}^k},$$

$$\mathbf{y}_{1,i}^{(k)} = (\omega \mathbf{v}^{(k)}_i, \xi \mathbf{v}^{(k)}_i, 0, 0, 1)_{\mathbb{B}^k},$$

$$\mathbf{f}_i^{(k)} = (\delta \mathbf{v}^{(k)}_i, \pi \mathbf{v}^{(k)}_i, 0, 0, 1)_{\mathbb{B}^k},$$

Output $(\text{param-v}^*, \mathbf{y}_0^{*}, \mathbf{y}_1^{*}, \mathbf{f}^{*}, \{\mathbf{y}_{\beta,i}^{*}, \mathbf{f}_i^{*}\}_{i=1,\ldots,n_k}^{k=1,2})$.

Let $\text{Adv}^{B_2^P} (\lambda)$ denote the corresponding advantage of a PPT algorithm $\mathcal{C}$ for the Basic Problem 2.

**Lemma 6.7.** For any adversary $\mathcal{C}$, there exists a probabilistic machine $\mathcal{F}$, whose running time is essentially the same as that of $\mathcal{C}$, such that for any security parameter $\lambda$, $\text{Adv}^{B_2^P} (\lambda) = \text{Adv}^{B_2^P_0} (\lambda)$ for $\overrightarrow{n} = (2; n_1, n_2)$.

**Proof.** $\mathcal{F}$ is given an instance of the Basic Problem 0, i.e. $(\text{param}_{B_2^P_0}, \hat{\mathbf{y}}^*, \mathbf{y}_0^*, \mathbf{y}_1^*, \mathbf{f}, \mathbf{aG}, \mathbf{bG})$. Using $\text{param}_{\mathcal{G}} = (q, \mathbb{G}, \mathbb{G}_T, \mathbb{G}, \mathbb{G}_R, \mathbb{G}_S)$ contained in $\text{param}_{B_2^P_0}$ it computes

$$\text{param-v}_0 = (q, V_0, G_T, A_0, e) \stackrel{R}{\leftarrow} G_{\text{gsas}}(1^\lambda, 5, \text{param}_{\mathcal{G}}),$$

$$\text{param-v}_i = (q, V_i, G_T, A_i, e) \stackrel{R}{\leftarrow} G_{\text{gsas}}(1^\lambda, 3n_l + 1, \text{param}_{\mathcal{G}}), \quad i = 1, 2,$$

$$\text{param-\pi} = \{(\text{param-v}_i)_{l=0,1,2}, g_{\mathcal{R}}\},$$

where $g_{\mathcal{R}}$ is contained in $\text{param}_{B_2^P_0}$. Then, $\mathcal{F}$ generates random linear transformation $\mathbf{W}_l$ on $\mathbb{V}_l (l = 0, 1, 2)$ given in Lemma 6.3 and sets

$$\mathbf{d}_l^{(0)} = W_0(b_l, 0, 0), \quad l = 1, 2; \quad \mathbf{d}_3^{(0)} = W_0(0, 0, 0, 0, bG),$$

$$\mathbf{d}_l^{(0)} = W_0(b_l, 0, 0), \quad l = 1, 2; \quad \mathbf{d}_3^{(0)} = W_0(0, 0, 0, bG, 0),$$

$$\mathbf{d}_l^{(0)} = (W_0^{-1})^T(b_l^*, 0, 0), \quad l = 1, 2; \quad \mathbf{d}_3^{(0)} = (W_0^{-1})^T(0, 0, 0, 0, aG),$$

$$\mathbf{d}_3^{(0)} = (W_0^{-1})^T(b_l^*, 0, 0), \quad \mathbf{d}_3^{(0)} = (W_0^{-1})^T(0, 0, 0, aG, 0),$$

$$\mathbf{y}_{\beta,l}^{(0)} = (W_0^{-1})^T(b_l^{\beta}, 0, 0), \quad \mathbf{g}^{(0)} = W_0(f, 0, 0).$$

For $k = 1, 2$:

For $l = 1, 2, 3$ and $i = 1, \ldots, n_k$:

$$\mathbf{d}_{l(i-1)n_k+i}^{(k)} = W_k(0^3(i-1), b_l, 0^3(n_k+i), 0),$$

$$\mathbf{d}_{3(n_k+1)}^{(k)} = W_k(0^3 n_k, bG),$$

For $l = 1, 2, 3$ and $i = 1, \ldots, n_k$:

$$\mathbf{d}_{l(i-1)n_k+i}^{(k)} = (W_k^{-1})^T(0^3(i-1), b_l^*, 0^3(n_k+i), 0),$$

$$\mathbf{d}_{3(n_k+1)}^{(k)} = (W_k^{-1})^T(0^3 n_k, aG),$$

For $i = 1, \ldots, n_k$:
6.4. Proof of Security

\( p_{3,i}^*(k) = (W_k^{-1})^T(0^{3(i-1)}, y_{3,i}^*, 0^{3(n_k-i)}, 0), \)
\( g_i^*(k) = W_1(0^{3(i-1)}, f, 0^{3(n_k-i)}), 0). \)

Observe that \( \mathbb{D}^{(0)} = (d_1^{(0)})_{l=1,...,5} \) and \( \mathbb{D}^{*(0)} = (d_i^{*(0)})_{l=1,...,5} \). Therefore, \( \mathbb{D}^{(j)} = (d_1^{(j)})_{l=1,...,3n_j+1} \) and \( \mathbb{D}^{*(j)} = (d_i^{*(j)})_{l=1,...,3n_j+1} \), \( j = 1, 2 \) are dual orthonormal bases.

Therefore, \( \mathcal{F} \) can use \( \tilde{\mathbb{B}} = (b_1, b_2), \mathbb{B}^*, bG, \) and \( aG \) to compute bases \( \mathbb{D}^{*(j)}, j = 0, 1, 2; \)
\( \mathbb{D}^{(j)} = (d_i^{(j)}, d_3^{(j)}, d_4^{(j)}, d_5^{(j)}), \) and \( \tilde{\mathbb{B}} = (d_1^{(j)}, \ldots, d_{n_j}^{(j)}, d_{2n_j+1}, \ldots, d_{3n_j+1}), \) \( j = 1, 2. \)

Finally, \( \mathcal{F} \) hands \( \{\text{param}_\pi, \mathbb{B}^{(0)}, \mathbb{D}^{*(0)}, p_{3,i}^{*(0)}, g^{(0)}, \{\tilde{\mathbb{B}}(k), \mathbb{D}^{*(k)}, \{p_{3,i}^{*(k)}, g_i^{(k)}\}_{i=1,...,n_k}\}_{k=1,2} \) over to \( \mathcal{C} \) and, if \( \mathcal{C} \) outputs a bit \( \beta' \), forwards this bit as its own output.

We observe that:

\[
\begin{align*}
 p_0^{* (0)} &= (\omega, 0, 0, \xi, 0)_{\mathbb{D}^{*(0)}}, \quad p_1^{* (0)} = (\omega, z, 0, \xi, 0)_{\mathbb{D}^{*(0)}}, \quad g^{(0)} = (\delta, \pi, 0, 0, 0)_{\mathbb{D}^{(0)}},
\end{align*}
\]

For \( k = 1, 2 \) and \( i = 1, \ldots, n_k \),

\[
\begin{align*}
 p_{0,i}^{* (k)} &= (\omega^{(k)}_{i,1}, \ldots, \omega^{(k)}_{i,n_k}, 0^{n_k}, \xi^{(k)}_{i,1}, \ldots, \xi^{(k)}_{i,n_k}, 1)_{\mathbb{D}^{*(k)}},
 p_{1,i}^{* (k)} &= (\omega^{(k)}_{i,1}, \ldots, \omega^{(k)}_{i,n_k}, 0^{n_k}, \xi^{(k)}_{i,1}, \ldots, \xi^{(k)}_{i,n_k}, 1)_{\mathbb{D}^{*(k)}},
 g_i^{(k)} &= (\delta e^{(k)}_i, 0^{n_k}, \pi e^{(k)}_i, 0^{n_k}, 1)_{\mathbb{D}^{(k)}}.
\end{align*}
\]

Therefore, the distribution of \( \{\text{param}_\pi, \mathbb{B}^{(0)}, \mathbb{D}^{*(0)}, p_{3,i}^{*(0)}, g^{(0)}, \{\tilde{\mathbb{B}}(k), \mathbb{D}^{*(k)}, \{p_{3,i}^{*(k)}, g_i^{(k)}\}_{i=1,...,n_k}\}_{k=1,2} \) is exactly the same as in the instance of the Basic Problem 2.

**Lemma 6.8.** For any adversary \( \mathcal{B} \), there exists a probabilistic machine \( \mathcal{C} \), whose running time is essentially the same as that of \( \mathcal{B} \), such that for any security parameter \( \lambda \), \( \text{Adv}_{\mathcal{B}}^{\mathcal{C}}(\lambda) = \text{Adv}_{\mathcal{C}}^{\mathcal{B}}(\lambda) \).

**Proof.** Given an instance of the Basic Problem 2, i.e., \( \{\text{param}_\pi, \mathbb{B}^{(0)}, \mathbb{D}^{*(0)}, y_{3,i}^{*(0)}, f^{(0)}, \{\tilde{\mathbb{B}}(k), \mathbb{D}^{*(k)}, \{y_{3,i}^{*(k)}, f_i^{(k)}\}_{k=1,...,n_k}\}_{k=1,2} \) algorithm \( \mathcal{C} \) computes \( r_i^{* (k)} \leftarrow \text{span} < b_{n_{k+1}}^{* (k)}, \ldots, b_{n_k}^{* (k)} > \) and sets \( h_{3,i}^{* (k)} = y_{3,i}^{* (k)} + r_i^{* (k)}, k = 1, 2. \)

Then, \( \mathcal{C} \) chooses \( z_j^{* (k)} \leftarrow \mathbb{R}^{* (k)}, (z_{i,j}^{* (k)}) \leftarrow \text{GL}(\mathbb{F}_q, n_k), i = 1, \ldots, n_k, j = 1, \ldots, n_k, k = 1, 2, \) and computes:

\[
\begin{align*}
 d_2^{(0)} &= (0, z_0, 0, 0)_{\mathbb{B}^{(0)}},
 d_{n_k+1}^{(k)} &= (0^{n_k}, z_1^{(k)}, \ldots, z_{i,1}^{(k)}, 0^{n_k}, 1)_{\mathbb{B}^{(k)}}, \quad i = 1, \ldots, n_k, k = 1, 2,
 d_{2^{(0)}} &= (0, z_0, 0, 0)_{\mathbb{B}^{*(0)}},
 d_{n_k+1}^{*(k)} &= (0^{n_k}, z_{i,1}^{(k)}, \ldots, z_{i,1}^{(k)}, 0^{n_k}, 1)_{\mathbb{B}^{*(k)}}, \quad i = 1, \ldots, n_k, k = 1, 2,
\end{align*}
\]

Then, \( \mathcal{C} \) sets \( z_0 = z^{-1} z_j^{*}, u_0 = z_0^{-1} (z_{i,j}^{* (k)}) = z^{-1} (z_{i,j}^{(k)}) \), and \( u_{i,j}^{(k)} = ((z_{i,j}^{(k)})^{-1} T, z_0 \) is defined as in the Basic Problem 2. This leads to

\[
\begin{align*}
 d_2^{*(0)} &= (0, z_0, 0, 0)_{\mathbb{B}^{*(0)}},
 d_{n_k+1}^{*(k)} &= (0^{n_k}, z_{i,1}^{(k)}, \ldots, z_{i,1}^{(k)}, 0^{n_k}, 1)_{\mathbb{B}^{*(k)}}, \quad i = 1, \ldots, n_k, k = 1, 2,
\end{align*}
\]
\[ d_2^{(0)} = (0, z^{-1} u_0, 0, 0)_{\mathcal{B}(0)}, \]
\[ d_{n_k+i}^{(k)} = (0^{n_k}, z^{-1} u_{n_k+1}, \ldots, z^{-1} u_{n_k+i}, 0, 0)_{\mathcal{B}(k)}, \quad i = 1, \ldots, n_k, \quad k = 1, 2. \]

\( \mathcal{C} \) then computes \( D^{(0)} = (b_1^{(0)}, d_2^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}), \hat{D}^{(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}), \mathcal{D}^{(0)} = (b_1^{(0)}, b_3^{(0)}, b_4^{(0)}, b_5^{(0)}), \mathcal{D}(k) = (b_1^{(k)}, b_3^{(k)}, b_4^{(k)}, b_5^{(k)}) \), \( k = 1, 2. \)

Finally, \( \mathcal{C} \) hands (param, \( \hat{D}^{(0)} \), \( \mathcal{D}^{(0)} \), \( \mathcal{D}^{(k)} \), \( \{ \beta_{\beta,i}^{(k)}, \beta_i^{(k)} \}_{i=1, \ldots, n_k} \{ k = 1, 2 \} \) over to \( \mathcal{B} \) and outputs \( \beta' \in \{ 0, 1 \} \) if \( \mathcal{B} \) outputs \( \beta' \).

For \( \pi \) in Basic Problem 2, let \( \pi' = z \pi \). Then, with respect to \( \pi', D(k), \mathcal{D}(k), k = 0, 1, 2 \), the above answer to \( \mathcal{B} \) has the same distribution as in the instance of Problem 2.

Next, we will prove our scheme FS-HPE using a sequence of games under Problem 1 in Definition 6.3 and Problem 2 in Definition 6.4.

**Lemma 6.9.** For \( p \in \mathbb{F}_q \), let \( C_p = \{ (\mathbf{z}, \mathbf{v}) \mathcal{A} \mathbf{v} = p \} \subset V \times V^* \) where \( V \) is \( n \)-dimensional vector space \( \mathbb{F}_q^n \), and \( V^* \) its dual. For all \( (\mathbf{z}, \mathbf{v}) \in C_p \), for all \( (\mathbf{z}, \mathbf{v}) \in \mathbb{P}_p \), \( \mathbf{P}[\mathbf{v} \mathcal{A} \mathbf{v} = \mathbf{w}] = \mathbf{P}[\mathbf{v} \mathcal{A} \mathbf{v} = \mathbf{w}] = 1/|C_p| \), where \( Z \mathcal{A} GL(n, \mathbb{F}_q) \), \( U = (Z^{-1})^T \), and \( \mathbf{z} \mathcal{A} \mathbf{p} \) denotes the number of elements in \( C_p \).

The proof of Lemma 6.9 was given in [43]. Lemma 6.9 will be used in the proof of Lemma 6.11 and Lemma 6.12 shown later.

**Proof of Theorem 6.1.** To prove Theorem 6.1. We consider the following games:

**Game 0.** Let Game 0 denote the real security game defined in Definition 6.2.

**Game 1.** Game 1 is almost identical to Game 0, except that the ciphertext for challenge attribute vectors \( Y^{(0)} = (\mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_{\mathcal{H}(0)}^{(0)}) \) and \( Y^{(1)} = (\mathbf{y}_1^{(1)}, \ldots, \mathbf{y}_{\mathcal{H}(1)}^{(1)}) \), challenge plaintexts \( (M^{(0)}, M^{(1)}) \) and a time period \( t \) is

\[ e^{(0)} = (\delta, w, \zeta, 0, \varphi)_{\mathcal{B}(0)}, \]
\[ e^{(1)} = (\delta(\mathbf{y}_1^{(0)}), \ldots, \mathbf{y}_{\mathcal{H}(0)+1}^{(0)}, \ldots, \mathbf{y}_d^{(0)}, 0^\varphi, \varphi^{(1)})_{\mathcal{B}(1)}, \]
\[ e^{(2)} = (\delta((1, -i_1), \ldots, (1, -i_k)), \mathbf{w}^T, 0^L, \varphi^{(2)})_{\mathcal{B}(2)}, \]
\[ e^{(M)} = g^e_\mathcal{C} M^{(0)}. \]

where \( \delta, w, \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)} \mathcal{U} \mathbb{F}_q \), \( b \mathcal{U} \{ 0, 1 \} \), \( \mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_{\mathcal{H}(0)+1}, \ldots, \mathbf{y}_d^{(0)} \) is \( \{ (y_1^{(0)}, \ldots, y_{\mathcal{H}(0)+1}, \ldots, y_d^{(0)}), (y_1^{(0)}+1, \ldots, y_{\mathcal{H}(0)+1}+1, \ldots, y_n) \) and \( \mathcal{Y}_{\mathcal{H}(0)+1}, \ldots, \mathbf{y}_d^{(0)} \) is \( \{ y_n^{(0)}, \ldots, y_n \} \). Game 2-(0, 0, 0) is Game 1. The number of keys in \( \mathcal{S}_{k,t} = (sk_{k,t}, \{ sk_{k-1,1,t} \}_{i=0}) \) which is a reply to the \( m \)-th Delegation query in the game, is less than or equal to \( \kappa + 1 \), where \( \kappa \) is the depth of the
6.4. Proof of Security

Game 2-(\(m, n + L + 1\)) is Game 2-(\(m + 1, 0, 0\)). Game 2-(\(m, k, j\)) is almost identical to Game 2-(\(m, k, j\)), except that the reply to the \((j + 1)\)-th key in the \(k\)-th \(sk_{w,l}\) in \(SK_{i,t}\) and in \(m\)-th key query and ciphertext are: When \(j = 0\)

\[
k_{w,l,dec}^{(0)\text{norm-semi}} = (\alpha_{dec}, 1, \eta_{dec}^{(0)}, 0)_{\mathbb{G}^{*}(0)},
\]

\[
k_{w,l,dec}^{(1)\text{norm-semi}} = (\sigma_{dec}^{(1)}, \beta_{dec}^{(1)}, \eta_{dec}^{(1)}, 0^\alpha_{dec}, 0^\mu_{dec}, 0, 0)_{\mathbb{G}^{*}(1)},
\]

\[
k_{w,l,dec}^{(2)\text{norm-semi}} = (\sigma_{dec}^{(2)}, \beta_{dec}^{(2)}, \eta_{dec}^{(2)}, 0^\alpha_{dec}, 0^\mu_{dec}, 0, 0)_{\mathbb{G}^{*}(2)},
\]

(6.5)

where \(\sigma_{dec}, \beta_{dec}, \eta_{dec}(i = 1, \ldots, r) \leftarrow \mathbb{F}_q, Z^{(1)} \leftarrow GL(\mathbb{F}_q, n), Z^{(2)} \leftarrow GL(\mathbb{F}_q, L), \) and all the other variables are generated as in Game 2-(\(m, k, j\)).

When \(j > 0\), we have

\[
k_{(m,k,j+1)}^{\text{norm-semi}} = k_{(m,k,j+1)}^{\text{norm}} + (0^n, (\sigma_{j+1,1}, \ldots, \sigma_{j+1,t}, 0^\alpha_{dec}, 0^\mu_{dec}, 0, 0)_{\mathbb{G}^{*}(1)}),
\]

(6.6)

for randomness and delegate components of the hierarchical predicate, or

\[
k_{(m,k,j+1)}^{\text{norm-semi}} = k_{(m,k,j+1)}^{\text{norm}} + (0^n, (\sigma_{j+1,1}, \ldots, \sigma_{j+1,t}, 0^\alpha_{dec}, 0^\mu_{dec}, 0, 0)_{\mathbb{G}^{*}(2)}),
\]

(6.7)

for randomness and delegate components of the time period. where \(k_{(m,k,j+1)}^{\text{norm}}\) is a correctly generated value of the reply to the \(j\)-th key in the \(k\)-th \(sk_{w,l}\) in \(SK_{i,t}\) and in \(m\)-th key query.

The ciphertext is

\[
\begin{align*}
&\epsilon^{(0)} = (\delta, \psi, \zeta, 0, \varphi)_{\mathbb{G}(0)}, \\
&\epsilon^{(1)} = (\delta, \gamma_{1}^{(b)}, \ldots, \gamma_{b}^{(b)}, y_{b}^{(b)}_{H^{(b)}}, y_{b+1}^{(b)}_{H^{(b)}}, \ldots, y_{d}^{(b)}), \\
&\quad (\gamma_{1}^{(b)}, \ldots, \gamma_{b}^{(b)}, y_{b}^{(b)}_{H^{(b)}}, y_{b+1}^{(b)}_{H^{(b)}}, \ldots, y_{d}^{(b)}), U^{(1)}, 0^n, \psi^{(1)})_{\mathbb{G}(1)}, \\
&\epsilon^{(2)} = (\delta (1, -i_1, \ldots, 1, -i_{\kappa}), (1, -i_1, \ldots, 1, -i_{\kappa}), U^{(2)}, 0^L, \psi^{(2)})_{\mathbb{G}(3)}, \\
&\epsilon^{(M)} = y_{S}^{(b)} M^{(b)}.
\end{align*}
\]

(6.8)

where \(U^{(1)} = (Z^{(1)} - 1)^T\) and \(U^{(2)} = (Z^{(2)} - 1)^T\), and all the other variables are generated as in Game 2-(\(m, k, j\)).

Game 2-(\(m, k, j + 1\)) \((m = 0, \ldots, \nu - 1; k = 0, \ldots, \kappa; j = 0, \ldots, n + L\)): The number of keys in \(SK_{i,t} = (sk_{i,t}, (sk_{k_j,i+1}, j = 0, \ldots, n)\) which is a reply to the \(m\)-th Delegation query in the game, is less than or equal to \(\kappa + 1\), where \(\kappa\) is the depth of the time tree. The number of keys in \(sk_{w,l} = (k_{w,l}, k_{w,l,ran,1}, \ldots, k_{w,l,ran,\kappa}, k_{w,l,ran,\kappa} + 1, \ldots, k_{w,l,ran,\kappa+1}, \ldots, k_{w,l,ran,\nu}, k_{w,l,del,n}, k_{w,l,del,(2(r+1)+1)}, \ldots, k_{w,l,del,(2(r+1)+1)})\) which is the \(k\)-th \(sk_{w,l}\) in \(SK_{i,t}\), is less than
or equal to \( n + L + 1 \), where \( k_{w,l} = k_{w,l,dec}^{(0)} k_{w,l,dec}^{(1)} k_{w,l,dec}^{(2)} \). Game 2-(\( m, k, j + 1 \)) is almost identical to Game 2-(\( m, k, j' \)), except that the reply to the \((j + 1)\)-th key in the \( k \)-th sk \( w,l \) in \( SK_{i,l} \) and in \( m \)-th key query and ciphertext are: When \( j = 0 \)

\[
\begin{align*}
& k_{w,l,dec}^{(0)} = (-\alpha_{dec, c, 1, \gamma_{dec, 0}}^{(0)}), \\
& k_{w,l,dec}^{(1)} = ((\alpha_{dec, c, 1, \gamma_{dec, 0}}^{(1)} + \beta_{dec, r}^{(1)} T^{1}, \ldots, \beta_{dec, r}^{(1)} T^{r}, 0^{n-\mu_1}), \overrightarrow{v}_1, \overrightarrow{y}_{dec}^{(0)}, (0)^{\gamma_{dec, 0}}), \\
& k_{w,l,dec}^{(2)} = ((\alpha_{dec, c, 1, \gamma_{dec, 0}}^{(2)} + \beta_{dec, r}^{(2)} T^{1}, \ldots, \beta_{dec, r}^{(2)} T^{r}, 0^{L-2r}), \overrightarrow{v}_2, \overrightarrow{y}_{dec}^{(2)}), \end{align*}
\]

(6.9)

where \( \overrightarrow{v}_1 \leftarrow F_q^n, \overrightarrow{v}_2 \leftarrow F_q^n \), \( Z^{(1)}(1, \varphi_{dec, 0}) \leftarrow GL(F_q, n), Z^{(2)}(1, \varphi_{dec, 0}) \leftarrow GL(F_q, L) \), and all the other variables are generated as in Game 2-(\( m + 1, k, j - 1 \)).

When \( j > 0 \), we have

\[
k_{semi}^{(m,k,j+1)} = k_{semi}^{(m,k,j+1)} + (0^n, \overrightarrow{v}_{j+1}, 0^n, 0)_{B^{(1)}}
\]

(6.10)

for randomness and delegate components of the hierarchical predicate, or

\[
k_{semi}^{(m,k,j+1)} = k_{semi}^{(m,k,j+1)} + (0^L, \overrightarrow{v}_{j+1}, 0^L, 0)_{B^{(2)}}
\]

(6.11)

for randomness and delegate components of the time period. where \( k_{semi}^{(m+1,k,j+1)} \) is a correctly generated value of the reply to the \( j \)-th key in the \( k \)-th sk \( w,l \) in \( SK_{i,l} \) and in \( m \)-th key query. \( \overrightarrow{v}_{j+1} \leftarrow F_q^n, k = 1, 2 \).

The ciphertext is

\[
\begin{align*}
& c^{(0)} = (\delta, w, \zeta, 0, \varphi)_{B^{(0)}}, \\
& c^{(1)} = (\delta(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_h^{(b)}) + \gamma, \overrightarrow{w}_1, 0^n, \varphi^{(1)}), \\
& c^{(2)} = (\delta(1, -1), \ldots, (1, -1)), \overrightarrow{w}_2, 0^L, \varphi^{(2)}), \end{align*}
\]

(6.12)

\[
c^{(M)} = g_{\varphi}^c M^{(b)}.
\]

where \( \overrightarrow{w}_1 \leftarrow F_q^n \setminus \{0\}, \overrightarrow{w}_2 \leftarrow F_q^n \setminus \{0\} \), and all the other variables are generated as in Game 2-(\( m, k, j' \)).

**Game 3.** Game 3 is almost identical to Game 2-(\( \nu, 0, 0 \)), except that the ciphertext for challenge attribute vectors \( Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_h^{(0)}) \) and \( Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_h^{(1)}) \), challenge plaintexts \( (M^{(0)}, M^{(1)}) \), and a time period \( I \) is

\[
\begin{align*}
& c^{(0)} = (\delta, w, \zeta', 0, \varphi)_{B^{(0)}}, \\
& c^{(1)} = (\delta(\overrightarrow{y}_1^{(b)}), \ldots, \overrightarrow{y}_h^{(b)}, \overrightarrow{w}_1, 0^n, \varphi^{(1)}), \\
& c^{(2)} = (\delta(1, -1), \ldots, (1, -1)), \overrightarrow{w}_2, 0^L, \varphi^{(2)}), \\
& c^{(M)} = g_{\varphi}^{c'} M^{(b)}.
\end{align*}
\]

(6.13)

where \( \zeta' \leftarrow F_q, (\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_h^{(b)}) \leftarrow F_q^n \), and all the other variables are generated as in Game
2-ν. We note that \( \zeta \) and \((\tilde{y}_1', \ldots, \tilde{y}_d')\) are chosen uniformly and independently from \( \zeta \) and \((Y^{(0)}, Y^{(1)})\), respectively.

Let \( \text{Adv}_{A}^{(0)}(\lambda) \) be \( \text{Adv}_{A}^{\text{FS-HPE}}(\lambda) \) in Game 0, and \( \text{Adv}_{A}^{(1)}(\lambda) \), \( \text{Adv}_{A}^{(2-(m,k,j))}(\lambda) \), \( \text{Adv}_{A}^{(2-(m,k,j'))}(\lambda) \), \( \text{Adv}_{A}^{(3)}(\lambda) \) be the advantage of \( A \) in Game 1, \( 2-(m, k, j) \), \( 2-(m, k, j') \), 3, respectively. It is clear that \( \text{Adv}_{A}^{(3)}(\lambda) = 0 \) by Lemma 6.14. We will show Lemmas [6.10][6.13] which evaluate the gaps between pairs of \( \text{Adv}_{A}^{(0)}(\lambda), \text{Adv}_{A}^{(1)}(\lambda), \text{Adv}_{A}^{(2-(m,k,j))}(\lambda), \text{Adv}_{A}^{(2-(m,k,j')+1)}(\lambda), \) for \( m = 0, \ldots, \nu - 1; k = 0, \ldots, \kappa; j = 0, \ldots, n + L \) and \( \text{Adv}_{A}^{(3)}(\lambda) \).

From these lemmas, we obtain

\[
\text{Adv}_{A}^{\text{FS-HPE}}(\lambda) = \text{Adv}_{A}^{(0)}(\lambda) \\
\leq |\text{Adv}_{A}^{(0)}(\lambda) - \text{Adv}_{A}^{(1)}(\lambda)| \\
\quad + \sum_{m=0}^{\nu-1} \sum_{k=0}^{n} \sum_{j=0}^{n+L} |\text{Adv}_{A}^{(2-(m,k,j))}(\lambda) - \text{Adv}_{A}^{(2-(m,k,j'))}(\lambda)| \\
\quad + \sum_{m=0}^{\nu-1} \sum_{k=0}^{n} \sum_{j=0}^{n+L} |\text{Adv}_{A}^{(2-(m,k,j'))}(\lambda) - \text{Adv}_{A}^{(2-(m,k,j'+1))}(\lambda)| \\
\quad + |\text{Adv}_{A}^{(2-(\nu,0,0))}(\lambda) - \text{Adv}_{A}^{(3)}(\lambda)| + |\text{Adv}_{A}^{(3)}(\lambda)| \\
\leq \text{Adv}_{B_1}^{P_1}(\lambda) + \sum_{m=0}^{\nu-1} \sum_{k=0}^{n} \sum_{j=0}^{n+L} \text{Adv}_{B_{2m,kj}}^{P_2}(\lambda) \\
\quad + \sum_{m=0}^{\nu-1} \sum_{k=0}^{n} \sum_{j=0}^{n+L} \text{Adv}_{B_{2m(k+1)}}^{P_2}(\lambda) + (10\nu(\kappa + 1)(n + L + 1) + 1)/q \\
\leq (2\nu(\kappa + 1)(n + L + 1) + 1)\text{Adv}_{D}^{\text{LIN}}(\lambda) + (20\nu(\kappa + 1)(n + L + 1) + 9)/q.
\]

This completes the proof of Theorem 6.1.

**Lemma 6.10.** For any adversary \( A \), there exists a probabilistic machine \( B_1 \), whose running time is essentially the same as that of \( A \), such that for any security parameter \( \lambda \), \( |\text{Adv}_{A}^{(0)}(\lambda) - \text{Adv}_{A}^{(1)}(\lambda)| \leq \text{Adv}_{B_1}^{P_1}(\lambda) \).

**Proof.** Suppose a polynomial time adversary \( A \) can successfully distinguish between Game 0 and Game 1. We construct a simulator \( B_1 \) that leverages \( A \) as a black box to solve Problem 1. The procedure is shown as follows:

1. \( B_1 \) is given an instance of Problem 1, i.e. \((\text{param}_{\mathcal{P}}, \{\mathcal{P}(k), \mathcal{P}^{*}(k)\}_{k=0,1,2}, t^{(0)}_j, t^{(k)}_j)_{k=1,2}, \{t^{(k)}_{i,j}\}_{i=2,\ldots,n,k=1,2}\) where \( n_1 = n \) and \( n_2 = L \), and plays the role of the challenger in the security game against adversary \( A \).

2. At the beginning of the game, \( B_1 \) gives \( A \) the public key \( PK = (1^\lambda, \text{param}_{\mathcal{P}}, (b^{(0)}_1, b^{(0)}_3, b^{(0)}_5, b^{(0)}_7, \ldots, b^{(1)}_1, b^{(1)}_{n+1}, b^{(2)}_1, b^{(2)}_{2n+1}, b^{(1)}_3, b^{(2)}_3, \ldots, b^{(1)}_9, b^{(2)}_9, b^{(1)}_{10}, b^{(2)}_{10}, \ldots, b^{(9)}_{10}, b^{(9)}_{10}, b^{(9)}_{10}) \), which is obtained from the Problem 1 instance.

3. When a delegation query is issued, \( B_1 \) computes a normal secret key using Delegate, Update and \( SK_{0,1} \), which is computed from \((\mathcal{P}^{*(0)}, \mathcal{P}^{*(1)}, \mathcal{P}^{*(2)})\).
Claim 6.1. For $0 \leq i \leq n$, first recall that 

$$ \text{Proof.} \quad \text{A computes and returns} $$

$$ C = (c^{(0)}, c^{(1)}, c^{(2)}, c^{(M)}), $$

where $c^{(0)} = t^{(0)}_{\beta} + \zeta y^{(0)}_1 + \cdots + y^{(b)}_1 t^{(b)}_{\beta 1} + \sum_{i=2}^{n} y_i t^{(1)}_i + \sum_{i=\mu_{x(b)} + 1}^{n} y_i t^{(1)}_i$, $c^{(2)} = t^{(2)}_{\beta 1} + (-i_1) t^{(2)}_{L-1} + \cdots + t^{(2)}_{L-1} + (-i_n) t^{(2)}_L$, and $c^{(M)} = g^{\beta}_{T} M^{(b)}$ using $(t^{(0)}_{\beta}, t^{(b)}_{\beta 1})_{k=1,2}, (t^{(2)}_{i})_{i=2,\ldots,n,k=1,2}, b^{(0)}_y$ from the instance of Problem 1 and $(\bar{y}_1^{(b)}, \ldots, \bar{y}_k^{(b)}), M^{(b)}, I$ where $\zeta, y_{\mu_{x(b)} + 1}, \ldots, y_n \leftarrow \mathbb{F}_q, b \leftarrow \{0,1\}$. $i_1, \ldots, i_n$ are parsed from $I$.

5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.

6. $A$ outputs a bit $b'$. If $b = b'$, $B_1$ outputs 1. Otherwise, $B_1$ outputs 0.

Claim 6.1. For $\beta = 0$ the challenge ciphertext $C = (c^{(0)}, c^{(1)}, c^{(2)}, c^{(M)})$ generated in step 4 is distributed exactly as in Game 0, whereas if $\beta = 1$, the challenge ciphertext $C = (c^{(0)}, c^{(1)}, c^{(2)}, c^{(M)})$ generated in step 4 is identically distributed to Game 1.

**Proof.** First recall that $y^{(b)}_1 = 1$. If $\beta = 0$ then the ciphertext given by

$$ c^{(0)} = (\delta, \zeta \cdot 0, \rho)_{B^{(0)}}, $$

$$ c^{(1)} = (\delta \cdot \bar{y}_1, \ldots, \bar{y}_d), 0^{2n}, \rho^{(1)}_{B^{(1)}}, $$

$$ c^{(2)} = (\delta((1, -i_1), \ldots, (1, -i_n)), 0^{2L}, \rho^{(2)}_{B^{(2)}}, $$

$$ c^{(M)} = g^{\beta}_{T} M. $$

is the challenge ciphertext from Game 0. In contrast, if $\beta = 1$ then the following components of the ciphertext have a different form

$$ c^{(0)} = (\delta, u, \zeta \cdot 0, \rho)_{B^{(0)}}, $$

$$ c^{(1)} = (\delta \cdot \bar{y}_1, \ldots, \bar{y}_d), u^{(1)}, 0^{n}, \rho^{(1)}_{B^{(1)}}, $$

$$ c^{(2)} = (\delta((1, -i_1), \ldots, (1, -i_n)), u^{(2)}, 0^{L}, \rho^{(2)}_{B^{(2)}}, $$

$$ c^{(M)} = g^{\beta}_{T} M. $$

the challenge ciphertext from Game 1. □

From the above claim, if $\beta = 0$ then simulated ciphertexts are distributed exactly as in Game 0, whereas for $\beta = 1$ their distribution is identical to Game 1. Therefore, $| \text{Adv}_{A}^{(0)} (\lambda) - \text{Adv}_{A}^{(1)} (\lambda) | = | \Pr [ B_1(1^\lambda, x) \rightarrow 1 | \sigma R g^{\beta}_{T} \gamma_{1} \cdot (1^\lambda, \gamma) ] - \Pr [ B_1(1^\lambda, x) \rightarrow 1 | \sigma R g^{\beta}_{T} \gamma_{1} \cdot (1^\lambda, \gamma) ] | \leq \text{Adv}_{B_1}^{(1)} (\lambda).$ This completes the proof of Lemma 6.10. □

**Lemma 6.11.** For any adversary $A$, there exists a probabilistic machine $B_{2m(k)}$, whose running time is essentially the same as that of $A$, such that for any security parameter $\lambda$, $| \text{Adv}_{A}^{2-(m,k,j)} (\lambda) - \text{Adv}_{A}^{2-(m,k,j')}(\lambda) | \leq \text{Adv}_{B_{2m(k)}}^{P_2} (\lambda) + 5/q.$ □
6.4. Proof of Security

Proof. Suppose a polynomial time adversary \( A \) can successfully distinguish between Game 2-(\( m, k, j \)) and Game 2-(\( m, k, j' \)). We construct a simulator \( \mathcal{B}'_{2mkj} \) that leverages \( A \) as a black box to solve Problem 2. The procedure is shown as follows:

1. \( \mathcal{B}'_{2mkj} \) is given an instance of Problem 2, i.e., \( (\text{param}_{\beta}, \hat{B}(0), \hat{s}(0), h_{\beta}^{(0)}, t(0), (\hat{R}(k), \mathcal{B}^{*}(k)), \{h_{\beta,j}^{(k)}, t_{i}^{(k)}\}_{i=1,\ldots,n_{k}})_{k=1,2} \) where \( n_{1} = n \) and \( n_{2} = L \), and acts as a challenger in the security game against adversary \( A \).

2. At the beginning of the game, \( \mathcal{B}'_{2mkj} \) gives \( A \) the public key \( PK = (1^{l}, \text{param}_{\beta}, (b_{0}^{(0)}, b_{3}^{(0)}), b_{1}^{(0)}, b_{2}^{(0)}; b_{1}^{(1)}, b_{2}^{(1)}, b_{3}^{(1)}, b_{2L+1}^{(1)}, \ldots, b_{3L}^{(1)}, b_{4}^{(0)}), \) which is obtained from the Problem 2 instance.

3. The answer to the \( s \)-th key in the \( k \)-th \( \text{sk}_{w,j} \) in the \( m \)-th \( SK_{i,l} \) query for time period \( i \) and hierarchical predicate vectors \( (\mathcal{F}_{1}, \ldots, \mathcal{F}_{l}) \) is as follows:

   a) For \( 0 \leq s \leq j \) the algorithm \( \mathcal{B}'_{2mkj} \) computes a semi-functional key using \( \{\hat{s}(k)\}_{k=0,1,2} \) of the Problem 2 instance as follows:

   b) For \( s = j + 1 \) it computes as follows:

   when \( j = 0 \), it computes \( (k_{0, \text{dec}}, k_{1, \text{dec}}, k_{2, \text{dec}}) \) using \( \{h_{\beta}^{(0)}, b_{1}^{(0)}, b_{3}^{(0)}, \{h_{\beta,j}^{(i)}, b_{j}^{(i)}\}_{i=1,2,j=1,\ldots,n_{i}} \) of the Problem 2 instance as follows:

   For \( i = 1, 2 \) : \( \theta_{i}, v_{i}, v_{i}', \theta_{i}, \hat{\theta}_{k}(k = 1, \ldots, l), \phi_{k}(k' = 1, \ldots, r) \in \mathbb{F}_{q} \);

   \[ s_{\beta}^{(0)} = \sum_{i=1}^{2}(\theta_{i}h_{\beta}^{(0)} + v_{i}b_{1}^{(0)}), \quad k_{0, \text{dec}}^{(0)} = -s_{\beta}^{(0)} + b_{1}^{(0)}, \]

   For \( i = 1, 2 \) and \( j = 1, \ldots, n_{i} \) :

   \[ s_{\beta,j}^{(i)} = \theta_{j}h_{\beta,j}^{(i)} + v_{j}b_{j}^{(i)}; \quad s_{\beta,j}^{(i)} = \theta_{j}h_{\beta,j}^{(i)} + v_{j}b_{j}^{(i)}; \]

   \[ k_{1, \text{dec}}^{(i)} = \sum_{k=1}^{r} \left( \theta_{k} \sum_{j=\mu_{k}+1}^{2k} x_{j}s_{\beta,j}^{(1)} \right) + s_{\beta,1}^{(1)}, \]

   \[ k_{2, \text{dec}}^{(i)} = \sum_{k=1}^{r} \left( \phi_{k} \sum_{j=2k'-1}^{2k'_{i}} I_{j}s_{\beta,j}^{(2)} \right) + s_{\beta,1}^{(2)}, \]

   when \( j > 0 \), compute randomness and delegation components using \( \{h_{\beta,j}^{(i)}, b_{j}^{(i)}\}_{i=1,2,j=1,\ldots,n_{i}} \) of the Problem 2 instance as follows:

   For \( i = 1, 2 ; \quad v_{i}', \theta_{i}, \psi', \hat{\theta}_{k}(k = 1, \ldots, l), \phi_{k}(k' = 1, \ldots, r) \in \mathbb{F}_{q} \):

   For \( i = 1, 2 \) and \( j = 1, \ldots, n_{i} \) :

   \[ s_{\beta,j}^{(i)} = \theta_{j}h_{\beta,j}^{(i)} + v_{j}b_{j}^{(i)}; \]

   \[ k_{1, \text{ran}, b}^{(i)} = \sum_{k=1}^{r} \left( \theta_{k} \sum_{j=\mu_{k}+1}^{2k} x_{j}s_{\beta,j}^{(1)} \right), \]
Chapter 6. A Forward-Secure Hierarchical Predicate Encryption Scheme

Let

\[ k_{w, l, \text{ran}, b'}^{(2)} = \sum_{k' = 1}^{r} (\phi_{k'} \sum_{j = 2k' - 1}^{2k'} I_j) s_{\beta, j}^{(2)}, \]

\[ k_{w, l, \text{del}, b}^{(1)} = \sum_{k = 1}^{l} (\mu_{k} \sum_{j = \mu_{k - 1} + 1}^{\mu_{k}} x_j s_{\beta, j}^{(1)}) + \psi b_{h}^{(1)}, \]

\[ k_{w, l, \text{del}, b'}^{(2)} = \sum_{k' = 1}^{r} (\phi_{k'} \sum_{j = 2k' - 1}^{2k'} I_j) s_{\beta, j}^{(2)} + \psi' b_{h}^{(2)}, \]

c) For \( s \geq j + 2 \) the algorithm \( B_{2mkj}' \) computes a normal key using \( \{B^{(k)}\}_{k=0,1,2} \) from the instance of Problem 2.

4. When \( B_{2mkj}' \) receives challenge attribute vectors \( Y^{(0)} = (\overrightarrow{y}_{1}^{(0)}, \ldots, \overrightarrow{y}_{h}^{(0)}) \) and \( Y^{(1)} = (\overrightarrow{y}_{1}^{(1)}, \ldots, \overrightarrow{y}_{h}^{(1)}) \), challenge plaintexts \( (M^{(0)}, M^{(1)}) \) and a time period \( I \) from \( A \), \( B_{2mkj}' \) computes and returns the ciphertext \( C = (c_{0}, c_{1}, c_{2}, c_{M}) \) where

\[ c_{0} = t^{(0)} + \zeta b_{h}^{(0)} + \varphi b_{h}^{(0)}, \]
\[ c_{1} = \sum_{i=1}^{\mu_{n}(i)} y_i t_i^{(1)} + \sum_{i=1^{\mu_{n}(i) + 1}}^{n} y_i t_i^{(1)} + \varphi^{(1)} b_{3n+1}^{(1)}, \]
\[ c_{2} = \sum_{j=1}^{L} I_j t_j^{(2)} + \varphi^{(2)} b_{3L+1}^{(2)}, \]
\[ c_{M} = \overrightarrow{y}_{i}^{(2)}, \]

using \( (t^{(0)}, \{t_i^{(1)}\}_{i=1 \ldots n}, \{t_i^{(2)}\}_{i=1 \ldots L}, b_{h}^{(0)}, b_{h}^{(0)}, b_{3n+1}^{(1)}, b_{3L+1}^{(2)}) \) from the instance of Problem 2 and \( (\overrightarrow{y}_{1}^{(0)}, \ldots, \overrightarrow{y}_{h}^{(0)}), M^{(b)}, I \) where \( \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)}, y_{\mu_{n}(i)+1}, \ldots, y_{n} \in \mathbb{F}_{q}, b \in \{0,1\} \). \( I_{2k-1 \ldots 2k} \) denotes the vector for \( i_{k} \), and \( i_{1} \ldots i_{k} \) is parsed from \( I \).

5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.

6. \( A \) outputs a bit \( b' \). If \( b = b' \), \( B_{2mkj}' \) outputs 1. Otherwise, \( B_{2mkj}' \) outputs 0.

**Claim 6.2.** The distribution of the view of adversary \( A \) in the above-mentioned game simulated by \( B_{2mkj}' \) given a Problem 2 instance with \( \beta \in \{0,1\} \) is the same as that in Game 2-(\( m, k, j' \)) (resp. Game 2-(\( m, k, j' \))) if \( \beta = 0 \) (resp. \( \beta = 1 \)) except with probability \( 4/q \) (resp. \( 1/q \)).

**Proof.** It is clear that \( B_{2mkj}' \)’s simulation of the public key generation (step 2) and the answers to the \( s \)-th query where \( s \neq j + 1 \) (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and delegation oracle in Game 2-(\( m, k, j' \)) and Game 2-(\( m, k, j' \)).

Next we analyze the distribution of the \( s \)-th key in the \( k \)-th \( sk_{w, l} \) in the \( m \)-th \( SK_{i, l} \) query for time period \( i \) and hierarchical predicate vectors \( (X_1, \ldots, X_l) \) where \( s = j + 1 \) (case (b) of steps (3) and (5)). In this case values \( s_{\beta}^{(0)}, s_{\beta, j}^{(1)}, s_{\beta, j}^{(2)} \), \( i = 1, 2, j = 1, \ldots, n_{j} \) can be expressed as follows. Let \( \beta^{(i)} = \theta_{i} \omega + v_{i}', \alpha^{(i)} = \theta_{i} \omega + v_{i}, \alpha = \alpha^{(1)} + \alpha^{(2)}, \gamma = \varphi_{1} + \varphi_{2}, \epsilon = \gamma z \). Then,

\[ s_{0}^{(0)} = (\alpha, 0, 0, \gamma \xi, 0)_{B^{(0)}}, \]
\[ s_{1}^{(0)} = (\alpha, \epsilon, 0, \gamma \xi, 0)_{B^{(0)}}. \]
where \( z_j^{(i)} = z_j^{(i)}; \ldots, z_j^{(i)} \), \( \omega, \zeta, \{ z_j^{(i)}, \tilde{z}_j^{(i)} \} \), are defined in Problem 2.

If \( \beta = 1 \) in the instance of Problem 2 then the decryption component \((k_{w,1}^{(0)}, k_{w,1}^{(1)}; k_{w,1}^{(2)})\)
has the same distribution as Eq. 6.5 except that \( \epsilon w = \gamma \), where \( \gamma = q_1 + q_2 \) and \( w = w \subseteq \mathbb{F}_q \)
of \( c_0 \) in Eq. 6.1. Randomness and delegation components for the hierarchical predicate and
time period have the same distribution as in Eq. 6.6 and 6.7 respectively.

Next, we show that the joint distribution of the response to \( j + 1 \)-th key in the \( k \)-th \( sk_{w,l} \)
in the \( m \)-th \( SK_{k,l} \) query and of the challenge ciphertext in the simulation by \( \mathcal{B}_{2mkj} \)
for the given instance of Problem 2 is equivalent to the distribution in Game 2-(\( m, k, j \)) if \( \beta = 0 \) and
to the distribution in Game 2-(\( m, k, j' \)) if \( \beta = 1 \).

If \( \beta = 0 \) then this equivalence follows easily, unless one of the following conditions holds:
(1) \( \omega \) defined in Problem 2 is zero, (2) \( w = 0 \), (3) \( \bar{w}_1 = \bar{0} \), (4) \( \bar{w}_2 = \bar{0} \). However, those
events occur with probability 4/\( q \).

If \( \beta = 1 \) then \( \mathcal{B}_{2mkj}' \)’s simulation for the key is the same as that expressed in Eq. 6.5, 6.6 and 6.7 and \( \mathcal{B}_{2mkj}' \)’s simulation for the challenge ciphertext is the same as that expressed in
Eq. 6.8 except that \( \epsilon w = \gamma \), where \( \gamma = q_1 + q_2 \) and \( w \subseteq \mathbb{F}_q \) of \( c_0 \) in Eq. 6.1.

Therefore, we will show that \( \gamma \) is uniformly distributed and is independent from the other
variables used in the simulation by \( \mathcal{B}_{2mkj}' \). Since \( \gamma \) is related to \( \overrightarrow{A}_1, \overrightarrow{A}_2, \overrightarrow{B}_1, \) and \( \overrightarrow{B}_2 \), where
\( \overrightarrow{A}_1 = (\vartheta_1 \overrightarrow{e}^{(1)} + \vartheta_1 \overrightarrow{e}^{(1)}, \ldots, \vartheta_{\kappa} \overrightarrow{e}^{(1)}, \vartheta_{\kappa} \overrightarrow{e}^{(1)}, \vartheta_{\kappa} \overrightarrow{e}^{(1)} \cdot Z^{(1)} \), \( \overrightarrow{A}_2 = (\vartheta_2 \overrightarrow{e}^{(2)} + \vartheta_2 \overrightarrow{e}^{(2)}, \ldots, \vartheta_{\kappa} \overrightarrow{e}^{(2)} \cdot Z^{(2)} \), and
\( \overrightarrow{B}_1 = (\gamma^{(b)} \overrightarrow{e}^{(b)}, \ldots, \gamma^{(b)} \overrightarrow{e}^{(b)}, \gamma^{(b)} \overrightarrow{e}^{(b)}, \gamma^{(b)} \overrightarrow{e}^{(b)}, \gamma^{(b)} \overrightarrow{e}^{(b)}, U^{(1)} \), \( \overrightarrow{B}_2 = (\tilde{T}_1^{(1)}, \ldots, \tilde{T}_1^{(1)}, \tilde{T}_1^{(1)} \cdot U^{(2)} \) where \( \tilde{T}_z = (1, -i_z) \)
and \( i_1, \ldots, i_{\kappa} \) is parsed from \( I \). We analyze joint distribution of these variables for the cases that
appear in Definition 6.2.

1. When \( i > I \), i.e., the time period of the queried key is after the time period of encoded
   in the ciphertext, due to Lemma 6.9, the pair \( \overrightarrow{A}_2, \overrightarrow{B}_2 \) is uniformly and independently
   distributed over \( C_{\sum_{z=1}^{\kappa} \theta''_z \cdot (\tilde{T}_z \cdot \tilde{T}_z) + \varrho_2} = \{(\bar{w}, \bar{v}) \cdot \bar{w}, \bar{v} = \sum_{z=1}^{\kappa} \theta''_z \cdot (\tilde{T}_z \cdot \tilde{T}_z) + \varrho_2 \}
   \) (over \( \mathbb{F}_q \)). Since \( \theta''_z \subseteq \mathbb{F}_q \), the pair \( \overrightarrow{A}_2, \overrightarrow{B}_2 \) is thus uniformly and
   independently distributed over \( \mathbb{F}_q^n \).

2. When \( i \leq I \) and \( f(\overrightarrow{x}, \ldots, \overrightarrow{x}) \cdot (\overrightarrow{y}^{(0)} \cdot \ldots, \gamma^{(b)} \gamma^{(b)}) = 0 \), the pair
   \( \overrightarrow{A}_2, \overrightarrow{B}_2 \) is uniformly and independently distributed over \( C_{\sum_{z=1}^{\kappa} \theta''_z \cdot (\vartheta, \overrightarrow{v}) + \varrho_1} \)
   (over \( \mathbb{F}_q \)). Since \( \theta''_z \subseteq \mathbb{F}_q \), the pair \( \overrightarrow{A}_1, \overrightarrow{B}_1 \) is thus uniformly and
   independently distributed over \( \mathbb{F}_q^n \).

Considering the adversary \( \mathcal{A}' \)’s restriction on key queries from Definition 6.2
in above two cases at least one of \( \overrightarrow{A}_1, \overrightarrow{B}_1 \) and \( \overrightarrow{A}_2, \overrightarrow{B}_2 \) is uniformly and independently distributed over \( \mathbb{F}_q^{2n_k} \) for \( k = 1, 2 \). Therefore, \( \gamma = \varrho_1 + \varrho_2 \) is independent from the distribution of \( \varrho_1 \) (resp. \( \varrho_2 \)),
which can be given by \( \overrightarrow{A}_1, \overrightarrow{B}_1 \) (resp. \( \overrightarrow{A}_2, \overrightarrow{B}_2 \)). Thus, \( \gamma \) is uniformly and independently
distributed from the other variables in the simulation of \( \mathcal{B}_{2mkj}' \).
Therefore, the view of $A$ in the game simulated by $B'_{2mkj}$ on input an instance of Problem 2 with $\beta = 1$ is the same as in Game 2-$(m, k, j')$ unless $\omega = 0$ occurs. This event happens with probability $1/q$.

This completes the proof of Lemma 6.11.

Lemma 6.12. For any adversary $A$, there exists a probabilistic machine $B_{2mk(j+1)}$, whose running time is essentially the same as that of $A$, such that for any security parameter $\lambda$,

$$\left| \text{Adv}_A(2^{(m,k,j')}) (\lambda) - \text{Adv}_A(2^{(m,k,j+1)}) (\lambda) \right| \leq \text{Adv}_{B_{2mk(j+1)}} (\lambda) + 5/q.$$

Proof. Suppose a polynomial time adversary $A$ can successfully distinguish between Game 2-$(m, k, j')$ and Game 2-$(m, k, j + 1)$. We construct a simulator $B_{2mk(j+1)}$ that leverages $A$ as a black box to solve Problem 2. The procedure is shown as follows:

1. $B_{2mk(j+1)}$ is given an instance of Problem 2, i.e., $(\text{param}_{2^{(m,k,j')}}(0), b^*(0), \hat{b}^*(0)), t(0), \{\hat{b}^*(k), h_{\beta,j}^{(k)}, l_i(1, \ldots, n_i) = k = 1, 2\}$ where $n_1 = n$ and $n_2 = L$, and acts as a challenger in the security game against adversary $A$.

2. At the beginning of the game, $B_{2mk(j+1)}$ gives $A$ the public key $PK = (1^\lambda, \text{param}_{2^{(m,k,j')}}, b^*_0, b^*_1, \ldots, b^*_n, b^*_1, b^*_2, \ldots, b^*_L, b^*_1, b^*_2, \ldots, b^*_L, b^*_n)$, which is obtained from the Problem 2 instance.

3. The answer to the $s$-th key in the $k$-th $sk_{w,l}$ in the $m$-th $SK_{i,l}$ query for time period $i$ and hierarchical predicate vectors $(\vec{x}_1, \ldots, \vec{x}_l)$ is as follows:

a) For $0 \leq s \leq j$ the algorithm $B_{2mk(j+1)}$ computes a semi-functional key using $\{B^*_k\}_{k=0,1,2}$ of the Problem 2 instance as follows:

$$\text{when } j=0, \text{ it computes }$$

$$\begin{align*}
(k_{w,l,\text{dec}}^{(0)}, k_{w,l,\text{dec}}^{(1)}, k_{w,l,\text{dec}}^{(2)})
\end{align*}$$

using $\{h^*_0, b^*_1, b^*_3, h^*_0(i), b^*_3(i), h_{\beta,j}^{(1)}, l_{i}(1, \ldots, n_i) = k = 1, 2, j = 1, \ldots, n_i\}$ of the Problem 2 instance as follows:

For $i = 1, 2$:

$$s_{\beta}^{(0)} = \sum_{i=1}^{2} (\theta_i h^*_\beta + v_i b^*_i), \quad k_{w,l,\text{dec}}^{(0)} = -s_{\beta}^{(0)} + r^* b^*_2 + b^*_3,$$

For $i = 1, 2$ and $j = 1, \ldots, n_i$:

$$s_{\beta,j}^{(i)} = \theta_i h^*_\beta + v_i b^*_i, \quad s_{\beta,j}^{(i)} = \theta_i h^*_\beta + v_i b^*_i,$$

$$k_{w,l,\text{dec}}^{(1)} = \sum_{k=1}^{l} (\theta_k s_{\beta,j}^{(1)} + v_k b_{n+k}^{(1)}), \quad k_{w,l,\text{dec}}^{(2)} = \sum_{k=1}^{l} (\theta_k s_{\beta,j}^{(2)} + v_k b_{n+k}^{(2)}),$$

where $v_k = (k = 1, \ldots, n), v_k' = (k' = 1, \ldots, L) \leftarrow F_q$. 

b) For $s = j + 1$ it computes as follows:

$$\begin{align*}
&\text{For } i = 1, 2 : \quad q_i, v_i, v'_i, \theta_i, r, \theta_i(k = 1, \ldots, l), \phi_{k'}(k' = 1, \ldots, r) \leftarrow F_q; \\
&s_{\beta}^{(0)} = \sum_{i=1}^{2} (\theta_i h^*_\beta + v_i b^*_i), \quad k_{w,l,\text{dec}}^{(0)} = -s_{\beta}^{(0)} + r^* b^*_2 + b^*_3, \\
&\text{For } i = 1, 2 \text{ and } j = 1, \ldots, n_i: \\
&s_{\beta,j}^{(i)} = \theta_i h^*_\beta + v_i b^*_i, \quad s_{\beta,j}^{(i)} = \theta_i h^*_\beta + v_i b^*_i, \\
&k_{w,l,\text{dec}}^{(1)} = \sum_{k=1}^{l} (\theta_k s_{\beta,j}^{(1)} + v_k b_{n+k}), \quad k_{w,l,\text{dec}}^{(2)} = \sum_{k=1}^{l} (\theta_k s_{\beta,j}^{(2)} + v_k b_{n+k}), \\
\end{align*}$$

where $v_k = (k = 1, \ldots, n), v_k' = (k' = 1, \ldots, L) \leftarrow F_q$. 

when \( j > 0 \), compute randomness and delegation components using \( \{b_{\beta,j}^{(i)},h_{\beta,j}^{(i)}\}_{i=1,2,j=1,\ldots,n} \) of the Problem 2 instance as follows:

4. When \( B_{2mk(j+1)} \) receives challenge attribute vectors \( Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h(0)}^{(0)}) \) and \( Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h(1)}^{(1)}) \), challenge plaintexts \( (M^{(0)}, M^{(1)}) \) and a timer period \( I \) from \( \mathcal{A} \), \( B_{2mk(j+1)} \) computes and returns the ciphertext \( C = (c^{(0)}, c^{(1)}, c^{(2)}, c^{(3)}) \) where

\[
\begin{align*}
c^{(0)} &= f^{(0)} + \zeta h_3^{(0)} + \phi b_5^{(0)}, \\
c^{(1)} &= \sum_{i=1}^{\mu(b)} g_i^{(b)} t_i^{(1)} + \sum_{i=\mu(b)+1}^n g_i t_i^{(1)} + \phi h_{\|n+1}^{(1)}, \\
c^{(2)} &= \sum_{j=1}^L I_j f_j^{(2)} + \phi b_{3L+1}^{(2)}, \\
c^{(3)} &= g_1^{(b)} M^{(b)},
\end{align*}
\]

using \( \{t_i^{(1)}\}_{i=1,\ldots,n}, \{t_i^{(2)}\}_{i=1,\ldots,L}, h_3^{(0)}, b_5^{(0)}, b_{\|n+1}^{(1)}, b_{3L+1}^{(2)} \) from the instance of Problem 2 and \( (\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h(0)}^{(b)}), M^{(b)}, I \) where \( \zeta, \phi \) \( h_3^{(0)}, f_j^{(2)}, h_{\|n+1}, \ldots, y_n \) \( F_q, b \) \( \in \{0,1\} \). \( I_{2k-1}, I_{2k} \) denotes the vector for \( i_k \), and \( i_1, \ldots, i_k \) is parsed from \( I \).

5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.

6. \( \mathcal{A} \) outputs a bit \( b' \). If \( b = b' \), \( B_{2mk(j+1)} \) outputs 1. Otherwise, \( B_{2mk(j+1)} \) outputs 0.

**Claim 6.3.** The distribution of the view of adversary \( \mathcal{A} \) in the above-mentioned game simulated by \( B_{2mk(j+1)} \) given a Problem 2 instance with \( \beta \in \{0,1\} \) is the same as that in Game 2-(m, k, j+}
1) (resp. Game 2-(m, k, j')) if β = 0 (resp. β = 1) except with probability 4/q (resp. 1/q).

Proof. It is clear that $B_{2mk(j+1)}$’s simulation of the public key generation (step 2) and the answers to the s-th query where $s \neq j + 1$ (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and delegation oracle in Game 2-(m, k, j + 1) and Game 2-(m, k, j').

Next we analyze the distribution of the s-th key in the k-th sk_{w,l} in the m-th $SK_{i,t}$ query for time period $i$ and hierarchical predicate vectors ($\overrightarrow{x}_1, \ldots, \overrightarrow{x}_t$) where $s = j + 1$ (case (b) of steps (3) and (5)). In this case values $s^{(0)}_\beta, s^{(i)}_{\beta,j}, s^{(i)}_{\beta,j}$, $i = 1, 2, j = 1, \ldots, n_i$ can be expressed as follows. Let $\beta^{(i)} = \theta_i + \omega_i$, $\alpha^{(i)} = g_i + \omega_i$, $\alpha = \alpha^{(1)} + \alpha^{(2)}$, $\gamma = g_1 + g_2$. Then,

$$s^{(0)}_0 = (\alpha, 0, 0, \gamma, 0)_{B^{(0)}}, \quad s^{(0)}_1 = (\alpha, \epsilon, 0, \gamma, 0)_{B^{(0)}},$$

$$s^{(i)}_{\beta,j} = \left(\frac{\gamma^{(i)}_{\beta,j}}{\epsilon^{(i)}_{\beta,j}}, \frac{\theta^{(i)}_{\beta,j}}{\epsilon^{(i)}_{\beta,j}}, 1, 0\right)_{B^{(i)}}, \quad s^{(i)}_{\beta,j} = \left(\frac{\gamma^{(i)}_{\beta,j}}{\epsilon^{(i)}_{\beta,j}}, \frac{\theta^{(i)}_{\beta,j}}{\epsilon^{(i)}_{\beta,j}}, 1, 0\right)_{B^{(i)}},$$

where $\gamma^{(i)}_{\beta,j} = z^{(i)}_{\beta,j}, \ldots, z^{(i)}_{\beta,j}, \omega, \xi, \{\gamma^{(i)}_{\beta,j}(0)\}_{i=1,2, \ldots, n_i}$ are defined as in Problem 2. If $\beta = 1$ in the instance of Problem 2 then the decryption component $(\epsilon^{(i)}_{w,l,dec} F^{(i)}_{w,l,dec} F^{(i)}_{w,l,dec})$ has the same distribution as in Eq. 5.5 except that $(\gamma^{(i)}_{dec} F^{(i)}_{1, dec} F^{(i)}_{1, dec}) (1 + \sigma^{(1)}_{dec} F^{(1)}_{1, dec}, \ldots, \sigma^{(1)}_{dec} F^{(1)}_{1, dec}, 0^{n-\mu})$. $Z^{(1)} + \tilde{v}_1$ and $(\gamma^{(2)}_{dec} F^{(2)}_{1, dec} F^{(2)}_{1, dec}) (1 + \sigma^{(1)}_{dec} F^{(1)}_{1, dec}, \ldots, \sigma^{(1)}_{dec} F^{(1)}_{1, dec}, 0^{L-2r_{2}}). Z^{(2)} + \tilde{v}_2$ where $\gamma^{(1)}_{1} \equiv \tilde{v}_1 \equiv \tilde{v}_2 \equiv F^{(1)}_{1, dec}.$

Randomness and delegation components for the hierarchical predicate and time period have the same distribution as in Eq. 6.6 and 7.7, respectively, except the added $\gamma^{1}_{\beta,j}$ and $\tilde{v}_2^{1}$ randomness.

Next, we show that the joint distribution of the response to j-th key in the k-th sk_{w,l} in the m-th $SK_{i,t}$ query and of the challenge ciphertext in the simulation by $B_{2mk(j+1)}$ for the given instance of Problem 2 is equivalent to the distribution in Game 2-(m, k, j + 1) if $\beta = 0$ and to the distribution in Game 2-(m, k, j') if $\beta = 1$.

If $\beta = 0$ then this equivalence follows easily, unless one of the following conditions holds:

(1) $\omega$ defined in Problem 2 is zero, (2) $w = 0, (3) \tilde{w}_1 = \tilde{0}, (4) \tilde{w}_2 = \tilde{0}$. However, those events occur with probability $4/q$.

If $\beta = 1$, then $B_{2mk(j+1)}$’s simulation for the key is the same as that expressed in Eqs. 6.5 and 6.6, and $B_{2mk(j+1)}$’s simulation for the challenge ciphertext is the same as that expressed in Eqs. 6.8 except that $(\gamma^{(1)}_{dec} F^{(1)}_{1}) + \sigma^{(1)}_{dec} F^{(1)}, \ldots, \sigma^{(1)}_{dec} F^{(1)}, 0^{n-\mu}) Z^{(1)} + \tilde{v}_1 = F^{(1)}_{1, dec}.$

Therefore, we will show that $(\gamma^{(2)}_{dec} F^{(2)}_{1}) + \sigma^{(1)}_{dec} F^{(1)}, \ldots, \sigma^{(1)}_{dec} F^{(1)}, 0^{L-2r_{2}}) Z^{(2)} + \tilde{v}_2$ are uniformly distributed and independent from the other variables used in the simulation by $B_{2mk(j+1)}$. Let $\tilde{A}_1 = (g_1 F^{(1)} + \theta_1 I_1, \ldots, \theta_{L-2r_{2}} I_1, 0^{n-\mu}), Z^{(1)} + \tilde{v}_1, \tilde{A}_2 = (g_2 F^{(2)} + \theta_1 I_1, \ldots, \theta_{L-2r_{2}} I_1, 0^{n-\mu}), Z^{(2)} + \tilde{v}_2$, and $\tilde{B}_1 = (F^{(1)}, \ldots, F^{(1)} I_1, \ldots, \theta_{L-2r_{2}} I_1, 0^{n-\mu}), U^{(1)}, \tilde{B}_2 = (I_1, \ldots, I_1) U^{(2)}$, where $I_1 = (1, -i_2)$ and $i_1, \ldots, i_k$ is parsed from I. We analyze joint distribution of these variables for the cases that appear in Definition 6.2.

1. When $i > I$, i.e., the time period of the queried key is after the time period of encoded
in the ciphertext, due to Lemma 6.9 the pair \( (\vec{A}, \vec{B}) \) is uniformly and independently distributed over \( \mathbb{F}_q^{2n} \).

2. When \( i \leq I \) and \( f(\vec{x}_1, \ldots, \vec{y}_{(0)}) = f(\vec{x}_1, \ldots, \vec{y}_{(1)}) = 0 \), the pair \( (\vec{A}, \vec{B}) \) is uniformly and independently distributed over \( \mathbb{F}_q^{2n} \). The pair \( (\vec{A}, \vec{B}) \) is uniformly and independently distributed over \( \mathbb{F}_q^{2L} \).

Considering the adversary \( A \)'s restriction on key queries from Definition 6.2 in above two cases at least one of \( (\vec{A}, \vec{B}) \) and \( (\vec{A}, \vec{B}) \) is uniformly and independently distributed over \( \mathbb{F}_q^{2n+1} \) for \( k = 1, 2 \). Therefore, \( (\gamma^{(1)}_{\text{dec}} \vec{v}_{1}^{(1)} + \sigma^{(1)}_{\text{dec}} \vec{r}_{1}^{(1)}, \ldots, \sigma^{(1)}_{\text{dec}} \vec{r}_{1}^{(1)}), \sigma^{(1)}_{\text{dec}} \vec{r}_{1}^{(1)}, \sigma^{(1)}_{\text{dec}} \vec{r}_{1}^{(1)} \) and \( (\gamma^{(2)}_{\text{dec}} \vec{v}_{1}^{(2)} + \sigma^{(2)}_{\text{dec}} \vec{r}_{1}^{(2)} + \sigma^{(2)}_{\text{dec}} \vec{r}_{1}^{(2)} + \sigma^{(2)}_{\text{dec}} \vec{r}_{1}^{(2)}) \) are uniformly distributed and independent from the other variables used in the simulation by \( B_{2m(k+1)} \).

Therefore, the view of \( A \) in the game simulated by \( B_{2m(k+1)} \) on input an instance of Problem 2 with \( \beta = 1 \) is the same as in Game 2-(\( m, k, j \)) unless \( \omega = 0 \) occurs. This event happens with probability \( 1/q \).

This completes the proof of Lemma 6.12

**Lemma 6.13.** For any adversary \( \mathcal{A} \), \( \text{Adv}^{(3)}(\lambda) \leq \text{Adv}^{(1-\nu,0)}(\lambda) + 1/q \).

**Proof.** First we show the distribution (parametrize, \( \{\widehat{\mathcal{B}}^{(k)}\}_{k=0,1,2}, \{SK_i^{(j)}\}_{j=1,\ldots,\nu}, C = (e^{(0)}, e^{(1)}, e^{(2)}, e^{(M)}) \) of Game 3 is same as that of Game 2-(\( \nu, 0, 0 \)), where \( SK_i^{(j)} \) is the answer to the \( j \)-th key query, and \( C = (e^{(0)}, e^{(1)}, e^{(2)}, e^{(M)}) \) is the challenge ciphertext. We will define new bases \( \mathbb{D}^{(k)} \) of \( \mathbb{V}_k \) and \( \mathbb{D}^{(k)} \) of \( \mathbb{V}_k \) for \( k = 0, 1, 2 \).

For \( k = 0 \), we set \( \mathbb{D}^{(1)} = \mathbb{D}^{(0)} - \lambda \mathbb{D}^{(0)} \) and \( \mathbb{D}^{(1)} = \mathbb{D}^{(0)} + \lambda \mathbb{D}^{(0)} \), where \( \lambda \in \mathbb{F}_q \). The new bases are \( \mathbb{D}^{(0)} = (b_{1}^{(0)}, b_{2}^{(0)}, b_{3}^{(0)}, b_{4}^{(0)}, b_{5}^{(0)}) \) and \( \mathbb{D}^{(1)} = (b_{1}^{(0)}, b_{2}^{(0)}, b_{3}^{(0)}, b_{4}^{(0)}, b_{5}^{(0)}) \). We can easily verify that \( \mathbb{D}^{(0)} \) and \( \mathbb{D}^{(1)} \) are dual orthonormal, and are distributed the same as the original bases \( \mathbb{B}^{(0)} \) and \( \mathbb{B}^{(1)} \) respectively.

For \( i, j = 1, \ldots, n \), choose \( Q^{(i)} = (\mu_{i,j}^{(1)})^{T} \in \mathbb{F}_q^{n \times n} \), and compute \( e^{(1)} = b_{n+i}^{(1)} + \sum_{j=1}^{n} \mu_{i,j} b_{j}^{(1)} \), \( e^{(1)} = b_{i}^{(1)} - \sum_{j=1}^{n} \mu_{j,i} b_{n+j}^{(1)} \), which are equivalent to the following matrix computations:

\[
\begin{pmatrix}
\vec{B}_{1}^{(1)} \\
\vec{B}_{2}^{(1)}
\end{pmatrix} =
\begin{pmatrix}
I_n & 0_n \\
Q^{(1)} & I_n
\end{pmatrix}
\begin{pmatrix}
\vec{B}_{1}^{(1)} \\
\vec{B}_{2}^{(1)}
\end{pmatrix},
\begin{pmatrix}
\vec{B}_{1}^{(1)} \\
\vec{B}_{2}^{(1)}
\end{pmatrix} =
\begin{pmatrix}
I_n & -Q^{(1)} \\
0_n & I_n
\end{pmatrix}
\begin{pmatrix}
\vec{B}_{1}^{(1)} \\
\vec{B}_{2}^{(1)}
\end{pmatrix}.
\]

where \( \vec{B}_{1}^{(1)} = (b_{1}^{(1)}, \ldots, b_{n+1}^{(1)})^{T} \), \( \vec{B}_{2}^{(1)} = (b_{n+1}^{(1)}, \ldots, b_{2n}^{(1)})^{T} \), \( \vec{B}_{1}^{(1)} = (b_{1}^{(1)}, \ldots, b_{n}^{(1)})^{T} \), \( \vec{B}_{2}^{(1)} = (b_{n}^{(1)}, \ldots, b_{2n+1}^{(1)})^{T} \).

The new bases are \( \mathbb{D}^{(1)} = (b_{1}^{(1)}, \ldots, b_{n+1}^{(1)}, d_{1}^{(1)}, d_{2}^{(1)}, b_{2n+1}^{(1)}, \ldots, b_{3n+1}^{(1)}) \) and \( \mathbb{D}^{(1)} = (d_{1}^{(1)}, \ldots, d_{n+1}^{(1)}, b_{n+1}^{(1)}, \ldots, b_{2n+1}^{(1)}, b_{2n+1}^{(1)}, \ldots, b_{3n+1}^{(1)}) \). It is clear that \( \mathbb{D}^{(1)} \) and \( \mathbb{D}^{(1)} \) are dual orthonormal, and are distributed the same as the original bases \( \mathbb{B}^{(1)} \) and \( \mathbb{B}^{(1)} \) respectively. Bases \( \mathbb{D}^{(2)} \) and \( \mathbb{D}^{(2)} \) are the same as bases \( \mathbb{B}^{(2)} \) and \( \mathbb{B}^{(2)} \) respectively.

Keys and challenge ciphertext \( \{(SK_i^{(j)}\}_{j=1,\ldots,\nu}, C = (e^{(0)}, e^{(1)}, e^{(2)}, e^{(M)}) \) of Game 2-\( \nu \) are expressed over bases \( \mathbb{B}^{(k)} \) and \( \mathbb{B}^{(k)} \) for \( k = 0, 1, 2 \) as follows:
Parse \( SK_{1,l}^{(j)} \) as \( (s_{k_{1},1}^{(j)}, \ldots, s_{k_{1},1,l-1,j}^{(j)}, s_{k_{1},1,l,j}^{(j)}) = 0) \). Parse each \( sk_{w,l} \) as \( (k_{w,l,dec,j}, k_{w,l,dec,j}^{(1)}, k_{w,l,dec,j}^{(2)}, \ldots, k_{w,l,ran,l+1,j}^{(2)}, \ldots, k_{w,l,ran,1,j}^{(1)}, \ldots, k_{w,l,ran,r+1,j}^{(2)}, \ldots, k_{w,l,dec,(2r+1),j}^{(1)}, k_{w,l,dec,(2r+1),j}^{(2)}) \).

For simplicity we only show the transformation of decryption key component \( (k_{w,l,dec,j}^{(0)}, k_{w,l,dec,j}^{(1)}, k_{w,l,dec,j}^{(2)}) \), randomness and delegation component can be similarly constructed.

\[
k_{w,l,dec,j}^{(0)} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(0)}}
\]

\[
k_{w,l,dec,j}^{(1)} = (\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(1)}}
\]

\[
k_{w,l,dec,j}^{(2)} = (\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(2)}}
\]

Above keys and challenge ciphertext can also be expressed over bases \( \mathbb{D}^{(k)} \) and \( \mathbb{D}^{(r)} \) for \( k = 0, 1, 2 \) as:

\[
k_{i,l,dec,j}^{(0)} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(0)}} = (-\alpha_j, \theta_j, 1, \eta_j, 0)_{B^{(0)}},
\]

where \( \theta_j = \epsilon_j - \lambda \) which are uniformly, independently distributed since \( \epsilon_j \overset{\$}{\leftarrow} \mathbb{F}_q \).

\[
k_{i,l,dec,j}^{(1)} = (\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(1)}}
\]

\[
k_{i,l,dec,j}^{(2)} = (\alpha_j, \epsilon_j, 1, \eta_j, 0)_{B^{(2)}}
\]

where \( \theta_{i,j} = \mu_{i,1}^{(1)} \alpha_{i,1}^{(1)} + \beta_{i,dec,j}^{(1)} \alpha_{i,1,j}^{(1)} \epsilon_{i,1}^{(1)} + \cdots + \beta_{i,dec,j}^{(1)} \alpha_{i,1,j}^{(1)} \epsilon_{i,1}^{(1)} + \gamma_{i,j}^{(1)} \) for \( i = 1, \ldots, n, j = 1, \ldots, \nu \), which are uniformly, independently distributed since \( \gamma_{i,j}^{(1)} \overset{\$}{\leftarrow} \mathbb{F}_q \).

\[
c_0 = (\delta, w, \zeta, 0, \varphi)_{\mathbb{D}^{(0)}} = (\delta, w, \zeta', 0, \varphi)_{\mathbb{D}^{(0)}},
\]

where \( \zeta' = \zeta + \lambda w \) which are uniformly, independently distributed since \( w, \zeta \overset{\$}{\leftarrow} \mathbb{F}_q \).
6.5 Conclusion

In this chapter, we introduced the accepted notion of forward security to a powerful setting of hierarchical predicate encryption. The resulting FS-HPE scheme offers time-independent delegation of predicates, autonomous update for users’ private keys, and the encryption process does not require knowledge of predicates at any level including when those predicates join the hierarchy. The scheme is forward-secure and adaptively attribute-hiding under chosen plaintext attacks under the DLIN assumption in the standard model.
Chapter 7

Conclusion

In this thesis, we study the predicate encryption with different properties. In summary we have investigated the following open problems.

In chapter 3, we focus on PE for multi inner product (PEM). We show the predicate of the multi inner product. We also present the syntax of PEM and its security definition. We then provide two PEM schemes that can be used to evaluate multi inner product predicate and prove the schemes to be selective secure in the standard model. Furthermore, we show a way to use the proposed PEMs to realize the system evaluating disjunctive comparison and disjunctive subset queries. Finally, we extend our technique for disjunctive queries to achieve arbitrary combinations of disjunctive and conjunctive predicate evaluations.

In Chapter 4, we show the notion of Revocable Predicate Encryption (RPE), which extends the previous PE setting with revocation support. We formalize attribute hiding in the presence of revocation and propose our first RPE with attribute hiding property, i.e., attribute is hidden in the ciphertext. The proposed scheme is proved to be secure under DLIN assumption in the standard model.

Chapter 5 is devoted to the study of a RPE with stronger security, namely RPE with full hiding. In FH-RPE, in addition to attribute hiding, the scheme ensures that no information about revoked users is leaked from a given ciphertext. We provide the security model and prove the proposed RPE scheme is adaptively full hiding against CPA in the standard model.

In Chapter 6, we present the first forward secure hierarchical predicate encryption (FS-HPE). We first give a new syntax and security definitions for FS-HPE. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates, local update for users’ private keys, forward security, and the scheme’s encryption process does not require knowledge of predicates at any level including when those predicates join the hierarchy. Finally, we analyze the security of proposed FS-HPE in the standard model.

In this thesis, we study some properties of PE, but there are still many open problems in this line of research. For example, the expressiveness of PE was extensively investigated by many researchers. However, we still do not know how to evaluate negation in the attribute hiding setting. In revocable PE, an authority may manage a revocation list, and create a token that
encodes the list. The token is published and it hides the revocation list. To encrypt a message, users download the token first, and encrypt it with the message as well as an attribute. The advantage of using token is that the revocation list is managed by an authority, meanwhile the privacy of the revoked indexes is preserved, i.e., the token hides the revocation list. However, at the time of writing, we cannot construct a security model for indistinguishability of token due to the technique limitation. The PE [43], that we used to construct our RPE, is in weak attribute hiding, i.e., the attribute hiding is only secure against the adversary that cannot decrypt the ciphertext. The security definition for revocation list hiding in token requires that the adversary gives two challenge lists to the challenger, and obtain the corresponding token computed by the challenger. The adversary then may be able to create a ciphertext encrypted with the challenge token, and decrypt that ciphertext. Since the system is in wak attribute hiding, the adversary may be able to obtain some information about the revocation list, which in turn breaks the security of revocation list hiding in token. Recently, Okamoto and Takashima [46] propose a new PE with full attribute hiding, i.e., the attribute hiding is secure against the adversary who can decrypt the ciphertext. It seems promising that we can construct a security model for revocation list hiding in token using their new technique, and propose the new schemes under that model.

We are also interested in other predicate-based primitives such as predicate signcryption. Signcryption [63] combines signature and encryption schemes in a way that achieves better efficiency. There are heaps of works addressing various problems in signcryption, but we note that the problem of predicate signcryption is still open. PE has been extensively studied, and anonymous attribute-based signature is also proposed. Can we using the existing schemes to construct an efficient predicate signcryption? or perhaps a generic construction? We may also propose online/offline predicate signcryption to further reduce the workload between senders and receivers, which in turn can be applied in lightweight devices.

Re-encryption [7] is another primitive that we would like to study in the predicate-based settings. It is not straightforward to construct predicate re-encryption due the rich algebra structures and complex relations between attribute and predicate. Moreover, we have to make sure that the re-encryption scheme will not compromise the attribute hiding property.

Since the constructions in this thesis are all pairing based, it is inherently expensive due to pairing computation. A promising solution is to migrate our system in the lattice settings. To our knowledge, there are no works addressing the properties investigated in this thesis based on lattice technique. Hence, one of our future aims is to construct systems with small and constant size keys and ciphertexts, and reducing the computational coast using lattice settings.

All in all, predicate primitive is a young field, we are confident that it will be more and more important in practice due to its expressiveness and enhanced privacy property.
Bibliography


