UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA "RENATO M. CAPOCELLI"

CORSO DI DOTTORATO IN
"TEORIE, METODOLOGIE E APPLICAZIONI AVANZATE PER LA COMUNICAZIONE, L'INFORMATICA E LA FISICA"
XI CICLO – NUOVA SERIE

ANNO ACCADEMICO 2011-2012

TESI DI DOTTORATO IN INFORMATICA

On the Generalizations of Identity-Based Encryption

Hidden Vector Encryption and Inner-Product

Tutor
prof. **Carlo Blundo**

Candidato
Angelo De Caro

Coordinatore
prof. **Giuseppe Persiano**

Abstract

Nowadays, public-key cryptographic is widely deployed and successfully used but still a major drawback exists. In fact, from encrypted data a party can either decrypt or cannot learn anything at all about the message other than the intentionally leaked information such as its length. In the recent years, the cloud computing paradigm has emerged as the new standard to use computing resources, such as storage devices, that are delivered as a service over a network. In such a scenario, the notion of public key cryptography is not enough. It would be desirable to specify a decryption policy in the encrypted data in such a way that only the parties who satisfy the policy can decrypt. In a more general form, we may want to only give access to a function of the message, depending on the decryptor's authorization.

Thus, in the last decade researchers have started looking at a more sophisticated type of encryption called *functional encryption*. A *functionality* F is a function $F : K \times M \to \Sigma$ where K is the *key space* and M is the *message space*. Then, a functional encryption scheme, in the public-key setting, for F is a special encryption scheme in which, for every *key* $k \in K$, the owner of the master secret key msk associated with the master public key mpk can generate a special secret-key sk_k that allows the computation of $F(k, m)$ from a ciphertext of $m \in M$ computed under public key mpk. In other words, whereas in traditional encryption schemes decryption is an all-or-nothing affair, in functional encryption it is possible to finely control the amount of information that is revealed by a ciphertext. One of the most notable example of functional encryption is *identity-based encryption* first introduced by Shamir as an alternative to the standard notion of public-key encryption.

In this thesis, we discuss several instantiations of function encryption that can all be seen as generalizations of identity-based encryption. We improve on previous constructions in terms of capabilities and security guarantees.

Acknowledgments

Non posso che ringraziare i miei tutor prof. Carlo Blundo e prof. Giuseppe Persiano per la loro pazienza e la loro disponibilità nell'assistermi in questi anni di dottorato. Vorrei inoltre rivolgere un ringraziamento sincero e speciale al prof. Michel Abdalla che mi ha accolto nel suo dipartimento in visita offrendomi la possibilità di maturare sia da un punto di vista scientifico sia umano. Infine, un ringraziamento "abbastanza" speciale al mio coautore Vincenzo Iovino, un amico prezioso e stravagante che sorprende sempre per la sua acutezza di spirito.

Contents

Chapter 1

Introduction

A set of parties can securely exchange information over an *insecure network* or *storage service* by using *encryption*. In the simpler scenario, called *symmetric setting or private-key setting*, the parties must first agree on a common secret key k (using a secondary channel assumed to be secure), before they can start any secret communication. In this setting, the secret key k is involved in both the encryption and decryption process. Such a solution can be practical for a small organization but it is not feasible for larger networks such as Internet where millions of parties are involved.

In the 1970s, with the advent of *public-key cryptography* the issue of having an a prior mutual secret (and thus a secure communication channel) has been solved radically. In the *asymmetric setting or public-key setting*, the decryption key (also called secret key) differs from the encryption key (also called public key) and it is infeasible to find the decryption key, given the encryption key. In this setting, a party P generates a pair of keys and keeps the decryption key. The encryption key is then published and now any party can send P private messages by encrypting them using P's encryption key. Then, only P will be able to recover those messages by using its decryption key.

Nowadays, public-key cryptographic is widely deployed and successfully used. But still some issues remain. In fact, from encrypted data a party can either decrypt or cannot learn anything at all about the message other than the intentionally leaked information such as its length. In the recent years, the cloud computing paradigm has emerged as the new standard to use computing resources, such as storage devices, that are delivered as a service over a network. In such a scenario, the notion of public key cryptography is not enough. It would be desirable to specify a decryption policy in the encrypted data in such a way that only the parties who satisfy the policy can decrypt. In a more general form, we may want to only give access to a function of the message, depending on the decryptor's authorization.

With these motivations in mind, researchers have started looking at a more sophisticated type of encryption called *functional encryption*. A *functionality* F is a function $F : K \times M \rightarrow \Sigma$ where K is the *key space* and M is the *message space*. Then, a functional encryption scheme, in the public-key setting, for F is a special encryption scheme in which, for every *key* $k \in K$, the owner of the master secret key msk associated with the master public key mpk can generate a special secret key (also called *token*) sk_k that allows the computation of $F(k, m)$ from a ciphertext of $m \in M$ computed under public key mpk. In other words, whereas in traditional encryption schemes decryption is an all-or-nothing affair, in functional encryption it is possible to finely control the amount of information that is revealed by a ciphertext.

One of the most notable example of functional encryption is *identity-based encryption* (IBE, for short) first introduced by Shamir [Sha85] as an alternative to the standard notion of public-key encryption. In an IBE scheme, the public key associated with a user can be an arbitrary identity string, such as his email address, and others can send encrypted messages to a user using this arbitrary identity without having to rely on a public-key infrastructure. In these systems, there also exists a *trusted central authority* capable of generating secret keys for any user in the system. The first practical constructions of an IBE are due to Boneh and Franklin [BF03] and Cocks [Coc01]. While the Boneh-Franklin IBE scheme is based on elliptic curve groups equipped with efficiently computable bilinear maps, the scheme by Cocks is based on the quadratic residuosity problem [Coc01]. In addition to the scheme, Boneh and Franklin also introduced a formal security definition for IBE schemes to model privacy, known as indistinguishability under chosen-identity-attacks, and showed that their scheme was provably secure under this notion based on a new computational problem, known as bilinear Diffie-Hellman problem, in the random oracle model [BR93]. However, access to the data itself is still inherently all-or-nothing. The decryptor either will be able to decrypt the data and learn everything or will not and thus learn nothing. This has motivated the study of *anonymous* identity-based encryption schemes, where the ciphertext does not leak the identity of the recipient. The Boneh-Franklin IBE scheme was already known to be inherently anonymous but the first anonymous IBE without random oracles was proposed by Boyen and Waters [BW06]. Moreover, anonymity can be leveraged to construct Public key Encryption with Keyword Search (PEKS) schemes, as observed by Boneh *et al.* [BDOP04] and later formalized by Abdalla *et al.* [ABC$^+$08]. Following the work of Boneh and Franklin, Horwitz and Lynn [HL02] proposed the concept of Hierarchical IBE (HIBE, for short), in which users are organized in a tree of depth L, with the root being the trusted central authority. In a HIBE scheme, intermediate nodes in the tree are capable of generating secret keys for any of their descendants. Since its introduction, several practical constructions of HIBE schemes have been proposed in the literature (e.g., [GS02, CHK03, BB04, BBG05]), with the first being due to Gentry and Silverberg [GS02], who proposed a scheme based on the Boneh-Franklin IBE scheme and proved it secure in the random-oracle model based on the bilinear Diffie-Hellman problem.

After the realization of the first Identity-Based Encryption schemes a number of new cryptosystems have provided increasing functionality and expressiveness of decryption capabilities. In particular, one of the most relevant generalization of identity based encryption, known in the literature as *Attribute-Based Encryption* (ABE, for short), was proposed by Sahai and Waters [SW05] to express complex access policies. Subsequently, Goyal, Pandey, Sahai and Waters [GPSW06] refined this concept into two different formulations of ABE: *Key Policy* ABE and *Ciphertext-Policy* ABE. In a key policy ABE, a ciphertexts contains a vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ of n Boolean values and a secret key contains a poly-sized boolean formula ϕ over n Boolean variables. The formula ϕ must evaluates to true over $\boldsymbol{v}$ to be able to recover the encrypted message. On the other side, two notable generalizations of anonymous identity based encryption are known in the literature as *hidden vector encryption* (HVE, for short) and *inner-product encryption* (IPE, for short). In an hidden vector encryption scheme, first proposed by Boneh and Waters [BW07], a ciphertext contains a vector $\boldsymbol{v} = (v_1, \ldots, v_\ell)$ of ℓ elements in $\{0,1\}^*$ and a secret key contains a vector $\boldsymbol{w} = (w_1, \ldots, w_\ell)$ of ℓ elements in $\{0,1\}^* \cup \{\star\}$ where we refer to $\star$ as a wildcard character. The decryptor can then compute a predicate of $\boldsymbol{v}$ and $\boldsymbol{w}$ that is satisfied if and only if for each $i = 1, \ldots, \ell$, $v_i = w_i$ whenever $v_i \neq \star$. Applications of such encryption scheme include conjunctive and range searches. Later, Katz, Sahai and Waters [KSW08] proposed inner-product encryption, a more general system to test if a dot product operation over the ring $\mathbb{Z}_N$ is equal to 0, where N is the product of three random primes chosen by the setup algorithm. Despite this apparently restrictive structure, IPE can support conjunction, subset and range queries on encrypted data as well as disjunction, polynomial evaluation, and CNF and DNF formulas. Following the work of Katz, Sahai and Waters, Okamoto and Takashima [OT09] proposed the concept of Hierarchical IPE (HIPE, for short). All the previous constructions of

IPE and HIPE schemes were based on elliptic curve groups equipped with efficiently computable bilinear maps. More recently, Agrawal *et al.* [AFV11] proposed a *lattice-based* functional encryption scheme for inner product whose security follows from the difficulty of the learning with errors problem (LWE, for short). The LWE problem, first introduced by Regev [Reg05], has received a lot of attention in the recent years due to its versatility and because it is also conjectured to be hard to solve even for quantum adversaries.

One of the most important achievements of the modern theory of Cryptography is the formalization of the notion of a *secure cryptosystem* given in the seminal work of Goldwasser and Micali [GM84]. There, two notions of security were proposed, one is a *game-based* notion and the other is *simulation-based*, and they were shown to be equivalent in the sense that an encryption scheme meets one security definition if and only if it meets the other. The study of simulation-based notions of security for functional encryption was initiated by Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10]. Specifically, in [BSW11] Boneh *et al.* showed that there exists a clearly insecure functional encryption scheme that is nonetheless deemed secure by the game-based notion of security. This motivated the study of a simulation-based notion of security for functional encryption. In the same paper [BSW11], Boneh *et al.* gave impossibility results in the non-programmable random oracle model for simple instances of functional encryption for which, instead, game-based security had been shown to be achievable (again, under appropriate complexity assumptions). Specifically, they showed that it is not possible to construct Identity Based Encryption that is secure against an *adaptive* adversary that can ask to see the encryption of an *unbounded* number of messages. Informally, this is because any simulation-based definition that allows the adversary to query for secret keys after seeing the challenge ciphertext must achieve something very similar to non-interactive non-committing encryption for which impossibility results are already known [Nie02].

The impossibility result of [BSW11] was recently shown to hold also in the standard model by Bellare and O'Neill in [BO12] under an additional complexity assumption. Recently, Gorbunov, Vaikuntanathan, and Wee [GVW12], building on the construction of Sahai and Seyalioglu [SS10], gave a functional encryption scheme for all polynomial size circuits that is simulation-based secure against adaptive adversaries that have access to a *bounded* number of tokens. This is complemented by the recent result of Agrawal, Gorbunov, Vaikuntanathan, and Wee [AGVW12] that shows that there exists no functional encryption scheme for the functionality of poly-size circuits that is simulation-based secure against non-adaptive adversaries that can see one ciphertext and ask for an unbounded number of tokens. O'Neill [O'N10] gave a different notion of simulation-based security and introduced a simple condition on a functionality (called *pre-image samplability*). O'Neill showed that, for pre-image sampleable functionalities, the game-based and the simulation-based notions of non-adaptive security coincide (just as it happens in regular encryption schemes).

1.1 Summary of Our Results

Generalized Key Delegation for Wildcarded Identity-Based and Inner-Product Encryption. One of the main applications of IBE and HIBE schemes is email encryption, where users can encrypt a message to the owner of the email address without having to obtain a certified copy of the owner's public key first. Motivated by the fact that many email addresses correspond to groups of users rather than single individuals, Abdalla *et al.* [ACD$^+$06] introduced the concept of identity-based cryptography with wildcards (WIBE, for short). In a WIBE scheme, decryption keys are issued exactly as in a standard HIBE scheme and the main difference lies in the encryption process. More specifically, in a WIBE scheme, the sender can encrypt messages to more general *patterns* consisting of identity strings and *wildcards* so that any identity matching the given pattern can decrypt. For instance, by encrypting a message to the pattern `UNISA.DI.*`, then any user belonging to the DI department at the University of Salerno would be able to decrypt it.

Unfortunately, like in standard HIBE schemes, the hierarchical key derivation of a WIBE scheme has its limitations. In particular, it does not allow any deviation from the hierarchical structure or prevent from further deriving keys below their identities. In order to overcome these limitations, Abdalla, Kiltz, and Neven [AKN07] introduced the concept of identity-based encryption with wildcard key derivation (WKD-IBE, for short) to generalize the key delegation mechanism of HIBE schemes. In a WKD-IBE primitive, which can be seen as the dual of a WIBE, secret keys are associated with *patterns* consisting of identity strings and *wildcards* and the owner of a key can derive keys for any identity that matches the pattern associated with this secret key. In addition to introducing the concept of WKD-IBE, Abdalla, Kiltz, and Neven also proposed several instantiations of the new primitive, both in the standard and random-oracle models, along with new applications such as identity-based broadcast encryption (IBBE, for short) and wildcard signatures. They also indicated that the concepts of WKD-IBE and WIBE could be combined into a universal primitive which allows for more general patterns to be used in both the encryption and key derivation algorithms. We refer to this WKD-IBE with generalized key delegation as WW-IBE. In Chapter 4 we show how to build a fully secure anonymous WW-IBE scheme.

Lattice-based Hierarchical Inner Product Encryption. In Chapter 5, we consider the problem of constructing hierarchical inner-product encryption scheme based on lattices assumptions. To achieve this goal, we extend the lattice-based IPE scheme by Agrawal *et al.* [AFV11] to the hierarchical setting by employing basis delegation technics by Peikert *et al.* [CHKP10] and by Agrawal *et al.* [ABB10]. As the underlying IPE scheme, our new scheme is shown to be weak selective secure based on the difficulty of the learning with errors problem in the standard model, as long as the total number of levels in the hierarchy is a constant.

Adaptive Simulation-Based Secure Constructions for Functional Encryption. In Chapter 6, we consider the problem of designing simulation-based secure functional encryption schemes. We look both at general functionalities (the functionality corresponding to the class of poly-sized circuits) and at a more specialized functionality.

Specifically, we give a general transformation that takes a game-based functional encryption scheme for all polynomial-size circuits and constructs a functional encryption scheme for the same functionality that is simulation-based secure against adaptive adversaries that can ask for a bounded number of tokens and can see one ciphertext. Our transformation is both black-box and for the standard model and is inspired by the work of Feige, Lapidot and Shamir [FLS90]. We remark that by the recent impossibility results of [AGVW12], the restriction to bounded number of tokens is necessary and thus our construction is essentially the best one can hope for.

Then, we show how to construct a HVE scheme whose simulation-based security can be proved under standard assumptions in the bilinear pairing setting in the standard model. Our construction is shown secure against *adaptive* adversaries obtaining *one* ciphertext and asking an *unbounded* number of tokens. Again, this is the best one can hope for in the standard model, given the impossibility result of [BSW11, BO12] for IBE. The only previous simulation-based construction for IBE secure against adaptive adversary was given in [BSW11] and is in the programmable random oracle model and imposes no bound on the number of tokens and ciphertexts obtained by the adversary.

Chapter 2

Functional Encryption

We start this chapter by describing the syntactic and security definition of functional encryption. To do so, we will follow the work of Boneh, Sahai, and Waters [BSW11] who initiated the formal study of functional encryption by giving precise definitions of the concept and its security.

Definition 2.0.1 [Functionality] A *functionality* $F = \{F_n\}$ is a family of functions $F_n : K_n \times X_n \rightarrow \Sigma$ where K_n is the *key space* for parameter n, X_n is the *plaintext space* for parameter n and Σ is the *output space*. Sometimes we will refer to functionality F as a function from $F : K \times X \rightarrow \Sigma$ with $K = \cup_n K_n$ and $X = \cup_n X_n$. For simplicity, and without loss of generality, we assume that $\Sigma = \{0, 1\}$.

An easy example of a functionality is that for the evaluation of Boolean circuits. Specifically, the $\mathsf{Circuit}$ functionality is defined as follow.

Definition 2.0.2 [$\mathsf{Circuit}$ Functionality] The $\mathsf{Circuit}$ functionality has key space K_n equals to the set of all n-input Boolean circuits and plaintext space X_n the set $\{0, 1\}^n$ of n-bit strings. For $C \in K_n$ and $x \in X_n$, we have

$$\mathsf{Circuit}(C, x) = C(x),$$

Then, informally, a functional encryption scheme for a functionality F defined over (K, X) enables the decryptor to evaluate $F(k, x)$ given the encryption of $x \in X$ and a secret key sk_k for $k \in K$.

Formally, a functional encryption scheme can be defined as follows.

Definition 2.0.3 [Functional Encryption Scheme] A *functional encryption* scheme for a functionality F defined over (K, X) is a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of 4 algorithms with the following syntax:

$\mathsf{Setup}(1^\lambda)$ outputs *public* and *master secret* keys (mpk, msk) for *security parameter* λ;

$\mathsf{KeyGen}(msk, k)$ on input a master secret key msk and *key* $k \in K$, outputs *token* sk_k;

$\mathsf{Enc}(mpk, x)$ on input public key mpk and *plaintext* $x \in X$, outputs *ciphertext* Ct;

$\mathsf{Dec}(mpk, sk_k, \mathsf{Ct})$ outputs a string y.

Correctness. We require that for all $k \in K$ and $x \in X$, and for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_k \leftarrow \mathsf{KeyGen}(msk, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk, x)$, then $\mathsf{Dec}(mpk, sk_k, \mathsf{Ct}) = F(k, x)$ with very high probability.

In the above definition the plaintext space and the key space only depend on the security parameter. In a more general definition both sets could also depend on the public key mpk as it is the case for most known candidate constructions of (functional and plain) public key encryption schemes. Nonetheless, since our results do not depend on this definitional detail, we choose not to overburden our notations and definitions.

In the next section, we will see two sub-classes of functional encryption well recognized in the literature. These sub-classes are characterized by having a plaintext space with additional structure.

2.1 Predicate Encryption

In the literature, many functional encryption schemes are defined for functionality whose plaintext space X consists of two subspaces I and M called respectively *index space* and *message space*. In this case, the functionality F is defined in terms of a polynomial-time predicate $P : K \times I \to \{0, 1\}$ as follows:

$$F(k, (\mathfrak{ind}, \mathfrak{m})) = \begin{cases} \mathfrak{m} & \text{if } P(k, \mathfrak{ind}) = 1 \\ \perp & \text{if } P(k, \mathfrak{ind}) = 0 \end{cases},$$

where $k \in K$, $\mathfrak{ind} \in I$ and $\mathfrak{m} \in M$.

Definition 2.1.1 [Predicate Encryption Schemes] A *predicate encryption* scheme for a functionality F defined over $(K, X = (I, M))$ for predicate $P : K \times I \to \{0, 1\}$ is a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ of 4 algorithms with the following syntax:

$\mathsf{Setup}(1^\lambda)$ outputs *public* and *master secret* keys (mpk, msk) for *security parameter* λ;

$\mathsf{KeyGen}(msk, k)$ on input a master secret key msk and *key* $k \in K$, outputs *token* sk_k;

$\mathsf{Enc}(mpk, (\mathfrak{ind}, \mathfrak{m}))$ on input public key mpk, an index $\mathfrak{ind} \in I$ and message $\mathfrak{m} \in M$, outputs ciphertext Ct;

$\mathsf{Dec}(mpk, sk_k, \mathsf{Ct})$ outputs a string $\mathfrak{m}$ or $\perp$.

Correctness. We require that for all $k \in K$ and $(\mathfrak{ind}, \mathfrak{m}) \in X$, and for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$, $sk_k \leftarrow \mathsf{KeyGen}(msk, k)$ and $\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk, (\mathfrak{ind}, \mathfrak{m}))$, then $\mathsf{Dec}(mpk, sk_k, \mathsf{Ct}) = \mathfrak{m}$ if and only if $P(k, \mathfrak{ind}) = 1$.

Predicate Only Encryption. A *predicate only encryption scheme* is a predicate encryption where the plaintext space coincides with the index space meaning that the ciphertext does not contain any message. More formally, in this case the functionality F is defined as follows:

$$F(k, \mathfrak{ind}) = P(k, \mathfrak{ind}) \in \{0, 1\},$$

where $k \in K$, $\mathfrak{ind} \in I$.

Predicate Encryption with Public Index. In a *predicate encryption scheme with public index*, the ciphertext always leaks the index. More formally, in this case the functionality F is defined as follows:

$$F(k, (\mathfrak{ind}, \mathfrak{m})) = \begin{cases} (\mathfrak{ind}, \mathfrak{m}) & \text{if } P(k, \mathfrak{ind}) = 1 \\ (\mathfrak{ind}, \perp) & \text{if } P(k, \mathfrak{ind}) = 0 \end{cases},$$

where $k \in K$, $\mathfrak{ind} \in I$ and $\mathfrak{m} \in M$.

2.1.1 Identity-Based Encryption

In an IBE scheme, ciphertexts and private keys are associated with identities, or strings, and a secret key can decrypt a ciphertext if they encode the same identity. An IBE scheme can then be formally defined as a predicate encryption scheme with public index in the following way. The keys space K and the index space I are the sets of binary strings and the IBE predicate P over (K, I) is defined as follows:

$$P(k, \mathfrak{ind}) = \begin{cases} 1 & \text{if } k = \mathfrak{ind} \\ 0 & \text{if } k \neq \mathfrak{ind} \end{cases}.$$

If the index is hidden by the ciphertext then we recover the definition of Anonymous IBE.

Hierarchical IBE. Following the work of Boneh and Franklin, Horwitz and Lynn [HL02] proposed the concept of Hierarchical IBE (HIBE, for short), in which users are organized in a tree of depth L, with the root being the trusted central authority.

An HIBE scheme for hierarchy of depth L can then be formally defined as a predicate encryption scheme with public index in the following way. The keys space K and the index space I are the sets of vectors $(\mathsf{ID}_1, \ldots, \mathsf{ID}_\ell)$ where $\ell \leq L$ and each $\mathsf{ID}_i \in \{0,1\}^*$. Then, the HIBE predicate P_L over (K, I) is defined as follows:

$$P_L(k = (\mathsf{ID}_1^k, \ldots, \mathsf{ID}_\ell^k), \mathfrak{ind} = (\mathsf{ID}_1^c, \ldots, \mathsf{ID}_{\ell'}^c)) = \begin{cases} 1 & \text{if } \ell \leq \ell' \text{ and for } i = 1 \ldots \ell, \mathsf{ID}_i^k = \mathsf{ID}_i^c \\ 0 & \text{otherwise} \end{cases}.$$

If the index is hidden by the ciphertext then we recover the definition of Anonymous HIBE.

2.1.2 Hidden Vector Encryption

An HVE scheme, first proposed by Boneh and Waters [BW07], is a generalization of IBE with the support of a special character $\star$ we refer to as a *wildcard* character. Formally, an HVE scheme can be defined as a predicate encryption scheme in the following way. The keys space K consists of all the ℓ-tuple $(v_1, \ldots, v_\ell)$ where each $v_i \in \{0,1\}^* \cup \{\star\}$ and the index space I consists of all the ℓ-tuple $(w_1, \ldots, w_\ell)$ of ℓ where each $w_i \in \{0,1\}^*$ and the HVE predicate P_ℓ over (K, I) is defined as follows:

$$P_\ell(k = (v_1, \ldots, v_\ell), \mathfrak{ind} = (w_1, \ldots, w_\ell)) = \begin{cases} 1 & \text{if } v_i = w_i \text{ whenever } v_i \neq \star \\ 0 & \text{otherwise} \end{cases}.$$

Applications of such encryption scheme include conjunctive and range searches.

2.1.3 Inner Product Predicate

Katz, Sahai and Waters [KSW08] proposed inner-product encryption (IPE, for short), a more general system to test if a dot product operation over the ring $\mathbb{Z}_N$ is equal to 0, where N is the product of three random primes chosen by the setup algorithm. Later, Okamoto and Takashima [OT09] and Lewko *et. al.* [LOS$^+$10] gave constructions over the field F_p. Formally, an IPE scheme can be defined as a predicate encryption scheme in the following way. Let p be a prime of length λ, chosen at setup time where λ is the security parameter. Then, the keys space K and the index space I consist of all the n-tuple $(v_1, \ldots, v_n)$, where each $v_i \in F_p$, and the IPE predicate $P_{p,n}$ over (K, I) is defined as follows:

$$P_{p,n}(k = (v_1, \ldots, v_n), \mathfrak{ind} = (w_1, \ldots, w_n)) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} v_i \cdot w_i = 0 \\ 0 & \text{otherwise} \end{cases} .$$

Despite this apparently restrictive structure, IPE can support conjunction, subset and range queries on encrypted data as well as disjunction, polynomial evaluation, and CNF and DNF formulas.

Hierarchical IPE. Following the work of Katz, Sahai and Waters, Okamoto and Takashima [OT09] proposed the concept of Hierarchical IPE (HIPE, for short). Specifically,

Definition 2.1.2 [Hierarchical Format] Let $\boldsymbol{\mu}$ be a tuple of positive integers $\boldsymbol{\mu} = (n, d; \mu_1, \ldots, \mu_d)$ such that $\mu_0 = 0 < \mu_1 < \cdots < \mu_d = n$. For $i = 1, \ldots, d$, let $\Sigma_i = \mathbb{F}_p^{\mu_i - \mu_{i-1}} \setminus \{\mathbf{0}_{\mu_i - \mu_{i-1}}\}$ the set of attributes. Let Σ be the hierarchical attributes $\Sigma = \cup_{i=1}^{d} (\Sigma_1 \times \ldots \times \Sigma_i)$, where the union is a disjoint union. Them, we call $\boldsymbol{\mu}$ an *hierarchical format* of depth d for the attribute space Σ. Sometimes, we will use notation $\Sigma_{|t} = \cup_{i=1}^{t} (\Sigma_1 \times \ldots \times \Sigma_i)$

Let p be a prime of length λ, chosen at setup time where λ is the security parameter. Then, an HIPE scheme can be formally defined as a predicate encryption scheme in the following way. For a given hierarchical format $\boldsymbol{\mu}$, the keys space K and the index space I correspond to the hierarchical attributes set Σ, and the HIPE predicate $P_{p,\boldsymbol{\mu}}$ over (K, I) is defined as follows:

$$P_{p,\boldsymbol{\mu}}(k = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_\ell), \mathfrak{ind} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{\ell'})) = \begin{cases} 1 & \text{if } \ell \le \ell' \text{ and for each } i,\ \langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0 \\ 0 & \text{otherwise} \end{cases} .$$

2.2 Security Definitions

One of the most important achievements of the modern theory of Cryptography is the formalization of the notion of a *secure cryptosystem* given in the seminal work of Goldwasser and Micali [GM84]. There, two notions of security were proposed, one is a *game-based* notion and the other is *simulation-based*, and they were shown to be equivalent in the sense that an encryption scheme meets one security definition if and only if it meets the other. The study of simulation-based notions of security for functional encryption was initiated by Boneh, Sahai, and Waters [BSW11] and O'Neill [O'N10]. Specifically, in [BSW11] Boneh *et al.* showed that there exists a clearly insecure functional encryption scheme that is nonetheless deemed secure by the game-based notion of security. This motivated the study of a simulation-based notion of security for functional encryption. In the same paper [BSW11], Boneh *et al.* gave impossibility results in the non-programmable random oracle model for simple instances of functional encryption for which, instead, game-based security had been shown to be achievable (again, under appropriate complexity assumptions).

In the next sections, we give formal definitions of game-based and simulation-based security for functional encryption.

2.2.1 Game-based security

The game-based notion of security for functional encryption, also called indistinguishability under-chosen message attack (IND-CPA, for short), can be formalized by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Let $\mathcal{FE}$ be a functional encryption scheme for functionality F over (K, X) and let $\mathcal{A}$ be a probabilistic polynomial-time adversary. Game IND-CPA$_{\mathcal{A}}^{\mathcal{FE}}(1^\lambda)$, where λ is the security parameter, is defined as follows.

Setup: $\mathcal{C}$ generates public key and master secret key by invoking the setup algorithm on input the security parameter λ given in unary. Specifically, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$. Then, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input mpk.

Query Phase 1: $\mathcal{A}$ adaptively submits key queries. On input a key $k \in K$, $\mathcal{C}$ generates a secret key for k, $sk_k = \mathsf{KeyGen}(msk, k)$, and gives sk_k back to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two equal length plaintexts $x_0, x_1 \in X$. $\mathcal{C}$ chooses random $\eta \in \{0, 1\}$ and generates ciphertext $\mathsf{Ct}^\star = \mathsf{Enc}(mpk, x_\eta)$. $\mathcal{C}$ sends back $\mathsf{Ct}^\star$ to $\mathcal{A}$.

Query Phase 2: It is the same as Query Phase 1.

Guess: Eventually, $\mathcal{A}$ submits its guess η'.

Winning Condition: $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. $\eta = \eta'$.
2. For all k queried by $\mathcal{A}$, it holds that $F(k, x_0) = F(k, x_1)$.

The *advantage* of $\mathcal{A}$ in the above game is defined as

$$\mathbf{Adv}_{\text{IND-CPA}}^{\mathcal{FE},\mathcal{A}}(\lambda) = \text{Prob}[\text{IND-CPA}_{\mathcal{A}}^{\mathcal{FE}}(1^\lambda) = 1] - 1/2 \ .$$

Definition 2.2.1 A functional encryption scheme $\mathcal{FE}$ is IND-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the IND-CPA game described above, *i.e.,* $\mathbf{Adv}_{\text{IND-CPA}}^{\mathcal{FE},\mathcal{A}}(\lambda)$ is negligible function of λ.

CCA Security. In a chosen-ciphertext attack (IND-CCA, for short), the adversary is additionally given access to a decryption oracle that for a given key $k \in K$ and a given ciphertext Ct returns the decryption $\mathsf{Dec}(mpk, \mathsf{KeyGen}(msk, k), \mathsf{Ct})$.

Selective Security. It is a weaker security model, introduced by [CHK03, BB04] to obtain the first IBE scheme secure in the standard model, where the adversary must commit to its challenge before seeing the public key. We refer to this definition of security as *selective-indistinguishability under chosen-message attack* (*chosen-ciphertext-attack*), sIND-CPA (sIND-CCA) for short. In the literature, the non-selective model of security (IND-CPA) is also sometimes called *full-security* model.

Predicate Encryption. In this case, Definition 2.2.1 is sufficient. In particular in the IND-CPA game, the adversary's challenge consists of two indexes $\mathfrak{ind}_0, \mathfrak{ind}_1$ and two messages m_0, m_1. This models the privacy of the index and that of the message. This definition of security is also know in the literature as *anonymity under chosen-plaintext attack* (ANO-CPA, for short). Moreover, in the literature a weaker security model has been defined. It is known as *weakly attribute-hiding* model (wAH-IND-CPA or wAH-ANO-CPA for short) where the winning conditions is relaxed. Specifically, it is required that, for all k queried by $\mathcal{A}$, it holds that $F(k, x_0) = F(k, x_1) = 0$, meaning that the adversary cannot see secret keys that can be used to decrypt the ciphertext. This relaxation is still reasonable and has been used to achieve full-security.

Instead, for public index predicate encryption schemes it is enough to model only the privacy of the message because the index is public. Thus in the IND-CPA game, the adversary's challenge consists of an index $\mathfrak{ind}$ and two messages m_0, m_1. In this case, the weakly attribute-hiding model is a minimum requirement because otherwise the adversary would be forced to submit a challenge with $m_0 = m_1$.

2.2.2 Simulation-based security

In the context of functional encryption, the intuition that we would like to be captured by a simulation-based definition of security is that getting the secret key sk_k corresponding to the key $k \in K$ should only reveal $F(k, x)$ when given an encryption of x. The following formal definition, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10], tries to capture this intuition avoiding to fall into the impossibility results of [BSW11], namely we restrict the adversary to request the encryption of a single message.

Definition 2.2.2 A functional encryption scheme $\mathcal{FE}$ for functionality F over (K, X) is *adaptively simulation-based secure* (SIM-Secure, for short) if there exists a *simulator* algorithm $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all *adversary* algorithms $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ the output of the following two experiments are computationally indistinguishable.

```
$\mathsf{RealExp}^{\mathcal{FE},\mathcal{A}}(1^\lambda)$

$(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(x, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk, x)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(msk,\cdot)}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, x, \alpha)$
```

```
$\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathcal{FE},\mathcal{A}}(1^\lambda)$

$(mpk, msk) \leftarrow \mathsf{Sim}_0(1^\lambda)$;
$(x, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$(\mathsf{Ct}', \mathsf{aux}') \leftarrow \mathsf{Sim}_1(mpk, |x|, (k_i, F(k_i, x), sk_i)_{i=1}^{q_1})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathcal{O}(msk,\mathsf{aux}',\cdot)}(mpk, \mathsf{Ct}', \mathsf{aux})$;
**Output:** $(mpk, x, \alpha)$
```

Here, $k_1, \dots, k_{q_1}$ are the keys for which $\mathcal{A}_0$ has asked to see a secret key in the first query phase for some $q_1 = \mathrm{poly}(\lambda)$. Moreover, oracle $\mathcal{O}(msk, \mathsf{aux}', \cdot)$ is algorithm $\mathsf{Sim}_2(msk, \mathsf{aux}', \cdot, F(\cdot, x))$ that answers to the adversary's queries of the second phase. In particular, Sim_2 receives as third argument the key k for which the adversary has asked a secret key and as fourth argument the value $F(k, x)$, along with the master secret key msk and current state aux' of the simulator.

We observe that the simulator algorithm Sim_2 is stateful in that after each invocation, it updates the state aux' which is carried over to its next invocation. Also, we note that the notion of *non-adaptive* SIM-Security (NA-SIM-Secure, for short), is obtained by giving $\mathcal{A}_1$ no oracle access in both experiments.

Recently, Agrawal *et al.* [AGVW12] showed that for adversaries that can obtain an unbounded number of tokens, Definition 2.2.2 cannot be achieved for *incompressible* functionalities (i.e., weak pseudo-random functions). Thus, by restricting the range of adversaries to those that make an a-priori bounded number of queries (counting both first stage and second stage token queries), we obtain the following weaker notion.

Definition 2.2.3 A functional encryption scheme $\mathcal{FE}$ is *q-bounded simulation-based secure* (q-SIM-Secure, for short) if there exists a *simulator* algorithm Sim such that for all *adversary* algorithms $\mathcal{A}$ that make at most q token queries, the output of experiments $\mathsf{RealExp}^{\mathcal{FE},\mathcal{A}}$ and $\mathsf{IdealExp}_{\mathsf{Sim}}^{\mathcal{FE},\mathcal{A}}$ are indistinguishable.

Chapter 3

Tools and Notations

In this chapter we introduce the mathematical tools we will use in our constructions.

3.1 Notations

We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n.

If D is a probability distribution, the writing "$x \leftarrow D$" means that x is chosen according to D. If D is a finite set, the writing "$x \leftarrow D$" means that x is chosen according to uniform probability on D. If A is an algorithm then $A(x_1, x_2, \ldots)$ denotes the probability distribution of the output of A when A is run on input $(x_1, x_2, \ldots)$ and randomly chosen coin tosses. Instead $A(x_1, x_2, \ldots; R)$ denotes the output of A when run on input $(x_1, x_2, \ldots)$ and (sufficiently long) coin tosses R. All algorithms, unless explicitly noted, are probabilistic polynomial time (PPT, for short) and all adversaries are modeled by non-uniform polynomial time algorithms. If A is an algorithm and B is an algorithm with access to an oracle then A^B denotes the execution of A with oracle access to B. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$.

If $q > 0$ is an integer the $[q]$ denotes the set $\{1, \ldots, q\}$. The notation $\lfloor x \rceil$ denotes the nearest integer to x, rounding towards 0 for half-integers. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q and we represent $\mathbb{Z}_q$ as integers in $(q/2, q/2]$. We let $\mathbb{Z}_q^{n \times m}$ denote the set of $n \times m$ matrices with entries in $\mathbb{Z}_q$.

We use bold capital letters (e.g. $\boldsymbol{A}$) to denote matrices, bold lowercase letters (e.g. $\boldsymbol{w}$) to denote vectors. The notation $\boldsymbol{A}^\top$ denotes the transpose of the matrix $\boldsymbol{A}$. When we say a matrix defined over $\mathbb{Z}_q$ has *full rank*, we mean that it has full rank modulo each prime factor of q. If $\boldsymbol{A}_1$ is an $n \times m$ matrix and $\boldsymbol{A}_2$ is an $n \times m'$ matrix, then $[\boldsymbol{A}_1 \| \boldsymbol{A}_2]$ denotes the $n \times (m + m')$ matrix formed by concatenating $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. If $\boldsymbol{w}_1$ is a length m vector and $\boldsymbol{w}_2$ is a length m' vector, then we let $[\boldsymbol{w}_1 \| \boldsymbol{w}_2]$ denote the length $(m + m')$ vector formed by concatenating $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. However, when doing matrix-vector multiplication we always view vectors as column vectors. The norm of a matrix $\boldsymbol{R} \in \mathbb{R}^{k \times m}$ is defined as $\|\boldsymbol{R}\| := \sup_{\|\boldsymbol{u}\|=1} \|\boldsymbol{R}\boldsymbol{u}\|$.

3.2 Bilinear Settings

The study of cyclic groups in cryptography has been motivated by the assumed hardness of taking discrete logarithms on which the security of several cryptosystems (most notably the ElGamal encryption scheme) and cryptographic protocols has been based. The observation that some cyclic groups admit bilinear maps has been used at first to break cryptosystems, see [MOV91, FMR99, FR94]. Later, it was discovered that they could also be used to build cryptosystems, so rather than avoiding this additional properties of the cyclic groups, one can exploit them to construct new schemes. The Boneh and Franklin's identity-based encryption scheme [BF01] is one of the most famous examples of what can be achieved using bilinear maps. Extending the basic idea leads to identity-based schemes with additional useful properties such as authenticated or hierarchical identity-based encryption [HL02]. Moreover identity-based signature [BLS01, Hes03] identity-based identification [KH04, KH06] schemes have been constructed using bilinear maps.

Definition 3.2.1 [Bilinear Setting] Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime-order N. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. A *bilinear pairing* or *bilinear map* is an efficiently computable function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that:

1. (*Bilinearity*) $\forall\, a, b \in \mathbb{Z}_N$ it holds that $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$.

2. (*Non-degeneracy*) $\hat{e}(g_1, g_2) \neq 1$.

Then, we call the tuple $(N, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, \hat{e})$ an *asymmetric bilinear setting*. Moreover, if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and g is a generator of $\mathbb{G}$ then the tuple $(N, \mathbb{G}, \mathbb{G}_T, g, \hat{e})$ is called *symmetric bilinear setting*.

Finally, we suppose the existence of an efficient algorithm BSGen which takes in input the security parameter λ and outputs a description of a bilinear setting over cyclic groups whose order is of $\Theta(\lambda)$ bits.

The cryptographic relevance of bilinear maps stem from the fact that in cyclic groups that admit such a mapping the decisional Diffie-Hellman assumption does not hold. Indeed, given g, g^x, g^y, g^z it is possible to check if $z = xy$ (and thus solve the decisional DH problem) by testing $\hat{e}(g, g^z)$ and $\hat{e}(g^x, g^y)$ for equality.

Decisional Linear Assumption. One of the most well studied assumption in the contest of prime-order bilinear settings is the *Decisional Linear Assumption* (DL, for short). It can be described in the following way.

For a given prime-order bilinear setting generator BSGen, define the following distribution: Pick a random bilinear setting $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BSGen}(1^\lambda)$, and then pick

$$g, f, v \leftarrow \mathbb{G} \text{ and } c_1, c_2, c_3 \in \mathbb{Z}_N$$

and set $D = (N, g, f, v, f^{c_1}, v^{c_2})$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking the DL Assumption is

$$\mathbf{Adv}_{\mathsf{DL}}^{\mathcal{A}}(\lambda) = \left|\text{Prob}[\mathcal{A}(D, g^{c_1+c_2}) = 1] - \text{Prob}[\mathcal{A}(D, g^{c_1+c_2+c_3}) = 1]\right| .$$

Definition 3.2.2 [Decisional Linear Assumption] We say that the DL Assumption holds for generator BSGen if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_{\mathsf{DL}}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

3.2.1 Composite-Order Bilinear Setting

Composite-order bilinear settings were first used in cryptography by [BGN05] (see also [Bon07]) and then widely used to construct functional encryption schemes and more.

Definition 3.2.3 [Composite-Order Bilinear Setting] Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of composite order N product of $m > 1$ primes, $N = p_1 \cdot \ldots \cdot p_m$. Let g be a generator of $\mathbb{G}$ and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ an efficiently computable non-degenerate bilinear map. Then, we call the tuple $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e})$ a *composite-order bilinear setting*.

Moreover, we suppose the existence of an efficient algorithm CBSGen which takes as input the security parameter λ and the number of primes $m > 1$, and outputs a description of a composite-order bilinear setting over cyclic groups whose order N is the product of m primes of $\Theta(\lambda)$ bits each.

For notational convenience, for each non-empty subset $S \subseteq [m]$ we denote by $\mathbb{G}_S$ the subgroup of order $\prod_{i \in S} p_i$ in the bilinear group $\mathbb{G}$.

Orthogonality. From the fact that the group is cyclic, it is easy to verify that if g and h are group elements of co-prime orders then $\hat{e}(g, h) = 1$. This is called the *orthogonality* property and is a crucial tool in many proof of full security for functional encryption schemes.

General Subgroup Decision Assumption. In the contest of composite-order bilinear settings, one of the most common assumption is the *General Subgroup Decision Assumption* (GSD, for short) introduced by Bellare *et al.* [BWY11]. It can be described in the following way.

For a given composite-order bilinear setting generator CBSGen, define the following distribution: Pick a random bilinear setting $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 1^m)$. Let $S_0, S_1, S_2, \ldots, S_k$, for constant k, be non-empty subsets of $[m]$ such that for each $2 \le j \le k$, either $S_j \cap S_0 = S_j \cap S_1 = \emptyset$ or $S_j \cap S_0 \neq \emptyset$ and $S_j \cap S_1 \neq \emptyset$, and then pick

$$Z_2 \leftarrow \mathbb{G}_{S_2}, \ldots, Z_k \leftarrow \mathbb{G}_{S_k} \ ,$$

$$T_0 \leftarrow \mathbb{G}_{S_0},\ T_1 \leftarrow \mathbb{G}_{S_1} \ ,$$

and set $D = (N, Z_2, \ldots, Z_k)$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking the GSD Assumption is

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{GSD}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_0) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_1) = 1]| \ .$$

Definition 3.2.4 [General Subgroup Decision Assumption] We say that the GSD Assumption holds for generator CBSGen if, for all $S_0, \ldots, S_k$ that satisfy the conditions above and for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{GSD}}(\lambda)$ is a negligible function of λ.

We will use the following algorithm to sample a random instance of the GSD Assumption.

Algorithm GSDGen on input the security parameter 1^λ, the number of primes 1^m and $k+1$ non-empty subsets of $[m]$, $S_0, S_1, S_2, \ldots, S_k$, does the following: Pick a random composite-order bilinear setting $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 1^m)$, and then pick

$$Z_2 \leftarrow \mathbb{G}_{S_2}, \ldots, Z_k \leftarrow \mathbb{G}_{S_k} \ ,$$

$$T_0 \leftarrow \mathbb{G}_{S_0},\ T_1 \leftarrow \mathbb{G}_{S_1} \ .$$

Then the algorithm picks a random $\beta \in \{0, 1\}$ and returns the tuple $(N, T_\beta, Z_2, \ldots, Z_k)$.

For notational convenience, sometimes we will omit the parameter m when it is clear from the context.

3.2.2 Dual Pairing Vector Spaces

Dual paring vector spaces (DPVS, for short) are rich mathematical structures introduced by Okamoto and Takashima [OT08, OT09] in the context of functional encryption. DPVS can be constructed on prime-order symmetric and asymmetric bilinear settings. We will focus on the symmetric setting.

Given a symmetric bilinear setting $(N, \mathbb{G}, \mathbb{G}_\mathrm{T}, g, \hat{e})$, let $\mathbb{V} = \mathbb{G}^n$ be a *vector space* whose elements are expressed by vectors

$$\boldsymbol{x} = (g^{x_1}, \dots, g^{x_n}) ,$$

where $x_i \in \mathbb{Z}_N$ for each $i \in [n]$. The *canonical base* of $\mathbb{V}$ is

$$\mathbb{A} = (\boldsymbol{a}_i)_{i \in [n]} ,$$

where $\boldsymbol{a}_i = (1, \dots, 1, g, 1, \dots, 1)$. Thus, a vector $\boldsymbol{x}$ can be expressed in base $\mathbb{A}$ as:

$$\boldsymbol{x} = (x_1, \dots, x_n)_\mathbb{A} = \sum_{i \in [n]} x_i \cdot \boldsymbol{a}_i .$$

Then, a well-defined non-degenerate *pairing operation* between $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{V}$, can be defined as follows:

$$\hat{e}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i \in [n]} \hat{e}(g^{x_i}, g^{y_i}) = \hat{e}(g, g)^{\sum_{i \in [n]} x_i \cdot y_i} .$$

Notice that over $\mathbb{A}$ it is easy to decompose $x_i \boldsymbol{a}_i = (1, \dots, 1, g_i^x, 1, \dots, 1)$ from $\boldsymbol{x} = (x_1, \dots, x_n)_\mathbb{A} = \sum_{i \in [n]} x_i \cdot \boldsymbol{a}_i$. Thus, the canonical base $\mathbb{A}$ can be further randomized to obtain a new base $\mathbb{B} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ of $\mathbb{V}$ by applying a random (regular) linear transformation $\boldsymbol{R} = (r_{i,j})_{i \in [n], j \in [n]} \leftarrow \mathsf{GL}(n, \mathbb{Z}_N)$ as follows:

$$\boldsymbol{b}_i = \sum_{j \in [n]} r_{i,j} \cdot \boldsymbol{a}_j .$$

Moreover, if we apply to $\mathbb{A}$ randomization $\boldsymbol{R}$ to form base $\mathbb{B}$ and randomization $(\boldsymbol{R}^\mathsf{T})^{-1}$ to form base $\mathbb{B}^\star = (\boldsymbol{b}_1^\star, \dots, \boldsymbol{b}_n^\star)$. Then $(\mathbb{B}, \mathbb{B}^\star)$ are dual orthonormal bases of $\mathbb{V}$ such that $\hat{e}(\boldsymbol{b}_i, \boldsymbol{b}_j^\star) = \hat{e}(g, g)^{\delta_{i,j}}$ where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise.

Definition 3.2.5 [Dual Pairing Vector Spaces] Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order N. Let g be a generator of $\mathbb{G}$ and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ an efficiently computable non-degenerate bilinear map. Let $\mathbb{V} = \mathbb{G}^n$, for integer $n > 0$, be a *vector space* obtained as direct product of $\mathbb{G}$ and $\mathbb{A} = (\boldsymbol{a}_i)_{i \in [n]}$ its canonical base, and let $(\mathbb{B}, \mathbb{B}^\star)$ random dual orthonormal bases of $\mathbb{V}$. Let $\mathsf{param}_\mathbb{V} = (N, \mathbb{V}, \mathbb{G}_T, \mathbb{A})$. Then, we call the tuple $(\mathsf{param}_\mathbb{V}, \mathbb{B}, \mathbb{B}^\star)$ a *dual pairing vector space.*

Finally, we suppose the existence of an efficient algorithm $\mathsf{DPVSGen}$ which takes in input the security parameter λ and integer n, and outputs a description of a dual pairing vector space over cyclic groups whose order is of $\Theta(\lambda)$ bits. Notice that algorithm $\mathsf{DPVSGen}$ can be obtained by using algorithm BSGen and applying the randomization described above.

Decisional Subspace Problem. First introduced by [OT08]. It asks to distinguish between vector $\boldsymbol{u} = (r_1, \dots, r_m)_\mathbb{B}$ and vector $\boldsymbol{v} = (\boldsymbol{0}_n, r_{n+1}, \dots, r_m)_\mathbb{B}$ where $n + 1 < m$ and $r_i \leftarrow \mathbb{Z}_N$ for $i \in [m]$. Although the decisional subspace problem is assumed to be intractable, it can be efficiently solved by using *trapdoor* $\boldsymbol{t}^\star \in \mathsf{span}(\boldsymbol{b}_1^\star, \dots, \boldsymbol{b}_n^\star)$. In fact $\hat{e}(\boldsymbol{u}, \boldsymbol{t}^\star) \neq 1$ and $\hat{e}(\boldsymbol{v}, \boldsymbol{t}^\star) = 1$ with high probability.

3.3 Lattices

In 1994, Peter Shor [Sho96] showed that quantum computers can solve in polynomial time, integer factorization and discrete logarithms including elliptic curve discrete logarithms [BL95]. Because of the threat that Shor's algorithm poses to existing encryption techniques, there is a great deal of interest in new hard problems believed to resist against quantum attacks.

Lattice assumptions are hard assumptions that have received in the very recent years a lot of attentions both in cryptanalysis and cryptography. In this section, we will introduce the background relevant to this work and the main hardness assumption in the field, the *Learning with Error* problem formally introduced by Regev [Reg05]. Specifically, we collect, almost verbatim, results from [ABB10, AFV11, AP09, MR04, GPV08, CHKP10].

Lattice. An m-dimensional lattice Λ is a full-rank discrete subgroup of $\mathbb{R}^m$. A basis of Λ is a linearly independent set of vectors whose span is Λ. We will focus on integer lattices and among these we will focus on the q-ary lattices defined as follows: for any integer $q \geq 2$ and any $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, we define

$$\begin{aligned}
\Lambda_q^{\perp}(\boldsymbol{A}) &:= \{\boldsymbol{e} \in \mathbb{Z}_m : \boldsymbol{A} \cdot \boldsymbol{e} = 0 \mod q\} \\
\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) &:= \{\boldsymbol{e} \in \mathbb{Z}_m : \boldsymbol{A} \cdot \boldsymbol{e} = \boldsymbol{u} \mod q\} \\
\Lambda_q(\boldsymbol{A}) &:= \{\boldsymbol{e} \in \mathbb{Z}_m : \exists\, \boldsymbol{s} \in \mathbb{Z}_q^m \text{ with } \boldsymbol{A}^t \cdot \boldsymbol{s} = \boldsymbol{e} \mod q\}.
\end{aligned}$$

The lattice $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ is a coset of $\Lambda_q^{\perp}(\boldsymbol{A})$; namely, $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \Lambda_q^{\perp}(\boldsymbol{A}) + \boldsymbol{t}$ for any $\boldsymbol{t}$ such that $\boldsymbol{A} \cdot \boldsymbol{t} = \boldsymbol{u} \mod q$.

Gram-Schmidt norm. Let $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k\}$ be a set of vectors in $\mathbb{R}^m$. Let $\|\boldsymbol{S}\|$ denotes the length of the longest vector in $\boldsymbol{S}$, i.e., $\max_{1 \leq i \leq k} \|\boldsymbol{s}_i\|$, and $\widetilde{\boldsymbol{S}} := \widetilde{\boldsymbol{s}}_1, \ldots, \widetilde{\boldsymbol{s}}_k \subset \mathbb{R}^m$ denotes the *Gram-Schmidt orthogonalization* of the vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k$. We refer to $\|\widetilde{\boldsymbol{S}}\|$ as the *Gram-Schmidt norm* of $\boldsymbol{S}$.

Gaussian distributions. Let L be a discrete subset of $\mathbb{Z}^n$. For any vector $\boldsymbol{c} \in \mathbb{R}^n$ and any positive parameter $\sigma \in \mathbb{R}_{>0}$, let

$$\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w}) := \exp\left(-\pi\|x - \boldsymbol{c}\|^2/\sigma^2\right)$$

be the Gaussian function on $\mathbb{R}^n$ with center $\boldsymbol{c}$ and parameter σ. Let

$$\rho_{\sigma,\boldsymbol{c}}(L) := \sum_{\boldsymbol{w} \in L} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w})$$

be the discrete integral of $\rho_{\sigma,\boldsymbol{c}}$ over L, and let $\mathcal{D}_{L,\sigma,\boldsymbol{c}}$ be the discrete Gaussian distribution over L with center $\boldsymbol{c}$ and parameter σ. Specifically, for all $\boldsymbol{v} \in L$, we have

$$\mathcal{D}_{L,\sigma,\boldsymbol{c}}(\boldsymbol{v}) = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{v})}{\rho_{\sigma,\boldsymbol{c}}(L)}\,.$$

For notational convenience, $\rho_{\sigma,0}$ and $\mathcal{D}_{L,\sigma,0}$ are abbreviated as ρ_σ and $\mathcal{D}_{L,\sigma}$ respectively. The following lemma captures standard properties of these distributions.

Lemma 3.3.1 Let $q \geq 2$ and let $\boldsymbol{A}$ be a matrix in $\mathbb{Z}_q^{n \times m}$ with $m > n$. Let $\boldsymbol{T_A}$ be a basis for $\Lambda_q^{\perp}(\boldsymbol{A})$ and $\sigma \geq \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega(\sqrt{\log m})$. Then for $\boldsymbol{c} \in \mathbb{R}^m$ and $\boldsymbol{u} \in \mathbb{Z}_q^n$:

1. $\Pr\left[\|\boldsymbol{w}-\boldsymbol{c}\| > \sigma\sqrt{m} \ : \ \boldsymbol{w} \leftarrow \mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}\right] \leq \operatorname{negl}(n)$

2. A set of $O(m \log m)$ samples from $\mathcal{D}_{\Lambda_q^{\perp}(A),\sigma}$ contains a full rank set in $\mathbb{Z}^m$, except with negligible probability.

3. There is a PPT algorithm $\mathsf{SampleGaussian}(\boldsymbol{A}, \boldsymbol{T_A}, \sigma, \boldsymbol{c})$ that returns $\boldsymbol{x} \in \Lambda_q^{\perp}(\boldsymbol{A})$ drawn from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}$.

4. There is a PPT algorithm $\mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{T_A}, \boldsymbol{u}, \sigma)$ that returns $\boldsymbol{x} \in \Lambda_q^{\perp}(A)$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$, whenever $\Lambda_q^{\boldsymbol{u}}(A)$ is not empty.

Norm of a random matrix. The following lemmata can be used to bound the norm of a random matrix in $\{-1,1\}^{m\times m}$.

Lemma 3.3.2 ([ABB10, Lemma 15]) Let $\boldsymbol{R}$ be a $k\times m$ matrix chosen at random from $\{-1,1\}^{k\times m}$. Then,

$$\Pr\left[\|\boldsymbol{R}\| > 12\sqrt{k+m}\right] < e^{-(k+m)} .$$

Lemma 3.3.3 ([ABB10, Lemma 16]) Let $\boldsymbol{u} \in \mathbb{R}^m$ be some vector of norm 1. Let $\boldsymbol{R}$ be a $k \times m$ matrix chosen at random from $\{-1,1\}^{k\times m}$. Then

$$\Pr\left[\|\boldsymbol{Ru}\| > \sqrt{k}\omega(\sqrt{\log k})\right] < \operatorname{negl}(k) .$$

Randomness extraction. We will use the following lemma, which follows from a generalization of the leftover hash lemma due to Dodis *et al.* [DRS04]. Agrawal, Boneh, and Boyen [ABB10] prove the lemma for prime moduli q but that the result extends to square-free values of q by the Chinese remainder theorem.

Lemma 3.3.4 [[ABB10, Lemma 13]] Suppose that $m > (n+1)\lg q + \omega(\log n)$ and that $q > 2$ is square free. Let $\boldsymbol{R}$ be an $m \times k$ matrix chosen uniformly in $\{1,-1\}^{m\times k} \mod q$ where $k = k(n)$ is polynomial in n. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be matrices chosen uniformly in $\mathbb{Z}^{n\times m}$ and $\mathbb{Z}^{n\times k}$ respectively. Then for all vectors $w \in \mathbb{Z}^m$, the distribution $(\boldsymbol{A}, \boldsymbol{AR}, \boldsymbol{R}^{\top} w)$ is statistically close to the distribution $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}^{\top} w)$.

Lattice problems. Two of the most well known computational problems on lattices are the *shortest vector problem* and the *shortest independent vectors problem.* Specifically, let $\lambda_1(\Lambda)$ denote the length of the shortest nonzero vector in the lattice Λ and $\lambda_n(\Lambda)$ the minimum length of a set of n linearly independent vectors from Λ, where the length of a set is defined as the length of longest vector in it, then the shortest vector problem and the shortest independent vectors problem are defined as follows:

Definition 3.3.5 [Shortest Vector Problem, GapSVP$_\gamma$] GapSVP is a promise problem. Specifically, an instance of GapSVP$_\gamma$, for $\gamma = \gamma(n) > 1$, is given by an n-dimensional lattice Λ, and a number $d > 0$. In *YES* instances, $\lambda_1(\Lambda) \leq d$ whereas in *NO* instances $\lambda_1(\Lambda) > \gamma(n) \cdot d$.

Definition 3.3.6 [Shortest Independent Vectors Problem, SIVP$_\gamma$] An instance of SIVP$_\gamma$, for $\gamma = \gamma(n) > 1$, is given by an n-dimensional lattice Λ. The goal is to output a set of n linearly independent lattice vectors of length at most $\gamma(n) \cdot \lambda_n(\Lambda)$.

3.3.1 Sampling algorithms

The following algorithms are used to sample short vectors and random basis from specific lattices.

Algorithm ToBasis : Micciancio and Goldwasser [MG02] showed that a full-rank set $\boldsymbol{S}$ in a lattice Λ can be converted into a basis T for Λ with an equally low Gram-Schmidt norm.

Lemma 3.3.7 ([MG02, Lemma 7.1]) Let Λ be an m-dimensional lattice. There is a deterministic polynomial-time algorithm that, given an arbitrary basis of Λ and a full-rank set $\boldsymbol{S} = \boldsymbol{s}_1, \dots, \boldsymbol{s}_m$ in Λ, returns a basis T of Λ satisfying $\|\widetilde{\boldsymbol{T}}\| \leq \|\widetilde{\boldsymbol{S}}\|$ and $\|\boldsymbol{T}\| \leq \|\boldsymbol{S}\|\sqrt{m}/2$

Algorithm TrapGen : Ajtai [Ajt96] and later Alwen and Peikert [AP09] showed how to sample an essentially uniform matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ along with a basis $\boldsymbol{S}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ with low Gram-Schmidt norm.

Theorem 3.3.8 ([AP09, Theorem 3.2] with $\delta = 1/3$) Let q, n, m be positive integers with $q \geq 2$ and $m \geq 6n \lg q$. There is a probabilistic polynomial-time algorithm $\mathsf{TrapGen}(q, n, m)$ that outputs a pair $(\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{S} \in \mathbb{Z}^{m\times m})$ such that $\boldsymbol{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\boldsymbol{S}$ is a basis for $\Lambda_q^{\perp}(\boldsymbol{A})$, satisfying $\|\widetilde{\boldsymbol{S}}\| \leq O(\sqrt{n \log q})$ and $\|\boldsymbol{S}\| \leq O(n \log q)$ with overwhelming probability in n.

We let $\sigma_{\mathrm{TG}} = O(\sqrt{n \log q})$ denote the maximum with high probability Gram-Schmidt norm of a basis produced by TrapGen.

Algorithm ExtendBasis : Peikert et al. [CHKP10] shows how to construct a basis for $\Lambda_q^{\perp}(\boldsymbol{A}\|\boldsymbol{B}\|\boldsymbol{C})$ from a basis for $\Lambda_q^{\perp}(\boldsymbol{B})$.

Theorem 3.3.9 For $i = 1, 2, 3$ let $\boldsymbol{A}_i$ be a matrix in $\mathbb{Z}_q^{n\times m_i}$ and let $\boldsymbol{A} := (\boldsymbol{A}_1\|\boldsymbol{A}_2\|\boldsymbol{A}_3)$. Let $\boldsymbol{T}_2$ be a basis of $\Lambda_q^{\perp}(\boldsymbol{A}_2)$. There is deterministic polynomial time algorithm $\mathsf{ExtendBasis}(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3, \boldsymbol{T}_2)$ that outputs a basis T for $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\widetilde{\boldsymbol{T}}\| = \|\widetilde{\boldsymbol{T}_2}\|$

Algorithm SampleLeft : The algorithm takes as input a full rank matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, a short basis $\boldsymbol{T}_{\boldsymbol{A}}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$, a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m_1}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ. Let $\boldsymbol{F} := (\boldsymbol{A}\|\boldsymbol{B})$, then the algorithm outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ in the coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F})$.

Theorem 3.3.10 ([ABB10, Theorem 17], [CHKP10, Lemma 3.2]) Let $q > 2, m > n$ and $\sigma > \|\boldsymbol{T}_{\boldsymbol{A}}\| \cdot \omega(\sqrt{\log(m+m_1)})$. Then $\mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{u}, \sigma)$ outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}$.

Algorithm SampleRight : The algorithm takes as input matrices $\boldsymbol{A} \in \mathbb{Z}_q^{n\times k}$ and $\boldsymbol{R} \in \mathbb{Z}^{k\times m}$, a full rank matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ and a short basis $\boldsymbol{T}_{\boldsymbol{B}}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ. Let $\boldsymbol{F} := (\boldsymbol{A}\|\boldsymbol{A}\boldsymbol{R} + \boldsymbol{B})$, then the algorithm outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{k+m}$ in the coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F})$.

Let S^m be the m-sphere $\{\boldsymbol{x} \in \mathbb{R}^{m+1} : \|\boldsymbol{x}\| = 1\}$. We define $s_{\boldsymbol{R}} := \|\boldsymbol{R}\| = \sup_{\boldsymbol{x}\in S^{m-1}} \|\boldsymbol{R}\cdot \boldsymbol{x}\|$.

Theorem 3.3.11 ([ABB10, Theorem 19]) Let $q > 2, m > n$ and $\sigma > \|\boldsymbol{T}_{\boldsymbol{B}}\| \cdot s_R \cdot \omega(\sqrt{\log(k+m)})$. Then $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{T}_{\boldsymbol{B}}, \boldsymbol{u}, \sigma)$ outputs a vector $\boldsymbol{e} \in Z^{k+m}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}$.

3.3.2 The LWE Problem

The Learning with Errors problem (LWE, for short) is the problem of determining a secret vector over $\mathbb{F}_q$ given a polynomial number of noisy inner products. The decision variant is to distinguish such samples from random. More formally, we define the (average-case) problem as follows:

Definition 3.3.12 ([Reg05]) Let $n \geq 1$ and $q \geq 2$ be integers, and let χ be a probability distribution on $\mathbb{Z}_q$. For $\boldsymbol{r} \in \mathbb{Z}_q^n$, let $A_{\boldsymbol{r},\chi}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to χ, and outputting $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{r}\rangle + e)$.

(a) The *search*-$\mathsf{LWE}_{q,n,\chi}$ problem is: for uniformly random $\boldsymbol{r} \in \mathbb{Z}_q^n$, given a $\mathrm{poly}(n)$ number of samples from $A_{\boldsymbol{r},\chi}$, output $\boldsymbol{r}$.

(b) The *decision*-$\mathsf{LWE}_{q,n,\chi}$ problem is: for uniformly random $\boldsymbol{r} \in \mathbb{Z}_q^n$, given a $\mathrm{poly}(n)$ number of samples that are either (all) from $A_{\boldsymbol{r},\chi}$ or (all) uniformly random in $\mathbb{Z}_q^n \times Z_q$, output 0 if the former holds and 1 if the latter holds.

We say the *decision*-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the *decision*-LWE problem (over $\boldsymbol{r}$ and $\mathcal{A}$ random coins) is negligibly close to $1/2$ as a function of n.

The power of the LWE problem comes from the fact that for certain noise distributions χ, solving the search-LWE problem is as hard as finding approximate solutions to the shortest independent vectors problem (SIVP) and the decision version of the shortest vector problem (GapSVP) in the worst case. For polynomial size q there is a quantum reduction due to Regev, while for exponential size q there is a classical reduction due to Peikert. Furthermore, the search and decision versions of the problem are equivalent whenever q is a product of small primes. These results are summarized in the following:

Definition 3.3.13 For $\alpha \in (0,1)$ and an integer $q > 2$, let $\overline{\Psi}_\alpha$ denote the probability distribution over $\mathbb{Z}_q$ obtained by choosing $x \in \mathbb{R}$ according to the normal distribution with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and outputting $\lfloor qx \rceil$.

Theorem 3.3.14 ([Reg05]) Let n, q be integers and $\alpha \in (0,1)$ such that $q = \mathrm{poly}(n)$ and $\alpha q > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves decision-$\mathsf{LWE}_{q,n,\overline{\Psi}_\alpha}$, then there exists an efficient quantum algorithm that approximates SIVP and GapSVP to within $\widetilde{O}(n/\alpha)$ in the worst case.

Theorem 3.3.15 ([Pei09]) Let n, q be integers and $\alpha \in (0,1)$, and $q = \sum_i q_i \leq 2^{n/2}$, where the q_i are distinct primes satisfying $\omega(\log n)/\alpha \leq q_i \leq \mathrm{poly}(n)$. If there exists an efficient (classical) algorithm that solves decision-$\mathsf{LWE}_{q,n,\overline{\Psi}_\alpha}$, then there exists an efficient (classical) algorithm that approximates GapSVP to within $\widetilde{O}(n/\alpha)$ in the worst case.

The following lemma will be used to show correctness of decryption.

Lemma 3.3.16 ([ABB10, Lemma 12]) Let $\boldsymbol{e}$ be some vector in $\mathbb{Z}^m$ and let $\boldsymbol{v} \leftarrow \overline{\Psi}_\alpha^m$. Then the quantity $|\langle \boldsymbol{e}, \boldsymbol{v}\rangle|$ when treated as an integer in $(-q/2, q/2]$ satisfies

$$|\langle \boldsymbol{e}, \boldsymbol{v}\rangle| \leq \|\boldsymbol{e}\| \cdot \Big(q\alpha \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2\Big)$$

with overwhelming probability in m.

Chapter 4

Generalized Key Delegation for Wildcarded Identity-Based and Inner-Product Encryption

Even though the WIBE and WKD-IBE constructions in [ACD$^+$06, AKN07] are very practical, they had two significant shortcomings. First, their security proofs only hold in cases where the maximum hierarchy depth L is a constant due to the fact that they are not tight and lose a factor which is exponential in L. As a result, these schemes can only be used in scenarios where such a restriction is acceptable. In particular, when using WKD-IBE schemes to build identity-based broadcast encryption schemes, such a limitation on the maximum hierarchy depth will have a direct impact on the maximum size of the target group. Second, their solutions do not hide the pattern associated with the ciphertext. Hence, their schemes cannot be used in any application where the anonymity of the recipient needs to be preserved.

The main goal here is to overcome the limitations of existing WKD-IBE and WIBE schemes. To achieve the first of these goals, we show in Section 4.2 how to convert the WIBE scheme by Abdalla *et al.* [ACD$^+$06] based on the Boneh-Boyen HIBE scheme [BB04] into a non-anonymous WW-IBE scheme which is fully secure even when the maximum hierarchy depth L is a polynomial in the security parameter. Towards this goal, we make use of bilinear groups of composite order [BGN05], whose order is a product of three primes. Our non-anonymous fully secure WW-IBE scheme has three main ingredients in its design. The first one is the generalization of the secret key delegation mechanism already present in the underlying Boneh-Boyen HIBE scheme. This technique is similar to one used by Abdalla *et al.* in [AKN07]. The second one is the addition of a ciphertext delegation mechanism, using a technique similar to the one employed in [ACD$^+$06]. Finally, the third key ingredient is the use of the dual system technique of Waters [Wat09, LW10] to achieve security with respect to adaptive adversaries. In a dual system, there are two kinds of keys and ciphertexts: normal and semi-functional. While normal keys can be used to decrypt both normal and semi-functional ciphertexts, semi-functional keys can only decrypt normal ciphertexts. When a semi-functional key is used to decrypt a semi-functional ciphertext, decryption will *fail* with all but negligible probability. In a dual system, a security proof usually employs a standard hybrid argument over a sequence of games. The first one is the real security game in which key derivation and encryption queries are answered with normal keys and ciphertexts. In the next game, the ciphertext given to the attacker is changed to semi-functional. Then, in each subsequent game, the secret keys used to answer key derivation queries are changed to semi-functional, one at a time. Finally, once everything the attacker receives is semi-functional, then it is straight-forward to prove security directly.

As we show in Section 4.2, these three ingredients fit nicely together and we are able to prove its security. However, this is not always the case. In particular, when considering WW-IBE schemes based on the Boneh-Boyen-Goh HIBE [BBG05], we did not succeed extending the proof strategy by Lewko and Waters [LW10] to the WW-IBE case. This is due to the fact that, when a normal secret key is changed to semi-functional, the information leaked by the delegation mechanisms of the secret keys together with the presence of wildcards in the challenge ciphertext breaks down the information theoretical argument used to hide the *nominality* of semi-functional keys. Nevertheless, it is worth pointing out that recent techniques by Lewko and Waters [LW12] might be useful in overcoming these issues.

One of the key elements in designing a WW-IBE scheme, starting from the HIBE schemes in [BB04, BBG05], is to have some sort of delegation mechanism also for the ciphertext in order to replace the wildcards with the identities required by the secret key. So we have two delegation mechanisms, one for the secret keys and one for the ciphertexts. These two mechanisms can be *symmetric* in the sense that they are essentially the same. Let us think to the [BBG05] scheme: If we want a delegation mechanism for the ciphertext what we can do is essentially to reuse the secret keys delegation mechanism. The issue here is that we are introducing a stronger correlation between ciphertexts and secret keys. Thus when we proceed to prove the semantic security by means of the dual system encryption, during the secret key games the information theoretical argument necessary to finish the proof breaks down because the ciphertext now release too much information. Instead this issue doesn't arise in the [BB04] scheme because we are able to define two different delegation mechanisms avoiding in this way the stronger correlation between ciphertexts and secret keys. Unfortunately when we try to achieve the anonymity property then it seems that this requires a symmetric delegation mechanism and this blocks us to achieve even semantic security.

In order to achieve, and to avoid all the issues of the previous approach, our second goal of building WW-IBE, which are not only fully secure but also anonymous, we first introduce in Section 4.3 a generalization of the notion of hierarchical inner predicate encryption [OT09], which we call inner-product encryption with generalized key delegation (WKD-IPE, for short). Next, after introducing the new WKD-IPE notion, we present a generic transform which converts any WKD-IPE into a WW-IBE scheme. The reduction uses the same trick used in [KSW08] to build hidden vector encryption from inner product encryption. Specifically, zero entries are used to simulate wildcards.

Since the transform preserves anonymity and has a tight security reduction, the resulting WW-IBE scheme will be anonymous and fully secure as long as the underlying WKD-IPE is anonymous and fully secure. Finally, we show how to modify the hierarchical inner-product encryption scheme from [LOS$^+$10] to allow for more general key delegation patterns. As for the previous scheme, we also use the dual system encryption methodology of Waters [Wat09] in the security proof.

It is worth noticing that, even though our constructions allow for wildcards in the key derivation algorithm, they do not take into account the limited delegation property through which one can prevent users from further deriving keys for identities below them. As remarked by Abdalla *et al.* in [AKN07], this is without loss of generality as the limited delegation property can be easily achieved in the case where the maximum depth is fixed by padding the identity vector with dummy strings at the unused levels. Hence, here we only focuses on wildcard key delegation.

4.1 Definitions

In this section, we introduce a new primitive, called identity-based encryption with wildcards and generalized key delegation, which combines the notions of wildcarded IBE [ACD$^+$06] and

IBE with generalized key delegation [AKN07]. In doing so, we adopt the same notation and definition style used in [ACD$^+$06].

4.1.1 Identity-Based Encryption with Wildcards and Generalized Key Delegation

Like WIBE and WKD-IBE, Identity-Based Encryption with Wildcards and Generalized Key Delegation (WW-IBE, for short) allows for more general patterns either during the encryption or the key derivation processes. Such a pattern is described by a vector

$$P = (P_1, \ldots, P_\ell) \in (\{0,1\}^* \cup \{\star\})^\ell ,$$

where $\star$ is a special wildcard symbol. As in a WKD-IBE, a user in possession of the secret key for a given pattern P can generate secret keys for any pattern P' that matches P. We say that a pattern $P' = (P'_1, \ldots, P'_{\ell'})$ *matches* P, denoted $P' \in_\star P$, if and only if $\ell' \le \ell$; $\forall\ i = 1 \ldots \ell'$, $P'_i = P_i$ or $P_i = \star$; and $\forall\ i = \ell' + 1 \ldots \ell$, $P_i = \star$. As in a WIBE, a user in possession of the secret key for a given pattern P can decrypt ciphertexts for any pattern P' that matches P.

Formally, an WW-IBE scheme can be defined as a predicate encryption scheme with public index in the following way. Let L be the hierarchy depth then, the keys space K and the indices space I consist of all the patterns $P = (P_1, \ldots, P_\ell) \in (\{0,1\}^* \cup \{\star\})^\ell$ for $1 \le \ell \le L$ and the WW-IBE predicate P_L over (K, I) is defined as follows:

$$P_L(k = P', \mathfrak{ind} = P) = \begin{cases} 1 & \text{if } P' \in_\star P \\ 0 & \text{otherwise} \end{cases} .$$

Definition 4.1.1 [WW-IBE] An Identity-Based Encryption with Wildcards and Generalized Key Delegation scheme is a tuple of 5 algorithms $\mathsf{WW-IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^L)$ outputs *public* and *master secret* keys (mpk, msk) for *security parameter* λ and *hierarchy depth* L.

$\mathsf{KeyGen}(msk, P = (P_1, \ldots, P_\ell))$ on input a master secret key msk and pattern P, outputs *secret key* sk_P;

$\mathsf{KeyDer}(mpk, sk_P, P')$ on input a master public key mpk, a secret key sk_P for pattern $P = (P_1, \ldots, P_\ell)$ and pattern P' such that $P' \in_\star P$, outputs *secret key* $sk_{P'}$;

$\mathsf{Enc}(mpk, P, \mathfrak{m})$ on input public key mpk, pattern $P = (P_1, \ldots, P_\ell)$ and *plaintext* $\mathfrak{m}$, outputs *ciphertext* Ct;

$\mathsf{Dec}(mpk, sk_{P'}, \mathsf{Ct})$ outputs a string y.

Correctness. We require that for all key pairs (mpk, msk) output by Setup, all messages $\mathfrak{m} \in \{0,1\}^*$, all $0 \le \ell \le L$, all patterns $P, P' \in (\{0,1\}^* \cup \{\star\})^\ell$ such that $P' \in_\star P$, all $sk_{P'}$ obtained through a chain of key derivation $sk_{P_1} = \mathsf{KeyGen}(msk, P_1)$, $sk_{P_{i+1}} = \mathsf{KeyDer}(mpk, sk_{P_i}, P_{i+1})$ such that $P_1 \in_\star \ldots \in_\star P_n = P'$,

$$\mathsf{Dec}(\ sk_{P'}\ ,\ \mathsf{Enc}(mpk, P, \mathfrak{m})\ ) = \mathfrak{m}$$

with probability one.

4.1.2 Security

As for Definition 2.2.1, IND-CPA security for WW-IBE schemes is formalized by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Specifically, we define the security of WW-IBE schemes through a game with an adversary which is different from the security games of HIBE schemes in two points: first, the adversary can query for the secret keys corresponding to arbitrary patterns, rather than specific identity vectors and second, the adversary chooses a challenge pattern instead of an identity to which the challenge ciphertext will be encrypted. Of course, the adversary is not allowed to query the key derivation oracle for any pattern matched by the challenge pattern.

We give complete form of the security definition following [SW08]. Specifically, game IND-CPA$_{\mathcal{A}}^{WW-IBE}(1^\lambda)$, where λ is the security parameter, is defined as follows.

Setup: $\mathcal{C}$ generates master public key and secret key by invoking the setup algorithm on input the security parameter λ and the hierarchy depth L given in unary. Specifically, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$. Then, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input mpk.

We let S denote the set of private keys that $\mathcal{C}$ has created but not yet given to the adversary. At this point, $S = \emptyset$.

Query Phase 1: $\mathcal{A}$ makes *Create*, *Delegate*, and *Reveal* key queries. Specifically:

To make a *Create* query, $\mathcal{A}$ specifies a pattern $P \in \big(\{0,1\}^* \cup \{\star\}\big)^{\leq L}$. In response, $\mathcal{C}$ creates a key for this pattern by calling the key generation algorithm, $sk_P = \mathsf{KeyGen}(msk, P)$, and places this key in the set S. $\mathcal{C}$ only gives $\mathcal{A}$ a reference to this key, not the key itself.

To make a *Delegate* query, $\mathcal{A}$ specifies a key sk_P in the set S and a pattern P'. In response, $\mathcal{C}$ makes a key for this new vector by running the delegation algorithm on sk_P and P'. $\mathcal{C}$ adds this key to the set S and again gives $\mathcal{A}$ only a reference to it, not the actual key.

To make a *Reveal* query, $\mathcal{A}$ specifies an element of the set S. $\mathcal{C}$ gives this key to $\mathcal{A}$ and removes it from the set S. Notice that $\mathcal{A}$ needs no longer make any delegation queries for this key because it can run delegation algorithm on the revealed key for itself.

Challenge: $\mathcal{A}$ submits two equal length messages $\mathfrak{m}_0, \mathfrak{m}_1$ and a challenge pattern $P = (P_1, \ldots, P_\ell)$ where $0 \leq \ell \leq L$. $\mathcal{C}$ chooses random $\eta \in \{0,1\}$ and generates ciphertext $\mathsf{Ct}^\star = \mathsf{Enc}(mpk, P, \mathfrak{m}_\eta)$. $\mathcal{C}$ sends back $\mathsf{Ct}^\star$ to $\mathcal{A}$.

Query Phase 2: It is the same to Query Phase 1.

Guess: Eventually, $\mathcal{A}$ submits its guess η'.

Winning Condition: $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. $\eta = \eta'$.
2. No secret key for any pattern obtained through delegation from any of the revealed secret key is such that it matches the target pattern (i.e., any P' such that $P' \in_\star P$).

The *advantage* of $\mathcal{A}$ in the above game is defined as

$$\mathbf{Adv}_{\text{IND-CPA}}^{WW-IBE,\mathcal{A}}(\lambda) = \text{Prob}[\text{IND-CPA}_{\mathcal{A}}^{WW-IBE}(1^\lambda) = 1] - 1/2\,.$$

Definition 4.1.2 A WW-IBE scheme is IND-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the IND-CPA game described above, *i.e.,* $\mathbf{Adv}^{WW-IBE,\mathcal{A}}_{\text{IND-CPA}}(\lambda)$ is negligible function of λ.

4.1.3 Anonymous WW-IBE

An important property we investigate is the anonymity of WW-IBE schemes. The objective is to hide the pattern in the ciphertext. Therefore, the adversary could also choose two patterns to be challenged on. More specifically, game ANO-CPA$^{WW-IBE}_{\mathcal{A}}(1^\lambda)$, where λ is the security parameter, is defined as follows.

Setup: $\mathcal{C}$ generates master public key and secret key by invoking the setup algorithm on input the security parameter λ and the hierarchy depth L given in unary. Specifically, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$. Then, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input mpk.

We let S denote the set of private keys that $\mathcal{C}$ has created but not yet given to the adversary. At this point, $S = \emptyset$.

Query Phase 1: $\mathcal{A}$ makes *Create*, *Delegate*, and *Reveal* key queries. Specifically:

To make a *Create* query, $\mathcal{A}$ specifies a pattern $P \in \left(\{0,1\}^* \cup \{\star\}\right)^{\leq L}$. In response, $\mathcal{C}$ creates a key for this pattern by calling the key generation algorithm, $sk_P = \mathsf{KeyGen}(msk, P)$, and places this key in the set S. $\mathcal{C}$ only gives $\mathcal{A}$ a reference to this key, not the key itself.

To make a *Delegate* query, $\mathcal{A}$ specifies a key sk_P in the set S and a pattern P'. In response, $\mathcal{C}$ makes a key for this new vector by running the delegation algorithm on sk_P and P'. $\mathcal{C}$ adds this key to the set S and again gives $\mathcal{A}$ only a reference to it, not the actual key.

To make a *Reveal* query, $\mathcal{A}$ specifies an element of the set S. $\mathcal{C}$ gives this key to $\mathcal{A}$ and removes it from the set S. Notice that $\mathcal{A}$ needs no longer make any delegation queries for this key because it can run delegation algorithm on the revealed key for itself.

Challenge: $\mathcal{A}$ submits two equal length messages $\mathfrak{m}_0, \mathfrak{m}_1$ and two challenge patterns $\mathbf{P}_0 = (P_1^{(0)}, \ldots, P_\ell^{(0)})$ and $\mathbf{P}_1 = (P_1^{(1)}, \ldots, P_\ell^{(1)})$ where $0 \leq \ell \leq L$. $\mathcal{C}$ chooses random $\eta \in \{0,1\}$ and generates ciphertext $\mathsf{Ct}^\star = \mathsf{Enc}(mpk, P_\eta, \mathfrak{m}_\eta)$. $\mathcal{C}$ sends back $\mathsf{Ct}^\star$ to $\mathcal{A}$.

Query Phase 2: It is the same as Query Phase 1.

Guess: Eventually, $\mathcal{A}$ submits its guess η'.

Winning Condition: $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. $\eta = \eta'$.
2. No secret key for any pattern obtained through delegation from any of the revealed secret key is such that it matches one of the challenge pattern but not the other.

The *advantage* of $\mathcal{A}$ in the above game is defined as

$$\mathbf{Adv}^{WW-IBE,\mathcal{A}}_{\text{ANO-CPA}}(\lambda) = \text{Prob}[\text{ANO-CPA}^{WW-IBE}_{\mathcal{A}}(1^\lambda) = 1] - 1/2\ .$$

Definition 4.1.3 A WW-IBE scheme is ANO-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the ANO-CPA game described above, *i.e.,* $\mathbf{Adv}_{\text{ANO-CPA}}^{WW-IBE,\mathcal{A}}(\lambda)$ is negligible function of λ.

4.2 A WW-IBE Construction

Our first construction is based on a slight variant of the Boneh-Boyen WIBE scheme given in [ACD$^+$06]. We extend it to obtain generalized key delegation and prove its security in the standard model, under static assumptions in composite order bilinear groups [BGN05], by using Waters' Dual Encryption System [Wat09]. As in [LW10], we use groups $\mathbb{G}, \mathbb{G}_{\mathrm{T}}$ whose order $N = p_1p_2p_3$ is a product of three primes. We note that the actual computation occurs in $\mathbb{G}_{p_1}$, the subgroup of $\mathbb{G}$ of order p_1. $\mathbb{G}_{p_2}$, the subgroup of $\mathbb{G}$ of order p_2, is only used to define the semi-functional space and is not used in the real scheme. Thus, secret keys and ciphertexts will be semi-functional when they include terms in $\mathbb{G}_{p_2}$. Finally, $\mathbb{G}_{p_3}$, the subgroup of $\mathbb{G}$ of order p_3, is used to re-randomize the secret keys.

4.2.1 Complexity Assumptions

For completeness we restate here the complexity assumptions we use, first introduced by Lewko and Waters in [LW10]. The first two are instances of the general subgroup decision assumption.

Assumption LW.1 For a given prime-order bilinear setting generator BSGen, define the following distribution: Pick a random bilinear setting $(N = p_1p_2p_3, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BSGen}(1^\lambda)$, and then pick

$$g_1, T_1 \leftarrow \mathbb{G}_{p_1}, g_3 \leftarrow \mathbb{G}_{p_3}, T_2 \leftarrow \mathbb{G}_{p_1p_2} ,$$

and set $D = (N, g_1, g_3)$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking Assumption LW.1 is

$$\mathbf{Adv}_{LW.1}^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]| .$$

Definition 4.2.1 We say that Assumption LW.1 holds for generator BSGen if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_{LW.1}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption LW.2 For a given prime-order bilinear setting generator BSGen, define the following distribution: Pick a random bilinear setting $(N = p_1p_2p_3, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BSGen}(1^\lambda)$, and then pick

$$g_1, X_1 \leftarrow \mathbb{G}_{p_1}, X_2, Y_2 \leftarrow \mathbb{G}_{p_2}, g_3, X_3 \leftarrow \mathbb{G}_{p_3}, T_1 \leftarrow \mathbb{G}_{p_1p_3}, T_2 \leftarrow \mathbb{G}_{p_1p_2p_3} ,$$

and set $D = (N, g_1, g_3, X_1X_2, Y_2X_3)$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking Assumption LW.2 is

$$\mathbf{Adv}_{LW.2}^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]| .$$

Definition 4.2.2 We say that Assumption LW.2 holds for generator BSGen if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_{LW.2}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption LW.3 For a given prime-order bilinear setting generator BSGen, define the following distribution: Pick a random bilinear setting $(N = p_1p_2p_3, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{BSGen}(1^\lambda)$, and then pick

$$\rho, s \leftarrow \mathbb{Z}_N, g_1 \leftarrow \mathbb{G}_{p_1}, g_2, X_2, Y_2 \leftarrow \mathbb{G}_{p_2}, g_3 \leftarrow \mathbb{G}_{p_3}, T_2 \leftarrow \mathbb{G}_{\mathrm{T}} ,$$

and set $T_1 = \hat{e}(g_1, g_1)^{\rho s}$ and $D = (N, g_1, g_2, g_3, g_1^\rho X_2, g_1^s Y_2)$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking Assumption LW.3 is

$$\mathbf{Adv}^{\mathcal{A}}_{LW.3}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_2) = 1]| .$$

Definition 4.2.3 We say that Assumption LW.3 holds for generator BSGen if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}^{\mathcal{A}}_{LW.3}(\lambda)$ is a negligible function of λ.

4.2.2 Our Construction

We first introduce some additional notation. If $P = (P_1, ..., P_\ell)$ is a pattern, then let $|P| = \ell$ be the length of P, let $\mathrm{W}(P)$ be the set containing all wildcard indices in P, i.e. the indices $1 \leq i \leq \ell$ such that $P_i = \star$, and let $\overline{\mathrm{W}}(P)$ be the complementary set containing all non-wildcard indices. Clearly $\mathrm{W}(P) \bigcap \overline{\mathrm{W}}(P) = \emptyset$ and $\mathrm{W}(P) \bigcup \overline{\mathrm{W}}(P) = \{1, ..., \ell\}$.

$\mathsf{Setup}(1^\lambda, 1^L)$: Choose a description of a composite order bilinear setting $(N = p_1p_2p_3, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 3)$ with known factorization and random $g_1 \leftarrow \mathbb{G}_{p_1}$, $g_3 \leftarrow \mathbb{G}_{p_3}$. Choose random $\rho \leftarrow \mathbb{Z}_N$ and random $(u_{i,j} \leftarrow \mathbb{G}_{p_1})_{i \in [L], j \in \{0,1\}}$, and set $\Omega = \hat{e}(g_1, g_1)^\rho$. Then, the master public key is

$$mpk \leftarrow [N, g_1, g_3, (u_{i,j})_{i \in [L], j \in \{0,1\}}, \Omega] ,$$

and the master secret key is

$$msk \leftarrow [\rho] .$$

$\mathsf{KeyGen}(msk, P = (P_1, \ldots, P_\ell))$: Choose random $(r_i \leftarrow \mathbb{Z}_N)_{i \in \overline{\mathrm{W}}(P)}$, $W_0 \leftarrow \mathbb{G}_{p_3}$ and $(W_i \leftarrow \mathbb{G}_{p_3})_{i \in \overline{\mathrm{W}}(P)}$. Then set

$$K_0 = g_1^\rho \cdot \prod_{i \in \overline{\mathrm{W}}(P)} \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^{r_i} \cdot W_0 ,$$

and

$$(D_i = g_1^{r_i} \cdot W_i)_{i \in \overline{\mathrm{W}}(P)}$$

The secret key for pattern P is $sk_P \leftarrow [K_0, (D_i)_{i \in \overline{\mathrm{W}}(P)}]$.

$\mathsf{KeyDel}(mpk, sk_P, P' = (P'_1, \ldots, P'_\ell))$: If $P' \in_\star P$ then the key delegation algorithm chooses random $(r'_i \leftarrow \mathbb{Z}_N)_{i \in \overline{\mathrm{W}}(P')}$ and $W'_0 \leftarrow \mathbb{G}_{p_3}$ and $(W'_i \leftarrow \mathbb{G}_{p_3})_{i \in \overline{\mathrm{W}}(P')}$. Then the algorithm sets

$$K'_0 = K_0 \cdot \prod_{i \in \overline{\mathrm{W}}(P')} \left(u_{i,0} \cdot u_{i,1}^{P'_i}\right)^{r'_i} \cdot W'_0 ,$$

and

$$\left(D'_i = D_i \cdot g_1^{r'_i} \cdot W'_i\right)_{i \in \overline{\mathrm{W}}(P) \bigcap \overline{\mathrm{W}}(P')} \quad \text{and} \quad \left(D'_i = g_1^{r'_i} \cdot W'_i\right)_{i \in \mathrm{W}(P) \bigcap \overline{\mathrm{W}}(P')} .$$

The secret key for pattern P' is $sk_{P'} \leftarrow [K'_0, (D'_i)_{i \in \overline{\mathrm{W}}(P')}]$.

$\mathsf{Enc}(mpk, P = (P_1, \ldots, P_\ell), \mathfrak{m})$: Choose random $s \leftarrow \mathbb{Z}_N$ and then set

$$C_1 = g_1^s \;,$$

$$\left(C_{2,i} = \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^s\right)_{i \in \overline{\mathrm{W}}(P)} \;,$$

$$C_3 = \Omega^s \cdot \mathfrak{m} \;,$$

$$\left(C_{4,i,j} = u_{i,j}^s\right)_{i \in \mathrm{W}(P), j \in \{0,1\}} \;.$$

The ciphertext for pattern P and message $\mathfrak{m}$ is $\mathsf{Ct}_P \leftarrow [C_1, (C_{2,i}) C_3, (C_{4,i,j})]$.

$\mathsf{Dec}(mpk, \mathsf{Ct}_P, sk_{P'})$: The decryption algorithms returns $\perp$ if $P \not\in_\star P'$. If this is not the case then to decrypt it is enough to focus on the non-wildcard positions of the secret key. Then, each identity in these positions must be matched by a corresponding identity in the ciphertext. In particular, if position i corresponds to a non-wildcard in the secret key and to a wildcard in the ciphertext (these positions are those in $\mathrm{W}(P) \bigcap \overline{\mathrm{W}}(P')$) then that wildcard needs to be delegated to the required identity P'_i. Thus, to recover the message $\mathfrak{m}$ the decryption algorithms computes what follows:

$$C_2 = \prod_{i \in \overline{\mathrm{W}}(P) \bigcap \overline{\mathrm{W}}(P')} \hat{e}(D_i, C_{2,i}) \cdot \prod_{i \in \mathrm{W}(P) \bigcap \overline{\mathrm{W}}(P')} \hat{e}\left(D_i, C_{4,i,0} \cdot C_{4,i,1}^{P'_i}\right) \;,$$

then the blinding factor is obtained as

$$\frac{\hat{e}(K_0, C_1)}{C_2} = \frac{\Omega^s \cdot \prod_{i \in \overline{\mathrm{W}}(P')} \hat{e}\left(\left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^{r_i}, g_1^s\right)}{\prod_{i \in \overline{\mathrm{W}}(P')} \hat{e}\left(g_1^{r_i}, \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^s\right)} = \Omega^s \;.$$

Remember that the $\mathbb{G}_{p_3}$ part cancels out by the orthogonality property.

4.2.3 Security

In this section we will prove the following main theorem:

Theorem 4.2.4 If Assumptions LW.1, LW.2, and LW.3 hold, then our scheme is IND-CPA secure in the sense of Definition 4.1.2.

Following Lewko and Waters [LW10], we define two additional structures used in the proofs of security. Specifically:

Semi-Functional Ciphertext : Let g_2 denote a generator of $\mathbb{G}_{p_2}$. An semi-functional ciphertext is created as follows: first, we use the encryption algorithm to form a normal ciphertext $\mathsf{Ct}' \leftarrow [C'_1, (C'_{2,i}), C'_3, (C'_{4,i,j})]$. Then, we choose random exponents $x \leftarrow \mathbb{Z}_N, (z_{c,i} \leftarrow \mathbb{Z}_N)_{i \in \overline{\mathrm{W}}(P)}$ and $(z_{c,i,j} \leftarrow \mathbb{Z}_N)_{i \in \mathrm{W}(P), j \in \{0,1\}}$. Finally, we set

$$C_1 = C'_1 \cdot g_2^x \;,$$

$$\left(C_{2,i} = C'_{2,i} \cdot g_2^{x \cdot z_{c,i}}\right)_{i \in \overline{\mathrm{W}}(P)} \;,$$

$$C_3 = C'_3 \;,$$

$$\left(C_{4,i,j} = C'_{4,i,j} \cdot g_2^{x \cdot z_{c,i,j}}\right)_{i \in \mathrm{W}(P), j \in \{0,1\}} \;.$$

Semi-Functional Keys : To create a semi-functional key, we first create a normal key $sk_P \leftarrow (K'_0, (D'_i))$ using the key generation algorithm. Then, we choose random $y, z_k \leftarrow \mathbb{Z}_N$ and random $(z_{k,i} \leftarrow \mathbb{Z}_N)_{i \in \overline{\mathrm{W}}(P)}$. Finally, we set

$$K_0 = K'_0 \cdot g_2^{y \cdot z_k}$$

and

$$\left(D_i = D'_i \cdot g_2^{y \cdot z_{k,i}} \cdot W_i\right)_{i \in \overline{\mathrm{W}}(P)} .$$

Remark 4.2.5 Notice that semi-functional ciphertexts can still be decrypted by normal secret keys and semi-functional secret keys can still be used to decrypt normal ciphertext. But an semi-functional ciphertext, for pattern P', and an semi-functional secret key, for pattern P, lead to a random decryption with high probability. In fact the decryption algorithm will compute the blinding factor multiplied by the additional term $\hat{e}(g_2, g_2)^{xy \cdot (z_c - z_k)}$ where

$$z_c = \sum_{i \in \overline{\mathrm{W}}(P) \bigcap \overline{\mathrm{W}}(P')} z_{k,i} \cdot z_{c,i} + \sum_{i \in \overline{\mathrm{W}}(P) \bigcap \mathrm{W}(P')} z_{k,i} \cdot \left(z_{c,i,0} + z_{c,i,1}\right) .$$

If $z_c = z_k$, decryption will still work. We say that such a ciphertexts and secret keys are *nominally* semi-functional. We will use this concept during the security proofs arguing that the nominality is hidden even to an unbounded adversary under the constraints of the security game.

For a probabilistic polynomial-time adversary $\mathcal{A}$ which makes q key queries, our proof of security will consist of the following $q + 5$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$.

$\mathbf{Game_{Real}}$: It is the real IND-CPA security game.

$\mathbf{Game_{WD}}$: It is the same as $\mathbf{Game_{Real}}$ except that all key queries will be answered by fresh calls to the key generation algorithm.

$\mathbf{Game_{Restricted}}$: It is the same as $\mathbf{Game_{WD}}$ except that $\mathcal{A}$ cannot ask for keys for patterns which are prefixes of one of the challenge patterns modulo p_2. We will retain this restriction in all subsequent games.

$\mathbf{Game_k}$: For k from 0 to q, we define $\mathbf{Game_k}$ like $\mathbf{Game_{Restricted}}$ except that the ciphertext given to $\mathcal{A}$ is semi-functional and the first k keys are semi-functional. The rest of the keys are normal.

$\mathbf{Game_{Final}}$: It is the same game as $\mathbf{Game_q}$, except that the challenge ciphertext has C_3 random in $\mathbb{G}_\mathrm{T}$ (thus the ciphertext is independent from the messages provided by $\mathcal{A}$). It is clear that no adversary can have advantage greater than 0 here.

We start by giving an overview of the proof of security.

$\mathbf{Game_{Real}} = \mathbf{Game_{WD}}$: The keys are identically distributed whether they are produced by key delegation from a previous key or from a fresh call to the key generation.

$\mathbf{Game_{WD}} \approx_\mathbf{c} \mathbf{Game_{Restricted}}$: Essentially, if the adversary is able to ask for keys for patterns which are prefixes of one of the challenge patterns modulo p_2, then this means that the adversary can find a non-trivial factor of N and can be used to break either Assumption LW.2. This restriction is crucial to show that $\mathbf{Game_{k-1}} \approx \mathbf{Game_k}$.

$\mathbf{Game_{Restricted}} \approx_\mathbf{c} \mathbf{Game_0}$: In $\mathbf{Game_0}$, the challenge ciphertext $\mathsf{Ct}^\star$ is semi-functional, while all keys are normal. Notice that Assumption LW.1 (N, g_1, g_3) is enough to generate mpk and msk. Then, T can be used to generate $\mathsf{Ct}^\star$ and, depending on the nature of T, $\mathsf{Ct}^\star$ can be normal as in $\mathbf{Game_{Restricted}}$ or semi-functional as in $\mathbf{Game_0}$.

$\mathbf{Game_{k-1}} \approx_\mathbf{c} \mathbf{Game_k}$: Given an instance of Assumption LW.2 $(N, g_1, g_3, X_1X_2, Y_2X_3)$, mpk and msk can be generated by using g_1 and g_3. The first $k-1$ secret key queries, which are semi-functional, can be answered by employing Y_2X_3. The last $q-k$ queries can be answered by invoking KeyGen on input msk. Finally, the challenge ciphertext can be generated by employing X_1X_2 and the k-th secret key by employing T. Now, if $T \in \mathbb{G}_{p_1p_3}$, then the k-th secret key is normal and the joint distribution of the k-th secret key and the challenge ciphertext is as in $\mathbf{Game_{k-1}}$. On the other hand, if $T \in \mathbb{G}_{p_1p_2p_3}$, then the k-th key is nominally semi-functional with the respect to the challenge ciphertext. Hence, the simulator cannot test by himself the nature of T. Moreover, the nominality of k-th key is hidden to the adversary under the restriction of $\mathbf{Game_{Restricted}}$ and from the fact that the adversary cannot ask secret keys for patters matching the challenge pattern.

$\mathbf{Game_q} \approx_\mathbf{c} \mathbf{Game_{Final}}$: In $\mathbf{Game_q}$, the challenge ciphertext and secret keys are semi-functional. It is easy to see that these two games are indistinguishable under Assumption LW.3.

$\mathbf{Game_{Final}}$ gives no advantage: In $\mathbf{Game_{Final}}$, η is information-theoretically hidden from the attacker. Hence the attacker can obtain no advantage in breaking the scheme.

We give now formal proofs of the above statements.

Lemma 4.2.6 $\mathbf{Game_{Real}} = \mathbf{Game_{WD}}$.

Proof: We note that the keys are identically distributed whether they are produced by the key delegation algorithm from a previous key or from a fresh call to the key generation algorithm. Thus, in the attacker's view, there is no difference between these games. ▌

Lemma 4.2.7 If Assumption LW.2 holds, then for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Game_{WD}} \approx_\mathbf{c} \mathbf{Game_{Restricted}}$.

Proof: Suppose that $\mathcal{A}$ has probability ϵ of producing a pattern $P = (P_1, \dots, P_k)$, that is a prefix of one of the challenge pattern $P^\star = (P_1^\star, \dots, P_j^\star)$ modulo p_2. That is, there exists i such that that $P_i \neq P_i^\star$ modulo N and both are different from $\star$, and that p_2 divides $P_i - P_i^\star$ and thus $a = \gcd(P_i - P_i^\star, N)$ is a nontrivial factor of N. We notice that p_2 divides a and set $b = \frac{N}{a}$. The following two cases are exhaustive and at least one occurs with probability at least $\epsilon/2$:

1. $\mathrm{ord}(g_1) \mid b$. Suppose case 1 has probability at least $\epsilon/2$. We describe algorithm $\mathcal{B}$ that breaks Assumption LW.2. $\mathcal{B}$ receives $(N, g_1, g_3, X_1X_2, Y_2X_3)$ and T and constructs mpk and msk by choosing $\rho \leftarrow \mathbb{Z}_N$ and $(u_{i,j} \leftarrow \mathbb{Z}_N)_{i\in[L], j\in\{0,1\}}$, and setting $\Omega = \hat{e}(g_1, g_1)^\rho$, $mpk \leftarrow [N, g_1, g_3, (g_1^{u_{i,j}})_{i\in[L], j\in\{0,1\}}, \Omega]$ and $msk \leftarrow [\rho]$. Then $\mathcal{B}$ runs $\mathcal{A}$ on input mpk and uses knowledge of msk to answer $\mathcal{A}$'s queries. At the end of the game, for all pattern P's for which $\mathcal{A}$ has asked for the key and for the challenge pattern $P^\star$, $\mathcal{B}$ computes $a = \gcd(P_i - P_i^\star, N)$. Then, if $\hat{e}\left((X_1X_2)^a, Y_2X_3\right)$ is the identity element of $\mathbb{G}_T$ then $\mathcal{B}$ tests if $\hat{e}(T^b, X_1X_2)$ is the identity element of $\mathbb{G}_T$. If this second test is successful, then $\mathcal{B}$ declares $T \in \mathbb{G}_{p_1p_3}$. If it is not, $\mathcal{B}$ declares $T \in \mathbb{G}_{p_1p_2p_3}$. It is easy to see that if p_2 divides a and $p_1 = \mathrm{ord}(g_1)$ divides b, then $\mathcal{B}$'s output is correct.

2. $\mathrm{ord}(g_1) \nmid b$ and $\mathrm{ord}(g_3) \mid b$. In this case, $\mathcal{B}$ breaks Assumption LW.2 in the same way except that mpk by exchanging the roles of g_{p_1} and g_{p_3} and thus p_1 and p_3.

∎

Lemma 4.2.8 If Assumption LW.1 holds, then for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Game_{Restricted}} \approx_\mathbf{c} \mathbf{Game_0}$.

Proof: We describe algorithm $\mathcal{B}$ that breaks Assumption LW.1. $\mathcal{B}$ receives (N, g_1, g_3) and T and simulates $\mathbf{Game_{Restricted}}$ or $\mathbf{Game_0}$ with $\mathcal{A}$ depending on T.

Setup : $\mathcal{B}$ chooses random $\rho \leftarrow \mathbb{Z}_N$ and $(u_{i,j} \leftarrow \mathbb{Z}_N)_{i\in[L],j\in\{0,1\}}$, and set $\Omega = \hat{e}(g_1, g_1)^\rho$. Then, $\mathcal{B}$ set

$$mpk \leftarrow [N, g_1, g_3, (g_1^{u_{i,j}})_{i\in[L],j\in\{0,1\}}, \Omega] \ ,$$

as the public parameters and

$$msk \leftarrow [\rho] \ ,$$

as the master secret key. $\mathcal{B}$ starts the interaction with $\mathcal{A}$ on input mpk.

Secret Keys : Notice that $\mathcal{B}$ knows msk and thus can answer all $\mathcal{A}$'s queries.

Challenge : $\mathcal{A}$ sends $\mathcal{B}$ challenge messages $\mathfrak{m}_0, \mathfrak{m}_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \ldots, P_\ell)$ where $0 \le \ell \le L$. $\mathcal{B}$ chooses random $\eta \leftarrow \{0,1\}$ and computes the challenge ciphertext $\mathsf{Ct}^\star$ as follows:

$$C_1 = T \ ,$$
$$\left(C_{2,i} = T^{u_{i,0}+u_{i,1}\cdot P_i}\right)_{i\in\overline{\mathrm{W}}(P)} \ ,$$
$$C_3 = \hat{e}(T, g_1)^\rho \cdot \mathfrak{m},$$
$$\left(C_{4,i,j} = T^{u_{i,j}}\right)_{i\in\mathrm{W}(P),j\in\{0,1\}} \ .$$

Notice that if $T \in \mathbb{G}_{p_1}$, then T can be written as g_1^s and $\mathsf{Ct}^\star$ is a normal ciphertext with randomness s. Instead, if $T \in \mathbb{G}_{p_1p_2}$, then T can be written as $g_1^s g_2^x$ and $\mathsf{Ct}^\star$ is semi-functional with randomness $s, x, z_{c,i} = u_{i,0} + u_{i,1} \cdot P_i, z_{c,i,j} = u_{i,j}$. ∎

Lemma 4.2.9 If Assumption LW.2 holds, then for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Game_{k-1}} \approx_\mathbf{c} \mathbf{Game_k}$.

Proof: We describe algorithm $\mathcal{B}$ that breaks Assumption LW.2. $\mathcal{B}$ receives $(N, g_1, g_3, X_1X_2, Y_2X_3)$ and T and simulates $\mathbf{Game_{k-1}}$ or $\mathbf{Game_k}$ with $\mathcal{A}$ depending on T.

Setup : $\mathcal{B}$ chooses random $\rho \leftarrow \mathbb{Z}_N$ and $(u_{i,j} \leftarrow \mathbb{Z}_N)_{i\in[L],j\in\{0,1\}}$, and set $\Omega = \hat{e}(g_1, g_1)^\rho$. Then $\mathcal{B}$ set

$$mpk \leftarrow [N, g_1, g_3, (g_1^{u_{i,j}})_{i\in[L],j\in\{0,1\}}, \Omega] \ ,$$

as the public parameters and

$$msk \leftarrow [\rho] \ ,$$

as the master secret key. $\mathcal{B}$ starts the interaction with $\mathcal{A}$ on input mpk.

Secret Keys : Let us now explain how $\mathcal{B}$ answers the i-th key query for pattern $P = (P_1, \dots, P_\ell)$ where $0 \le \ell \le L$.

$i < k$: $\mathcal{B}$ creates an semi-functional secret key as follows: Choose random $z'_k \leftarrow \mathbb{Z}_N$, $(r_i, z'_{k,i} \leftarrow \mathbb{Z}_N)_{i \in \overline{W}(P)}$, $W_0 \leftarrow \mathbb{G}_{p_3}$ and $(W_i \leftarrow \mathbb{G}_{p_3})_{i \in \overline{W}(P)}$. Then set

$$K_0 = g_1^{\rho} \cdot \prod_{i \in \overline{W}(P)} \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^{r_i} \cdot (Y_2 X_3)^{z'_k} \cdot W_0 \ ,$$

and

$$\left(D_i = g_1^{r_i} \cdot (Y_2 X_3)^{z'_{k,i}} \cdot W_i\right)_{i \in \overline{W}(P)} \ .$$

By writing Y_2 as g_2^y, we have that this is a properly distributed semi-functional key with randomness y, $z_k = z'_k$ and $z_{k,i} = z'_{k,i}$.

$i = k$: To answer the k-th key query $\mathcal{B}$ chooses random $(r'_i \leftarrow \mathbb{Z}_N)_{i \in \overline{W}(P)}$, $W_0 \leftarrow \mathbb{G}_{p_3}$ and $(W_i \leftarrow \mathbb{G}_{p_3})_{i \in \overline{W}(P)}$, and sets

$$K_0 = g_1^{\rho} \cdot \prod_{i \in \overline{W}(P)} \left(T^{u_{i,0} + u_{i,1} \cdot P_i}\right)^{r'_i} \cdot W_0 \ ,$$

and

$$\left(D_i = T^{r'_i} \cdot W_i\right)_{i \in \overline{W}(P)} \ .$$

Notice that, if $T \in \mathbb{G}_{p_1 p_3}$, then T can be written as $g_1^r g_3^w$ and the k-th secret key is a normal key with randomness $r_i = r \cdot r'_i$, If $T \in \mathbb{G}_{p_1 p_2 p_3}$, then T can be written as $g_1^r g_2^y g_3^w$ and the k-th secret key is semi-functional with randomness $r_i = r \cdot r'_i$, y, $z_k = \sum_{i \in \overline{W}(P)} r'_i \cdot (u_{i,0} + u_{i,1} \cdot P_i)$ and $z_{k,i} = r'_i$.

$i > k$: $\mathcal{B}$ runs KeyGen using msk.

Challenge : $\mathcal{A}$ sends $\mathcal{B}$ challenge messages $\mathfrak{m}_0, \mathfrak{m}_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \dots, P_\ell)$ where $0 \le \ell \le L$. $\mathcal{B}$ chooses random $\eta \leftarrow \{0,1\}$ and computes the challenge ciphertext $\mathsf{Ct}^\star$ as follows:

$$C_1 = X_1 X_2 \ ,$$

$$\left(C_{2,i} = (X_1 X_2)^{u_{i,0} + u_{i,1} \cdot P_i}\right)_{i \in \overline{W}(P)} \ ,$$

$$C_3 = \hat{e}(X_1 X_2, g_1)^{\rho} \cdot \mathfrak{m} \ ,$$

$$\left(C_{4,i,j} = (X_1 X_2)^{u_{i,j}}\right)_{i \in W(P), j \in \{0,1\}} \ .$$

Notice that $X_1 X_2$ can be written as $g_1^s g_2^x$ for some random $s, x \in \mathbb{Z}_N$, thus the challenge ciphertext is semi-functional with randomness s, x, $z_{c,i} = u_{i,0} + u_{i,1} \cdot P_i$, $z_{c,i,j} = u_{i,j}$.

Since the k-th secret key pattern is not a prefix of the challenge pattern modulo p_2 we have that z_k and z_c are independent and randomly distributed. We observe that, if $\mathcal{B}$ attempts to test whether the k-th key is semi-functional by using the above procedure to create an semi-functional ciphertext for k-th secret key pattern then we will have that $z_k = z_c$ and thus decryption always works (independently of T). ▌

Lemma 4.2.10 If Assumption LW.3 holds, then for all probabilistic polynomial time adversaries $\mathcal{A}$, $\mathbf{Game_q} \approx_c \mathbf{Game_{Final}}$.

Proof: We describe algorithm $\mathcal{B}$ that breaks Assumption LW.3. $\mathcal{B}$ receives $(N, g_1, g_2, g_3, g_1^{\rho}X_2, g_1^sY_2)$ and T and simulates $\mathbf{Game_q}$ or $\mathbf{Game_{Final}}$ with $\mathcal{A}$ depending on T.

Setup : $\mathcal{B}$ chooses random $(u_{i,j} \leftarrow \mathbb{Z}_N)_{i\in[L],j\in\{0,1\}}$, and set $\Omega = \hat{e}(g_1^{\rho}X_2, g_1)$. Then $\mathcal{B}$ set

$$mpk \leftarrow [N, g_1, g_3, (g_1^{u_{i,j}})_{i\in[L],j\in\{0,1\}}, \Omega] \ .$$

as the public parameters. $\mathcal{B}$ starts the interaction with $\mathcal{A}$ on input mpk.

Secret Keys : For pattern $P = (P_1, \dots, P_\ell)$ where $0 \le \ell \le L$, $\mathcal{B}$ creates a semi-functional secret key as follows: Choose random $z'_k \leftarrow \mathbb{Z}_N$, $(r_i, z'_{k,i} \leftarrow \mathbb{Z}_N)_{i\in\overline{\mathrm{W}}(P)}$, $W_0 \leftarrow \mathbb{G}_{p_3}$ and $(W_i \leftarrow \mathbb{G}_{p_3})_{i\in\overline{\mathrm{W}}(P)}$. Then set

$$K_0 = (g_1^{\rho}X_2)\cdot \prod_{i\in\overline{\mathrm{W}}(P)} g_1^{r_i\cdot(u_{i,0}+u_{i,1}\cdot P_i)} \cdot g_2^{z'_k}\cdot W_0 \ ,$$

and

$$\left(D_i = g_1^{r_i}\cdot g_2^{z'_{k,i}}\cdot W_i\right)_{i\in\overline{\mathrm{W}}(P)} \ .$$

Challenge : $\mathcal{A}$ sends $\mathcal{B}$ challenge messages $\mathfrak{m}_0, \mathfrak{m}_1 \in \{0,1\}^*$ and a challenge pattern $P = (P_1, \dots, P_\ell)$ where $0 \le \ell \le L$. $\mathcal{B}$ chooses random $\eta \leftarrow \{0,1\}$ and computes the challenge ciphertext C as follows:

$$C_1 = (g_1^sY_2) \ ,$$
$$\left(C_{2,i} = (g_1^sY_2)^{u_{i,0}+u_{i,1}\cdot P_i}\right)_{i\in\overline{\mathrm{W}}(P)} \ ,$$
$$C_3 = T\cdot\mathfrak{m} \ ,$$
$$(C_{4,i,j} = (g_1^sY_2)^{u_{i,j}})_{i\in\mathrm{W}(P),j\in\{0,1\}} \ .$$

This sets $z_{c,i} = u_{i,0} + u_{i,1}\cdot P_i$ and $z_{c,i,j} = u_{i,j}$. We note that $g_1^{u_{i,j}}$ are elements of $\mathbb{G}_{p_1}$, so when $u_{i,j}$ are randomly chosen from $\mathbb{Z}_N$, their value modulo p_1 and modulo p_2 are random and independent.

We finish by observing that, if $T = \hat{e}(g_1, g_1)^{\rho s}$, then the challenge ciphertext is a properly distributed semi-functional with message $\mathfrak{m}_\eta$. If $T \leftarrow \mathbb{G}_T$, then the ciphertext is an semi-functional ciphertext with a random message. ▌

Wrapping up. We have proved by the previous lemmata that the real security game is indistinguishable from $\mathbf{Game_{Final}}$, where η is information-theoretically hidden from the attacker. Thus under assumptions LW.1-3, Theorem 4.2.4 holds.

4.3 WKD-IPE and Its application to Anonymous WW-IBE

In this section, we first introduce inner-product encryption with generalized key delegation (WKD-IPE, for short) and show how to generically reduce Anonymous WW-IBE to it in a way that preserves full security. The reduction is based on an observation by [KSW08] that zero entries can be used to simulate wildcards. The generalized key delegation capability of the WKD-IPE scheme will provide the final ingredient of the reduction. Finally, we give a new WKD-IPE construction based on a slight modification of the Okamoto-Takashima hierarchical inner-product predicate encryption scheme [LOS$^+$10], which allows for more general key delegation patterns.

4.3.1 Definition of Inner-Product Encryption with Generalized Key Delegation

In this section, following [OT09, LOS$^+$10], we define the WKD-IPE primitive and its security. We start by extending Definition 2.1.2 of hierarchical format to more general patterns. The main difference with Definition 2.1.2 is that we do not require any more that each Σ_i does not contain the all zero vector. Specifically:

Definition 4.3.1 [Generalized Hierarchical Format] Let $\boldsymbol{\mu}$ be a tuple of positive integers $\boldsymbol{\mu} = (n, d; \mu_1, \ldots, \mu_d)$ such that $\mu_0 = 0 < \mu_1 < \cdots < \mu_d = n$. For $i = 1, \ldots, d$, let $\Sigma_i = \mathbb{F}_p^{\mu_i - \mu_{i-1}}$ the set of attributes. Let Σ be the hierarchical attributes $\Sigma = \cup_{i=1}^{d} (\Sigma_1 \times \ldots \times \Sigma_i) \setminus \{\mathbf{0}_n\}$, where the union is a disjoint union. Them, we call $\boldsymbol{\mu}$ an *generalized hierarchical format* of depth d for the attribute space Σ.

Delegation. Given $\boldsymbol{y}, \boldsymbol{y'} \in \Sigma$, we say that $\boldsymbol{y'}$ is a *delegation* of $\boldsymbol{y}$, in symbols $\boldsymbol{y'} \prec \boldsymbol{y}$, if and only if for each $i \in [d]$ we have $\boldsymbol{y}_i = \mathbf{0}$ or $\boldsymbol{y}_i = \boldsymbol{y'}_i$. Again as opposed to [OT09, LOS$^+$10] any $\boldsymbol{y}_i = \mathbf{0}$ can be replaced by a non-zero vector, for any $i \in [d]$.

Definition 4.3.2 [WKD-IPE] A inner-product encryption with generalized key delegation with hierarchical format $\boldsymbol{\mu}$ is defined by the following tuple of algorithms: $\mathsf{WKD-IPE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}, \mathsf{KeyDel})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, \boldsymbol{\mu})$ takes as input security parameter λ and generalized hierarchical format $\boldsymbol{\mu}$ and outputs public parameters mpk and master secret key msk.

$\mathsf{KeyGen}(msk, \boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_d) \in \Sigma)$ takes as input msk, an attribute vector $\boldsymbol{y}$ following hierarchy $\boldsymbol{\mu}$ and outputs a private key $sk_{\boldsymbol{y}}$.

$\mathsf{KeyDel}(mpk, sk_{\boldsymbol{y}}, \boldsymbol{y'} \in \Sigma)$ takes as input mpk, secret key for attribute vectors $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_d)$ and $\boldsymbol{y'} = (\boldsymbol{y'}_1, \ldots, \boldsymbol{y'}_d)$ such that $\boldsymbol{y'} \prec \boldsymbol{y}$, and outputs a secret key for attribute vector $(\boldsymbol{y'}_1, \ldots, \boldsymbol{y'}_d)$.

$\mathsf{Enc}(mpk, \mathfrak{m}, \boldsymbol{x})$ takes as input mpk, $\mathfrak{m}$ in some associated message space and an attribute vector $\boldsymbol{x}$, and outputs a ciphertext Ct.

$\mathsf{Dec}(mpk, \mathsf{Ct}, sk_{\boldsymbol{y}})$ takes as input mpk, ciphertext Ct and secret key $sk_{\boldsymbol{y}}$ and outputs the message $\mathfrak{m}$.

Correctness. We make the following consistency requirement. Suppose ciphertext Ct is obtained by running Enc on input mpk, message $\mathfrak{m}$ and attribute vector $\boldsymbol{x}$ and that $sk_{\boldsymbol{y}}$ is a secret key for attribute vector $\boldsymbol{y}$ obtained through a sequence of KeyGen and KeyDel calls using the same mpk. Then Dec, on input mpk, Ct and $sk_{\boldsymbol{y}}$, returns $\mathfrak{m}$, except with negligible probability, if and only if $f_{\boldsymbol{y}}(\boldsymbol{x}) = 1$.

4.3.2 Security Definition

We give complete form of the security definition following [SW08]. Our security definition captures semantic security and ciphertext anonymity by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. More specifically, game ANO-CPA$_{\mathcal{A}}^{WKD-IPE}(1^\lambda)$, where λ is the security parameter, is defined as follows.

Setup: $\mathcal{C}$ generates master public key and secret key by invoking the setup algorithm on input the security parameter λ and the generalized hierarchical format $\boldsymbol{\mu}$. Specifically, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, \boldsymbol{\mu})$. Then, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input mpk.

We let S denote the set of private keys that $\mathcal{C}$ has created but not yet given to the adversary. At this point, $S = \emptyset$.

Query Phase 1: $\mathcal{A}$ makes *Create*, *Delegate*, and *Reveal* key queries. Specifically:

To make a *Create* query, $\mathcal{A}$ specifies an attribute vector $\boldsymbol{y}$. In response, $\mathcal{C}$ creates a key for this vector by calling the key generation algorithm, and places this key in the set S. $\mathcal{C}$ only gives $\mathcal{A}$ a reference to this key, not the key itself.

To make a *Delegate* query, $\mathcal{A}$ specifies a key $sk_{\boldsymbol{y}}$ in the set S and $\boldsymbol{y'}$. In response, $\mathcal{C}$ makes a key for this new vector by running the delegation algorithm on $sk_{\boldsymbol{y}}$ and $\boldsymbol{y'}$. $\mathcal{C}$ adds this key to the set S and again gives $\mathcal{A}$ only a reference to it, not the actual key.

To make a *Reveal* query, $\mathcal{A}$ specifies an element of the set S. $\mathcal{C}$ gives this key to $\mathcal{A}$ and removes it from the set S. Notice that $\mathcal{A}$ needs no longer make any delegation queries for this key because it can run delegation algorithm on the revealed key for itself

Challenge: $\mathcal{A}$ gives two pairs of message and identity $(\mathfrak{m}_0, \boldsymbol{y}_0)$ and $(\mathfrak{m}_1, \boldsymbol{y}_1)$ to $\mathcal{C}$. Then $\mathcal{C}$ chooses random $\eta \leftarrow \{0,1\}$, encrypts $\mathfrak{m}_\eta$ under $\boldsymbol{y}_\eta$ and sends the resulting ciphertext to $\mathcal{A}$.

Query Phase 2: This is the same as Phase 1.

Guess: Eventually, $\mathcal{A}$ submits its guess η'.

Winning Condition: $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. $\eta = \eta'$.
2. No secret key for any vector $\boldsymbol{w}$ obtained through delegation from any of the revealed secret key is such that $f_{\boldsymbol{w}}(\boldsymbol{x}_0) \neq f_{\boldsymbol{w}}(\boldsymbol{x}_1)$.

The *advantage* of $\mathcal{A}$ in the above game is defined as

$$\mathbf{Adv}^{WKD-IPE,\mathcal{A}}_{\text{ANO-CPA}}(\lambda) = \text{Prob}[\text{ANO-CPA}^{WKD-IPE}_{\mathcal{A}}(1^\lambda) = 1] - 1/2\ .$$

Definition 4.3.3 A inner-product encryption with generalized key delegation scheme is ANO-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the ANO-CPA game described above, meaning that $\mathbf{Adv}^{WKD-IPE,\mathcal{A}}_{\text{ANO-CPA}}(\lambda)$ is a negligible function of λ.

In the Section 4.4, we will describe a construction of inner-product encryption with generalized key delegation that achieves a weaker security definition called wAH-ANO-CPA and defined in Section 2.2.1. Specifically, game wAH-ANO-CPA is like ANO-CPA but with a relaxed winning condition. Specifically, wAH-ANO-CPA only requires that $f_{\boldsymbol{w}}(\boldsymbol{x}_0) = f_{\boldsymbol{w}}(\boldsymbol{x}_1) = 0$ for any vector $\boldsymbol{w}$ for which the relative secret key has been obtained through delegation from one of the revealed secret key.

Definition 4.3.4 A inner-product encryption with generalized key delegation scheme is wAH-ANO-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the wAH-ANO-CPA game, meaning that $\mathbf{Adv}^{WKD-IPE,\mathcal{A}}_{\text{wAH-ANO-CPA}}(\lambda)$ is a negligible function of λ.

4.3.3 Reducing Anonymous WW-IBE to WKD-IPE

Let $\mathsf{WKD-IPE} = (\mathsf{Setup}_{\mathsf{WKD}}, \mathsf{KeyGen}_{\mathsf{WKD}}, \mathsf{KeyDel}_{\mathsf{WKD}}, \mathsf{Enc}_{\mathsf{WKD}}, \mathsf{Dec}_{\mathsf{WKD}})$ be a Inner-product encryption with generalized key delegation. We can construct the scheme $\mathsf{WW-IBE} = (\mathsf{Setup}_{\mathsf{WW}}, \mathsf{KeyGen}_{\mathsf{WW}}, \mathsf{KeyDel}_{\mathsf{WW}}, \mathsf{Enc}_{\mathsf{WW}}, \mathsf{Dec}_{\mathsf{WW}})$ as follows:

$\mathsf{Setup}_{\mathsf{WW}}(1^\lambda, 1^L)$: The algorithm returns the output of $\mathsf{Setup}_{\mathsf{WKD}}(1^\lambda, \boldsymbol{\mu} = (2L, L; (\mu_i = 2i)_{i\in[L]}))$. So the hierarchy $\boldsymbol{\mu}$ is of depth L and each level has dimension 2.

$\mathsf{KeyGen}_{\mathsf{WW}}(msk, P)$: For a pattern $P = (P_1, \dots, P_\ell)$, the key generation algorithm constructs vector $\boldsymbol{y} \in \Sigma$ by setting, for each $i \in [\ell]$,

$$\boldsymbol{y}_i = \begin{cases} (1, P_i) & \text{if } P_i \neq \star \\ (0,0) & \text{otherwise} \end{cases} .$$

We denote this transformation by

$$\boldsymbol{y} = \mathsf{KEncode}(P) .$$

Then the key generation algorithm returns

$$sk_P = \mathsf{KeyGen}_{\mathsf{WKD}}(msk, \boldsymbol{y}) .$$

$\mathsf{KeyDel}_{\mathsf{WW}}(mpk, sk_P, P')$: The algorithm returns the output of

$$sk_{P'} = \mathsf{KeyDel}_{\mathsf{WKD}}(mpk, sk_P, \mathsf{KEncode}(P')) .$$

$\mathsf{Enc}_{\mathsf{WW}}(mpk, P)$: The algorithm constructs vector $\boldsymbol{x} \in \Sigma$ in the following way: For each $i \in [\ell]$ the algorithms sets

$$\boldsymbol{x}_i = \begin{cases} (-r_i \cdot P_i, r_i) & \text{if } P_i \neq \star \\ (0,0) & \text{otherwise} \end{cases} .$$

We denote this transformation by

$$\boldsymbol{x} = \mathsf{CEncode}(P) .$$

Then the encryption algorithm returns

$$\mathsf{Ct} = \mathsf{Enc}_{\mathsf{WKD}}(mpk, \boldsymbol{x}) .$$

$\mathsf{Dec}_{\mathsf{WW}}(sk_{P'}, \mathsf{Ct})$: Returns $\mathsf{Dec}_{\mathsf{WKD}}(sk_{P'}, \mathsf{Ct})$.

Correctness. It follows from the observation that for patterns P and P', we have that $f_{\mathsf{KEncode}(P')}(\mathsf{CEncode}(P)) = 1$ if and only if $P \in_\star P'$.

Theorem 4.3.5 If $\mathsf{WKD-IPE}$ is ANO-CPA secure then $\mathsf{WW-IBE}$ is so.

Proof: Let $\mathcal{A}$ be an adversary that breaks the ANO-CPA security of the $\mathsf{WW-IBE}$ scheme for an hierarchy of depth ℓ and consider the following adversary $\mathcal{B}$ that tries to break the ANO-CPA security of the $\mathsf{WKD-IPE}$ scheme for hierarchy format $\boldsymbol{\mu} = (2\ell, \ell; (\mu_i = 2i)_{i\in[\ell]})$ that uses $\mathcal{A}$ as a subroutine and interact with the challenger $\mathcal{C}$.

Then, $\mathcal{B}$ receives from $\mathcal{C}$ a master public key mpk for the $\mathsf{WKD}-\mathsf{IPE}$ scheme and passes it to $\mathcal{A}$. Whenever $\mathcal{A}$ asks for the key for pattern P, $\mathcal{B}$ constructs $\boldsymbol{y} = \mathsf{KEncode}(P)$ and asks $\mathcal{C}$ for a key $sk_{\boldsymbol{y}}$ for $\boldsymbol{y}$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge ciphertext by providing $(\mathfrak{m}_0, P_0^\star)$ and $(\mathfrak{m}_1, P_1^\star)$, $\mathcal{B}$ simply computes $\boldsymbol{x}_0 = \mathsf{CEncode}(P_0^\star)$ and $\boldsymbol{x}_1 = \mathsf{CEncode}(P_1^\star)$ and gives the pair $(\mathfrak{m}_0, \boldsymbol{x}_0), (\mathfrak{m}_1, \boldsymbol{x}_1)$ to $\mathcal{C}$. $\mathcal{B}$ then forwards the challenge ciphertext to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s guess.

First, $\mathcal{B}$'s simulation is perfect. Indeed, we have that if for all $\mathcal{A}$'s queries satisfy the game constraints, then all $\mathcal{B}$'s queries have the same property. Thus $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. ▮

4.4 Constructing WKD-IPE

In this section, we show our new WKD-IPE construction and prove its security. Our construction is based on a slight modification of the Okamoto-Takashima hierarchical predicate encryption scheme for the class of the inner-product predicates [LOS+10]. As the HPE scheme in [LOS+10], our scheme will be proved to be only *weakly attribute-hiding* secure.

4.4.1 Complexity Assumptions

For completeness we restate here the complexity assumptions used, first introduced by Okamoto and Takashima in [LOS+10]. All these assumptions can be seen as variants of the decisional subspace assumption.

Assumption OT.1 For a given prime-order dual-pairing vector spaces generator $\mathsf{DPVSGen}$, define the following distribution: Pick a random bilinear setting $(\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^\star) \leftarrow \mathsf{DPVSGen}(1^\lambda, 2n+3)$, and then pick

$$\delta \leftarrow \mathbb{Z}_N, \boldsymbol{\delta} \leftarrow \mathbb{Z}_N^n, \rho \leftarrow \mathbb{Z}_N, \boldsymbol{U} = (\boldsymbol{u}_i)_{i\in[n]} \leftarrow \mathsf{GL}(n, \mathbb{Z}_N)\ ,$$

and set

$$\hat{\mathbb{B}} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{b}_{2n+1}, \boldsymbol{b}_{2n+3}), \quad \hat{\mathbb{B}}^\star = (\boldsymbol{b}_1^\star, \ldots, \boldsymbol{b}_n^\star, \boldsymbol{b}_{2n+1}^\star, \boldsymbol{b}_{2n+2}^\star)\ ,$$

$$\boldsymbol{e}_0 = (\delta \cdot \boldsymbol{b}_i + \delta_i \cdot \boldsymbol{b}_{2n+3})_{i\in[n]}, \quad \boldsymbol{e}_1 = (\delta \cdot \boldsymbol{b}_i + \rho \cdot \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_{n+j} + \delta_i \cdot \boldsymbol{b}_{2n+3})_{i\in[n]}\ ,$$

and $\mathcal{D} = (\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \hat{\mathbb{B}}^\star)$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking Assumption OT.1 is

$$\mathbf{Adv}_{OT.1}^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, \boldsymbol{e}_0) = 1] - \mathrm{Prob}[\mathcal{A}(D, \boldsymbol{e}_1) = 1]|\ .$$

Definition 4.4.1 We say that Assumption OT.1 holds for generator $\mathsf{DPVSGen}$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_{OT.1}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Assumption OT.2 For a given prime-order dual-pairing vector spaces generator $\mathsf{DPVSGen}$, define the following distribution: Pick a random bilinear setting $(\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^\star) \leftarrow \mathsf{DPVSGen}(1^\lambda, 2n+3)$, and then pick

$$\omega, \gamma, \rho, \tau \leftarrow \mathbb{Z}_N, \boldsymbol{\gamma} \leftarrow \mathbb{Z}_N^n, \boldsymbol{U} = (\boldsymbol{u}_i)_{i\in[n]} \leftarrow \mathsf{GL}(n, \mathbb{Z}_N)\ ,$$

and set

$$\hat{\mathbb{B}} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{b}_{2n+1}, \boldsymbol{b}_{2n+3}), \quad \hat{\mathbb{B}}^\star = (\boldsymbol{b}_1^\star, \ldots, \boldsymbol{b}_{2n+2}^\star)\ ,$$

$$\boldsymbol{Z} = (\boldsymbol{z}_i)_{i\in[n]} = (\boldsymbol{U}^{\mathsf{T}})^{-1}, \quad \boldsymbol{e} = (\delta \cdot \boldsymbol{b}_i + \rho \cdot \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_{n+j})_{i\in[n]} ,$$

$$\boldsymbol{h}_0 = (\omega \cdot \boldsymbol{b}_i^\star + \gamma_i \cdot \boldsymbol{b}_{2n+2}^\star)_{i\in[n]}, \quad \boldsymbol{h}_1 = (\omega \cdot \boldsymbol{b}_i^\star + \tau \cdot \sum_{j\in[n]} z_{i,j} \cdot \boldsymbol{b}_{n+j}^\star + \gamma_i \cdot \boldsymbol{b}_{2n+2})_{i\in[n]} ,$$

and $\mathcal{D} = (\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \hat{\mathbb{B}}^\star, \boldsymbol{e})$.

Then, the advantage of a PPT adversary $\mathcal{A}$ in breaking Assumption OT.2 is

$$\mathbf{Adv}_{OT.2}^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, \boldsymbol{h}_0) = 1] - \mathrm{Prob}[\mathcal{A}(D, \boldsymbol{h}_1) = 1]| .$$

Definition 4.4.2 We say that Assumption OT.2 holds for generator $\mathsf{DPVSGen}$ if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_{OT.2}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

Remark 4.4.3 In [LOS+10], Okamoto and Takashima proved that Assumptions OT.1 and OT.2 can be reduced to the n-Extended Decisional Diffie-Hellman Assumption. More recently in [OT10] Okamoto and Takashima proved that these kind of decisional subspace problems can be reduced to the Decision Linear Assumption [BBS04].

4.4.2 Our Construction

The algorithms of our $\mathsf{WKD} - \mathsf{IPE}$ are defined as follows:

$\mathsf{Setup}(1^\lambda, \mu = (n, d; \mu_1, \dots, \mu_d))$: Choose a random DPVS instance

$$(\mathsf{param}_{\mathbb{V}}, \mathbb{B}, \mathbb{B}^\star) \leftarrow \mathsf{DPVSGen}(1^\lambda, 2n+3) .$$

Then set $\hat{\mathbb{B}} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n, \boldsymbol{b}_{2n+1}, \boldsymbol{b}_{2n+3})$, $\Omega = \hat{e}(g, g)$. The master public key is defined as

$$mpk \leftarrow [\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \Omega] ,$$

and the master secret key

$$msk \leftarrow [\mathbb{B}^\star] .$$

$\mathsf{KeyGen}(msk, \boldsymbol{y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_d))$: Given $\boldsymbol{y}$ according to hierarchy μ, generate the following sub-keys:

Decryption Key : First apply a random translation to $\boldsymbol{y}$ choosing $(r_i \leftarrow \mathbb{Z}_N)_{i\in[d]}$ so that

$$\hat{\boldsymbol{y}} = (\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_d) = (r_1 \cdot \boldsymbol{y}_1, \dots, r_d \cdot \boldsymbol{y}_d) .$$

Then choose $r \leftarrow \mathbb{Z}_N$ and computes the decryption key:

$$sk^{dec} = (\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_d, \mathbf{0}_n, 1, r, 0)_{\mathbb{B}^\star} .$$

Randomization Keys : Let $S_{\boldsymbol{y}} = \{k \in [d] : \boldsymbol{y}_k \neq \mathbf{0}\}$ and $T_{\boldsymbol{y}} = S_{\boldsymbol{y}} \bigcup \{d+1\}$. For each $k \in T_{\boldsymbol{y}}$ apply a random translation to $\boldsymbol{y}$ choosing $(s_{k,i} \leftarrow \mathbb{Z}_N)_{i\in[d]}$ so that

$$\hat{\boldsymbol{y}}_k^{ran} = (\hat{\boldsymbol{y}}_{k,1}^{ran}, \dots, \hat{\boldsymbol{y}}_{k,d}^{ran}) = (s_{k,1} \cdot \boldsymbol{y}_1, \dots, s_{k,d} \cdot \boldsymbol{y}_d) .$$

Then choose $s_k \leftarrow \mathbb{Z}_N$ and compute the randomization keys:

$$sk_k^{ran} = (\hat{\boldsymbol{y}}_{k,1}^{ran}, \dots, \hat{\boldsymbol{y}}_{k,d}^{ran}, \mathbf{0}_n, 0, s_k, 0)_{\mathbb{B}^\star} .$$

Delegation Keys : Let $D_{\boldsymbol{y}} = \{j \in [n] : \exists k \in [d] \text{ s.t. } \boldsymbol{y}_k = \mathbf{0} \text{ and } \mu_{k-1} + 1 \leq j \leq \mu_k\}$. Choose $t \leftarrow \mathbb{Z}_N$. Then for each $j \in D_{\boldsymbol{y}}$, let k be the index such that $\boldsymbol{y}_k = \mathbf{0}$ and $\mu_{k-1} + 1 \leq j \leq \mu_k$, apply a random translation to $\boldsymbol{y}$ choosing $(t_{j,i} \leftarrow \mathbb{Z}_N)_{i \in [d]}$ so that

$$\hat{\boldsymbol{y}}_j^{del} = (\hat{\boldsymbol{y}}_{j,1}^{del}, \ldots, \hat{\boldsymbol{y}}_{j,d}^{del}) = (t_{k,1} \cdot \boldsymbol{y}_{j,1}^{del}, \ldots, t_{k,d} \cdot \boldsymbol{y}_{j,d}^{del}) ,$$

where $\hat{\boldsymbol{y}}_{j,i}^{del} = t \cdot \boldsymbol{e}_j$ if $j = i_k$, otherwise $\hat{\boldsymbol{y}}_{j,i}^{del} = t_{j,i} \cdot \boldsymbol{y}_i$ where $\boldsymbol{e}_j$ is the zero vector with 1 in the j-th coordinate.

Then choose $t_j \leftarrow \mathbb{Z}_N$ and compute the delegation keys:

$$sk_j^{del} = (\hat{\boldsymbol{y}}_{j,1}^{del}, \ldots, \hat{\boldsymbol{y}}_{j,d}^{del}, \mathbf{0}_n, 0, t_j, 0)_{\mathbb{B}^\star} .$$

The secret key for $\boldsymbol{y}$ is $sk_{\boldsymbol{y}} \leftarrow [sk^{dec}, (sk_k^{ran})_{k \in T_{\boldsymbol{y}}}, (sk_j^{del})_{j \in D_{\boldsymbol{y}}}]$. Notice that $sk_{\boldsymbol{y}}$ contains at most $n+1$ sub-keys.

$\mathsf{KeyDel}(msk, d_{\boldsymbol{y}}, \boldsymbol{y}' = (\boldsymbol{y}'_1, \ldots, \boldsymbol{y}'_d))$: Given $\boldsymbol{y}'$ according to the hierarchy μ, Let $\overline{S_{\boldsymbol{y}}} = \{i \in [d] : \boldsymbol{y}_i = \mathbf{0}\}$ and let $E_k = \{i \in [n] : \mu_{k-1} + 1 \leq i \leq \mu_k\}$. Then the key delegation algorithm does the following:

Decryption Key : Chooses $r' \leftarrow \mathbb{Z}_N$, $(r'_j \leftarrow \mathbb{Z}_N)_{j \in [d]}$ and computes

$$sk'^{dec} = sk^{dec} + \sum_{j \in T_{\boldsymbol{y}}} r'_j \cdot sk_j^{ran} + \sum_{j \in \overline{S_{\boldsymbol{y}}}} \left[r'_j \cdot \sum_{i \in E_j} y'_i \cdot sk_i^{del} \right] .$$

Randomization Keys : For each $k \in T_{\boldsymbol{y}'}$ chooses $s'_k \leftarrow \mathbb{Z}_N$ and $(s'_{k,j} \leftarrow \mathbb{Z}_N)_{j \in [d]}$. Then computes

$$sk_k'^{ran} = \sum_{j \in T_{\boldsymbol{y}}} s'_{k,j} \cdot sk_j^{ran} + \sum_{j \in \overline{S_{\boldsymbol{y}}}} \left[s'_{k,j} \cdot \sum_{i \in E_j} y'_i \cdot sk_i^{del} \right] .$$

Delegation Keys : Chooses $t' \leftarrow \mathbb{Z}_N$ and for each $j \in D_{\boldsymbol{y}'}$, chooses $(t'_{j,i} \leftarrow \mathbb{Z}_N)_{i \in [n]}$, $t'_j, \leftarrow \mathbb{Z}_N$. Then computes

$$sk_j'^{del} = \sum_{i \in T_{\boldsymbol{y}}} t'_{j,i} \cdot sk_i^{ran} + \sum_{i \in \overline{S_{\boldsymbol{y}}}} \left[t'_{j,i} \cdot \sum_{k \in E_j} y'_k \cdot sk_k^{del} \right] + t' \cdot sk_j^{del} .$$

Thus $sk_{\boldsymbol{y}'} = [sk'^{dec}, (sk_k'^{ran})_{k \in T_{\boldsymbol{y}'}}, (sk_j'^{del})_{j \in D_{\boldsymbol{y}'}}]$.

$\mathsf{Enc}(mpk, \boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell), \mathfrak{m})$: Choose $s \leftarrow \mathbb{Z}_N$, $(s_i \leftarrow \mathbb{Z}_N)_{i \in [d]}$ and pads $\boldsymbol{x}$ to the maximum length d with random vectors following the hierarchy μ. Then apply a random translation to $\boldsymbol{x}$ so that $\hat{\boldsymbol{x}} = (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_d) = (s_1 \cdot \boldsymbol{x}_1, \ldots, s_d \cdot \boldsymbol{x}_d)$, and compute

$$C_1 = (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_d, \mathbf{0}_n, \alpha, 0, s)_{\mathbb{B}} \text{ and } C_2 = \Omega^\alpha \cdot \mathfrak{m} .$$

The ciphertext for vector x and message $\mathfrak{m}$ is $\mathsf{Ct} \leftarrow [C_1, C_2]$.

$\mathsf{Dec}(mpk, \mathsf{Ct}, sk_y)$: The decryption algorithms computes the blinding

$$\hat{e}(C_1, sk^{dec}) = \hat{e}(g, g)^{\alpha + \sum_{i \in [d]} \langle \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}_i \rangle}$$

Remark 4.4.4 Notice that we need $|T_{\boldsymbol{y}}|$ randomization sub-keys because in order to re-randomize the decryption key it is necessary to generate $|T_{\boldsymbol{y}}|$ random coefficients, one for each $i \in S_{\boldsymbol{y}}$ plus r. By taking a random linear combination of the randomization sub-keys we achieve the task.

4.4.3 Security

We will prove the following main theorem:

Theorem 4.4.5 If Assumptions OT.1 and OT.2 hold then our WKD-IPE scheme is secure in the sense of Definition 4.3.4.

Following [LW10], we define two additional structures that will be used in the proof of security. Specifically:

Semi-Functional Ciphertext : An semi-functional ciphertext has additional dimensions. More specifically, choose $\boldsymbol{r} \leftarrow \mathbb{Z}_N^n$ and set

$$C_1 = (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_d, \boldsymbol{r}, \alpha, 0, s)_{\mathbb{B}} .$$

Semi-Functional Keys : A secret key is semi-functional if all its sub-keys are semi-functional. More specifically:

Decryption Key :

$$sk^{sf.dec} = (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_d, \boldsymbol{r}^{dec}, 1, r, 0)_{\mathbb{B}^\star} ,$$

Randomization Keys :

$$sk_k^{sf.ran} = (\hat{\boldsymbol{y}}_{k,1}^{ran}, \ldots, \hat{\boldsymbol{y}}_{k,d}^{ran}, \boldsymbol{r}_k^{ran}, 0, s_k, 0)_{\mathbb{B}^\star} ,$$

Delegation Keys :

$$sk_j^{sf.del} = (\hat{\boldsymbol{y}}_{j,1}^{del}, \ldots, \hat{\boldsymbol{y}}_{j,d}^{del}, \boldsymbol{r}_j^{del}, 0, t_j, 0)_{\mathbb{B}^\star} ,$$

where $\boldsymbol{r}^{dec}, \boldsymbol{r}_k^{ran}, \boldsymbol{r}_j^{del} \leftarrow \mathbb{Z}_N^n$.

Remark 4.4.6 Notice that semi-functional ciphertexts and semi-functional secret keys lead to a random decryption with high probability. In fact with high probability the $\langle \boldsymbol{r}, \boldsymbol{r}^{dec} \rangle \neq 0$ given that $\boldsymbol{r}$ and $\boldsymbol{r}^{dec}$ are uniformly and independently distributed. If $\boldsymbol{r}$ and $\boldsymbol{r}^{dec}$ were correlated such that

$$\boldsymbol{r} = \rho \cdot \hat{\boldsymbol{x}} \cdot \boldsymbol{U} \quad \text{and} \quad \boldsymbol{r}^{dec} = \tau \cdot \hat{\boldsymbol{y}} \cdot \boldsymbol{Z} ,$$

where $\rho, \tau \leftarrow \mathbb{Z}_N$, $\boldsymbol{U} = (u_{i,j} \leftarrow \mathbb{Z}_N)_{i \in [n], j \in [n]}$ and $\boldsymbol{Z} = (\boldsymbol{U}^\top)^{-1}$, then the decryption would succeed for all vectors $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ such that their inner-product is 0.

For a PPT adversary $\mathcal{A}$ which makes q key queries, our proof of security will consist of the following $q \cdot (n+1) + 3$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$.

$\mathbf{Game_{Real}}$: It is the real Inner-product encryption with generalized key delegation security game.

$\mathbf{Game_{WD}}$: It it the same as $\mathbf{Game_{Real}}$ except that all key queries will be answered by fresh calls to the key generation.

$\mathbf{Game_0}$: It is the same as $\mathbf{Game_{WD}}$ except that the challenge ciphertext given to $\mathcal{A}$ is semi-functional.

$\mathbf{Game_{i,j}}$: For $i \in [q]$ and $0 \leq j \leq n+1$, is the same as $\mathbf{Game_0}$ except that the first $i-1$ keys are semi-functional and the remaining $q-i$ are normal. For the i-th key, only the first j sub-keys are semi-functional and the rest of the sub-keys are normal. Thus $\mathbf{Game_{1,0}} = \mathbf{Game_0}$ and $\mathbf{Game_{i,n+1}} = \mathbf{Game_{i+1,0}}$.

$\mathbf{Game_{Final}}$: It is the same as $\mathbf{Game_{q,n+1}}$ expect that the challenge ciphertext is for a random message and random attribute vector. Notice that in $\mathbf{Game_{Final}}$ the value of η is information-theoretically hidden from the attacker. Hence the attacker has no advantage in breaking the scheme.

For completeness we restate some useful lemmas whose proofs can be found in [LOS$^+$10].

Lemma 4.4.7 Let $C = \{(\boldsymbol{x}, \boldsymbol{y}) \mid \langle \boldsymbol{x}, \boldsymbol{y} \rangle \neq 0\} \subset V \times V^\star$ where V is an n-dimensional vector space over final field $\mathbb{F}_N$ and $V^\star$ its dual. Then for all $(\boldsymbol{x}, \boldsymbol{y}) \in C$ and for all $(\boldsymbol{r}, \boldsymbol{w}) \in C$ we have that $\Pr_{\boldsymbol{Z} \leftarrow \mathsf{GL}(n, \mathbb{F}_N), \rho, \tau \leftarrow \mathbb{F}_N^\times} [\boldsymbol{x}(\rho \boldsymbol{U}) = \boldsymbol{r} \text{ and } \boldsymbol{v}(\tau \boldsymbol{Z}) = \boldsymbol{w}] = \frac{1}{|C|}$ where $\boldsymbol{U} = (\boldsymbol{Z}^\mathsf{T})^{-1}$.

Lemma 4.4.8 Let $N > 2$ and $\Delta = \{\boldsymbol{M} \mid \det \boldsymbol{M} = 0\} \subset \mathbb{F}_N^{n \times n}$. Then $|\Delta|/N^{n^2} < 2/N$

We give now formal proofs of the above statements.

Lemma 4.4.9 For any algorithm $\mathcal{A}$, $|\mathbf{Adv}_{\mathcal{A}}^{\mathbf{Game_{Real}}} - \mathbf{Adv}_{\mathcal{A}}^{\mathbf{Game_{WD}}}| \leq 3q/N$.

Proof: We show that the distribution of $sk_{\boldsymbol{y'}} \leftarrow \mathsf{KeyGen}(msk, \boldsymbol{y'})$ is equivalent to that of $\mathsf{KeyDel}(mpk, sk_{\boldsymbol{y}}, \boldsymbol{y'})$, for $sk_{\boldsymbol{y}} = \mathsf{KeyGen}(msk, \boldsymbol{y})$ such that $\boldsymbol{y'}$ can be delegated from $\boldsymbol{y}$, except with probability $3/N$. Notice that the distribution of the secret key $sk_{\boldsymbol{y'}}$ is represented by the coefficients of the square matrix $\boldsymbol{S} = ((s_{k.i})_{i \in S_{\boldsymbol{y'}}}, s_k)_{k \in T_{\boldsymbol{y'}}}$ (which is a $|T_{\boldsymbol{y'}}| \times |T_{\boldsymbol{y'}}|$ matrix in $\mathbb{Z}_N$) and by t used in the delegation sub-keys, which are all uniformly and independently distributed. By Lemma 4.4.8 the span of matrix $\boldsymbol{S}_{\boldsymbol{y'}}$ is $\mathbb{Z}_N^{|T_{\boldsymbol{y'}}|}$ except with probability $2/N$ and the coefficient t is different from 0 except with probability $1/N$. Thus the distribution of $\mathsf{KeyDel}(mpk, sk_{\boldsymbol{y}}, \boldsymbol{y'})$ is equivalent to that of $sk_{\boldsymbol{y'}}$ except with probability $2/N + 1/N = 3/N$. The Shoup's difference lemma gives us the upper bound. ▌

Lemma 4.4.10 Suppose there exists an algorithm $\mathcal{A}$ such that $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{Game_{WD}}} - \mathbf{Adv}^{\mathcal{A}}_{\mathbf{Game_0}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption OT.1.

Proof: We describe algorithm $\mathcal{B}$ that breaks Assumption OT.1. $\mathcal{B}$ receives $(\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \hat{\mathbb{B}}^\star)$ and $\boldsymbol{e}_\beta$. Depending on the value of β, $\mathcal{B}$ simulates $\mathbf{Game_{WD}}$ or $\mathbf{Game_0}$.

Setup : $\mathcal{B}$ sets $\Omega = \hat{e}(g, g)$ and gives $\mathcal{A}$ $mpk \leftarrow [\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \Omega]$ and keeps $msk \leftarrow [\hat{\mathbb{B}}^\star]$ secret.

Secret Keys : $\mathcal{B}$ uses the msk to generate secret keys.

Challenge : $\mathcal{A}$ sends $\mathcal{B}$ the two challenges $((\mathfrak{m}^{(0)}, \boldsymbol{x}^{(0)}), (\mathfrak{m}^{(1)}, \boldsymbol{x}^{(1)}))$. $\mathcal{B}$ chooses $\eta \leftarrow \{0, 1\}$ and then encrypts $(\mathfrak{m}^{(\eta)}, x^{(\eta)})$ as follows: $\mathcal{B}$ chooses $\alpha \leftarrow \mathbb{Z}_N$, $(s_i \leftarrow \mathbb{Z}_N)_{i \in [d]}$ and pads $\boldsymbol{x}^\eta$ to the maximum length d with random vectors following the hierarchy $\boldsymbol{\mu}$. Then $\mathcal{B}$ applies a random translation to $\boldsymbol{x}^\eta$ so that

$$\begin{aligned} \hat{\boldsymbol{x}}^{(\eta)} &= (\hat{\boldsymbol{x}}_1^{(\eta)}, \ldots, \boldsymbol{x}_d^{(\eta)}) \\ &= (s_1 \cdot \boldsymbol{x}_1^{(\eta)}, \ldots, s_d \cdot \boldsymbol{x}_d^{(\eta)}) \end{aligned},$$

and computes

$$C_1 = \sum_{i \in [n]} \hat{x}_i^{(\eta)} \cdot \boldsymbol{e}_{\beta,i} + \alpha \cdot \boldsymbol{b}_{2n+1} \text{ and } C_2 = \Omega^\alpha \cdot \mathfrak{m}^{(\eta)} .$$

Now, if $\beta = 0$ then

$$C_1 = (\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_d, \mathbf{0}_n, \alpha, 0, s)_{\mathbb{B}} \ ,$$

where $\bar{\boldsymbol{x}}_i = \delta \cdot s_i \cdot \boldsymbol{x}_i^{(\eta)}$ and $s = \langle \hat{\boldsymbol{x}}^{(\eta)}, \boldsymbol{\delta} \rangle$, and the challenge ciphertext is distributed as in $\mathbf{Game_{WD}}$. Instead if $\beta = 1$ then

$$C_1 = (\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_d, \boldsymbol{r}, \alpha, 0, s)_{\mathbb{B}} \ ,$$

where $\bar{\boldsymbol{x}}_i = \delta \cdot s_i \cdot \boldsymbol{x}_i^{(\eta)}$, $\boldsymbol{r} = (r_1, \ldots, r_n)$ with $r_i = \rho \cdot \langle \hat{\boldsymbol{x}}^{(\eta)}, \boldsymbol{u}_i \rangle$, and $s = \langle \hat{\boldsymbol{x}}^{(\eta)}, \boldsymbol{\delta} \rangle$. Because $\hat{\boldsymbol{x}}^{(\eta)} \neq \mathbf{0}_n$ and that $(\boldsymbol{u}_i)_{i \in [n]}$ and $\boldsymbol{\delta}$ are uniformly and independently distributed then the challenge ciphertext is properly distributed as in $\mathbf{Game_0}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish the value of β . ▌

Lemma 4.4.11 Suppose there exists an algorithm $\mathcal{A}$ such that $\mathbf{Adv}^{\mathcal{A}}_{\mathbf{Game_{i,j-1}}} - \mathbf{Adv}^{\mathcal{A}}_{\mathbf{Game_{i,j}}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\epsilon + 1/N$ in breaking Assumption OT.2.

Proof: We describe algorithm $\mathcal{B}$ that breaks Assumption OT.2. $\mathcal{B}$ receives $(\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \hat{\mathbb{B}}^\star, \boldsymbol{e})$ and $\boldsymbol{h}_\beta$. Depending on the value of β, $\mathcal{B}$ simulates $\mathbf{Game_{i,j-1}}$ or $\mathbf{Game_{i,j}}$.

Setup : $\mathcal{B}$ sets $\Omega = \hat{e}(g, g)$ and gives $\mathcal{A}$ $mpk \leftarrow [\mathsf{param}_{\mathbb{V}}, \hat{\mathbb{B}}, \Omega]$ and sets $msk \leftarrow [\hat{\mathbb{B}}^\star]$.

Secret Keys : For the first $i-1$ secret key queries, $\mathcal{B}$ creates semi-functional secret keys in the following way: $\mathcal{B}$ invokes the key generation algorithm on input the master secret key msk and the given vector $\boldsymbol{y}$ to obtain the secret key $sk_{\boldsymbol{y}} \leftarrow [sk^{dec}, (sk_k^{ran})_{k \in T_{\boldsymbol{y}}}, (sk_k^{del})_{k \in D_{\boldsymbol{y}}}] \leftarrow \mathsf{KeyGen}(msk, \boldsymbol{y})$. Then $\mathcal{B}$ introduces the additional dimensions to create semi-functional secret key $sk_{\boldsymbol{y}}^{sf} \leftarrow [sk^{sf.dec}, (sk_k^{sf.ran})_{k \in T_{\boldsymbol{y}}}, (sk_k^{sf.del})_{k \in D_{\boldsymbol{y}}}]$ by using $\hat{\mathbb{B}}^\star$ as:

Decryption Key :

$$sk^{sf.dec} = sk^{dec} + \sum_{i \in [n]} r_i^{dec} \cdot \boldsymbol{b}^\star_{n+i} \ ,$$

Randomization Keys :

$$sk_k^{sf.ran} = sk_k^{ran} + \sum_{i \in [n]} r_{k,i}^{ran} \cdot \boldsymbol{b}^\star_{n+i} \ ,$$

Delegation Key :

$$sk_j^{sf.del} = sk_j^{del} + \sum_{i \in [n]} r_{j,i}^{del} \cdot \boldsymbol{b}^\star_{n+i} \ .$$

where $r_i^{dec}, r_{k,i}^{ran}, r_{j,i}^{del} \leftarrow \mathbb{Z}_N^n$.

Then, for the i-th secret key query on vector $\boldsymbol{y}$, the first $j-1$ sub-keys are generated semi-functional as before. The remaining $n-j$ sub-keys are normal, so without the semi-functional part. For the j-th sub-key of the i-th key, $\mathcal{B}$ does the following: Depending on the type of the j-th sub-key (decryption, randomization, delegation), $\mathcal{B}$ applies the necessary translation to the vector $\boldsymbol{y}$ to obtain the vector $\hat{\boldsymbol{y}}$. Then it encodes $\hat{\boldsymbol{y}}$ as:

$$sk_j = \sum_{i \in [n]} \hat{y}_i \cdot \boldsymbol{h}_{\beta,i} + \boldsymbol{b}^\star_{2n+1} \ .$$

The remaining $q - i$ secret key queries are answered by generating normal secret keys using msk.

Challenge : $\mathcal{A}$ sends $\mathcal{B}$ the tuple $((\mathfrak{m}^{(0)}, \boldsymbol{x}^{(0)}), (\mathfrak{m}^{(1)}, \boldsymbol{x}^{(1)}))$. $\mathcal{B}$ chooses $\eta \leftarrow \{0,1\}$ and encrypts $(\mathfrak{m}^{(\eta)}, \boldsymbol{x}^{(\eta)})$ as follows: $\mathcal{B}$ chooses random $\alpha, s \leftarrow \mathbb{Z}_N$, $(s_i \leftarrow \mathbb{Z}_N)_{i\in[d]}$ and pads $\boldsymbol{x}^{(\eta)}$ to the maximum length d with random vectors following the hierarchy μ. Then $\mathcal{B}$ applies a random translation to $\boldsymbol{x}^{(\eta)}$ so that

$$\begin{aligned}\hat{\boldsymbol{x}}^{(\eta)} &= (\hat{\boldsymbol{x}}_1^{(\eta)}, \dots, \hat{\boldsymbol{x}}_d^{(\eta)}) \\ &= (s_1 \cdot \boldsymbol{x}_1^{(\eta)}, \dots, s_d \cdot \boldsymbol{x}_d^{(\eta)})\end{aligned},$$

$\mathcal{B}$ computes

$$\begin{aligned}C_1 &= \sum_{i\in[n]} \hat{x}_i^{(\eta)} \cdot \boldsymbol{e}_i + \alpha \cdot \boldsymbol{b}_{2n+1} + s \cdot \boldsymbol{b}_{2n+3} \\ &= (\bar{\boldsymbol{x}}_1, \dots, \bar{\boldsymbol{x}}_d, \boldsymbol{t}, \alpha, 0, s)_{\mathbb{B}}\end{aligned},$$

and

$$C_2 = \Omega^\alpha \cdot \mathfrak{m}^{(\eta)} ,$$

where $\bar{\boldsymbol{x}}_i = \delta \cdot s_i \cdot \boldsymbol{x}_i^{(\eta)}$, $\boldsymbol{t} = (t_1, \dots, t_n)$ with $(t_i = \rho \cdot \langle \hat{\boldsymbol{x}}^{(\eta)}, \boldsymbol{u}_i \rangle)_{i\in[n]}$.

Now, if $\beta = 0$ then the j-th sub-key can be written as

$$sk_j = (\bar{\boldsymbol{y}}_1, \dots, \bar{\boldsymbol{y}}_d, \boldsymbol{0}_n, 1, r, 0)_{\mathbb{B}^\star} ,$$

where $\bar{\boldsymbol{y}}_i = \omega \cdot \hat{\boldsymbol{y}}_i$ and $r = \langle \hat{\boldsymbol{y}}, \boldsymbol{\gamma} \rangle$. Thus sk_j is a normal sub-key and the joint distribution of the challenge ciphertext and sk_j is as in $\mathbf{Game_{i,j-1}}$. Instead if $\beta = 1$ then the j-th sub-key can be written

$$sk_j = (\bar{\boldsymbol{y}}_1, \dots, \bar{\boldsymbol{y}}_d, \boldsymbol{r}, 1, r, 0)_{\mathbb{B}^\star} ,$$

where $\bar{\boldsymbol{y}}_i = \omega \cdot \hat{\boldsymbol{y}}_i$, $\boldsymbol{r} = (r_1, \dots, r_n)$ with $(r_i = \tau \cdot \langle \hat{\boldsymbol{y}}, \boldsymbol{z}_i \rangle)_{i\in[n]}$, $r = \langle \hat{\boldsymbol{y}}, \boldsymbol{\gamma} \rangle$. Since $\langle \hat{\boldsymbol{y}}, \hat{\boldsymbol{x}} \rangle \neq 0$, by the constraints of the security game and from Lemma 4.4.7, the coefficients $(t_i, r_i)_{i\in[n]}$ are pairwise-independently and uniformly distributed under the condition that $\langle \boldsymbol{t}, \boldsymbol{r} \rangle \neq 0$. Thus the joint distribution of the challenge ciphertext and sk_j is as in $\mathbf{Game_{i,j}}$ expect with probability $1/N$ that is the probability of the event $\langle \boldsymbol{t}, \boldsymbol{r} \rangle = 0$. ▌

Lemma 4.4.12 For any algorithm $\mathcal{A}$, $\mathbf{Game_{q,n+1}} = \mathbf{Game_{Final}}$.

Proof: We show that the view of $\mathcal{A}$ in $\mathbf{Game_{q,n+1}}$ is identical to the view of $\mathcal{A}$ in $\mathbf{Game_{Final}}$. Given the dual orthonormal bases $(\mathbb{B}, \mathbb{B}^\star)$, used in $\mathbf{Game_{q,n+1}}$, of $\mathbb{V}$, we will construct new dual orthonormal bases $\mathbb{D}, \mathbb{D}^\star$ of $\mathbb{V}$ such that:

1. $(\mathbb{D}, \mathbb{D}^\star)$ is distributed like $(\mathbb{B}, \mathbb{B}^\star)$,
2. $(\mathbb{D}, \mathbb{D}^\star)$ is consistent with the public key seen by $\mathcal{B}$ in both games $\mathbf{Game_{q,n+1}}$ and $\mathbf{Game_{Final}}$, and
3. the secret keys and the challenge ciphertext in $\mathbf{Game_{Final}}$ can be written in those bases.

Thus, given

$$\mathbb{B} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_n, \boldsymbol{b}_{n+1}, \dots, \boldsymbol{b}_{2n}, \boldsymbol{b}_{2n+1}, \boldsymbol{b}_{2n+2}, \boldsymbol{b}_{2n+3})$$

and

$$\mathbb{B}^\star = (\boldsymbol{b}_1^\star, \dots, \boldsymbol{b}_n^\star, \boldsymbol{b}_{n+1}^\star, \dots, \boldsymbol{b}_{2n}^\star, \boldsymbol{b}_{2n+1}^\star, \boldsymbol{b}_{2n+2}^\star, \boldsymbol{b}_{2n+3}^\star) ,$$

we define new dual orthonormal bases $\mathbb{D}, \mathbb{D}^\star$ of $\mathbb{V}$ as follow: First choose $\boldsymbol{U} = (\boldsymbol{u}_i \leftarrow \mathbb{Z}_N^n)_{i\in[n]}$ and $\boldsymbol{v} = (v_i \leftarrow \mathbb{Z}_N)_{i\in[n]}$, then define

$$\mathbb{D} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n, \boldsymbol{d}_{n+1}, \ldots, \boldsymbol{d}_{2n}, \boldsymbol{b}_{2n+1}, \boldsymbol{b}_{2n+2}, \boldsymbol{b}_{2n+3}) ,$$

and

$$\mathbb{D}^\star = (\boldsymbol{d}_1^\star, \ldots, \boldsymbol{d}_n^\star, \boldsymbol{b}_{n+1}^\star, \ldots, \boldsymbol{b}_{2n}^\star, \boldsymbol{d}_{2n+1}^\star, \boldsymbol{b}_{2n+2}^\star, \boldsymbol{b}_{2n+3}^\star) ,$$

where $(\boldsymbol{d}_i^\star = \boldsymbol{b}_i^\star + \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_{n+j}^\star)_{i\in[n]}$ and $(\boldsymbol{d}_{n+i} = \boldsymbol{b}_{n+i} - \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_j - v_i \cdot \boldsymbol{b}_{2n+1})_{i\in[n]}$ and $\boldsymbol{d}_{2n+1}^\star = \boldsymbol{b}_{2n+1}^\star + \sum_{j\in[n]} v_i \cdot \boldsymbol{b}_{n+j}^\star$.

It is easy to see that $(\mathbb{D}, \mathbb{D}^\star)$ are dual orthonormal because the transformation applied preserve this property, distributed like $(\mathbb{B}, \mathbb{B}^\star)$ and consistent with public key in both games (notice that $\boldsymbol{b}_{n+1}, \ldots, \boldsymbol{b}_{2n}$ are secret to the adversary). Now in $\mathbf{Game_{q,n+1}}$ each sub-key on vector $\boldsymbol{y}$ can be written as the encoding of a randomly translated vector $\hat{\boldsymbol{y}}$ (the translation depends on the sub-key type) in base $\mathbb{B}^\star$. This sub-key can be rewritten in base $\mathbb{D}^\star$, affecting only the semi-functional part, as follows:

$$\begin{aligned}
sk &= (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_d, \boldsymbol{r}, 1, r, 0)_{\mathbb{B}^\star} \\
&= \sum_{i\in[n]} \hat{y}_i \cdot \boldsymbol{b}_i^\star + \sum_{i\in[n]} r_i \cdot \boldsymbol{b}_{n+i}^\star + \boldsymbol{b}_{2n+1}^\star + r \cdot \boldsymbol{b}_{2n+2}^\star \\
&= \sum_{i\in[n]} \hat{y}_i \cdot \left(\boldsymbol{d}_i^\star - \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_{n+j}^\star \right) + \sum_{i\in[n]} r_i \cdot \boldsymbol{b}_{n+i}^\star + \left(\boldsymbol{d}_{2n+1}^\star - \sum_{j\in[n]} v_i \cdot \boldsymbol{b}_{n+j}^\star \right) + r \cdot \boldsymbol{b}_{2n+2}^\star , \\
&= \sum_{i\in[n]} \hat{y}_i \cdot \boldsymbol{d}_i^\star + \sum_{i\in[n]} (r_i - \langle \hat{\boldsymbol{y}}_i, \boldsymbol{u}_i \rangle - v_i) \cdot \boldsymbol{b}_{n+i}^\star + \boldsymbol{d}_{2n+1}^\star + r \cdot \boldsymbol{b}_{2n+2}^\star \\
&= (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_d, \boldsymbol{r}', 1, r, 0)_{\mathbb{D}^\star}
\end{aligned}$$

where $\boldsymbol{r}' = (r_1', \ldots, r_n')$ with $r_i' = r_i - \langle \hat{\boldsymbol{y}}_i, \boldsymbol{u}_i \rangle - v_i$. Notice that $\boldsymbol{r}'$ is uniformly distributed.

Moreover in $\mathbf{Game_{q,n+1}}$ the challenge ciphertext on vector $\boldsymbol{x} = \boldsymbol{x}^{(\eta)}$ can be written as the encoding of a randomly translated vector $\hat{\boldsymbol{x}}$ in base $\mathbb{B}$. This challenge ciphertext can be rewritten in base $\mathbb{D}$ as follows:

$$\begin{aligned}
C_1 &= (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_d, \boldsymbol{t}, \alpha, 0, t)_{\mathbb{B}} \\
&= \sum_{i\in[n]} \hat{x}_i \cdot \boldsymbol{b}_i + \sum_{i\in[n]} t_i \cdot \boldsymbol{b}_{n+i} + \alpha \cdot \boldsymbol{b}_{2n+1} + t \cdot \boldsymbol{b}_{2n+3} \\
&= \sum_{i\in[n]} \hat{x}_i \cdot \boldsymbol{b}_i + \sum_{i\in[n]} t_i \cdot \left(\boldsymbol{d}_{n+i} + \sum_{j\in[n]} u_{i,j} \cdot \boldsymbol{b}_j + v_i \cdot \boldsymbol{b}_{2n+1} \right) + \alpha \cdot \boldsymbol{b}_{2n+1} + t \cdot \boldsymbol{b}_{2n+3} , \\
&= \sum_{i\in[n]} (\hat{x}_i + \langle \boldsymbol{t}, \boldsymbol{u}_i \rangle) \cdot \boldsymbol{b}_i + \sum_{i\in[n]} t_i \cdot \boldsymbol{b}_{n+i} + (\alpha \cdot \langle \boldsymbol{t}, \boldsymbol{u} \rangle) \cdot \boldsymbol{b}_{2n+1} + t \cdot \boldsymbol{b}_{2n+3} \\
&= (\bar{\boldsymbol{x}}_1, \ldots, \bar{\boldsymbol{x}}_d, \boldsymbol{t}, \alpha', 0, t)_{\mathbb{D}}
\end{aligned}$$

affecting only $\hat{\boldsymbol{x}}$ and α and where $\bar{x}_i = \hat{x}_i + \langle \boldsymbol{t}, \boldsymbol{u}_i \rangle$ and $\alpha' = \alpha \cdot \langle \boldsymbol{t}, \boldsymbol{u} \rangle$. Notice that $\bar{\boldsymbol{x}}$ and α' are independently and uniformly distributed as requested in $\mathbf{Game_{Final}}$. Thus, the view of $\mathcal{A}$ in $\mathbf{Game_{q,n+1}}$ and $\mathbf{Game_{Final}}$ is the same. ▌

Wrapping up. We have proved by the previous lemmata that the real security game is indistinguishable from $\mathbf{Game_{Final}}$, where η is information-theoretically hidden from the attacker. Thus under assumptions OT.1 and OT.2, Theorem 4.4.5 holds.

Chapter 5

Lattice-based Hierarchical Inner Product Encryption

In this chapter we consider the problem of constructing hierarchical inner-product encryption scheme based on lattices assumptions. To achieve this goal, we extend the lattice-based IPE scheme by Agrawal et al. [AFV11] to the hierarchical setting by employing basis delegation technics by Peikert *et al.* [CHKP10] and by Agrawal *et al.* [ABB10]. As the underlying IPE scheme, our new scheme is shown to be selective weakly attribute-hiding secure based on the difficulty of the learning with errors problem in the standard model, as long as the total number of levels in the hierarchy is a constant.

5.1 Definitions

In this section, we introduce the hierarchical inner-product encryption primitive and its security. In doing so, we adopt the same notation and definition style used in [AFV11].

5.1.1 Hierarchical inner-product encryption

Definition 5.1.1 An *hierarchical inner-product encryption* for hierarchical format $\boldsymbol{\mu}$ (Definition 2.1.2) is defined by the following tuple of algorithms $\mathsf{HIPE} = (\mathsf{Setup}, \mathsf{Derive}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, \boldsymbol{\mu})$ Takes as input security parameter λ and hierarchical format $\boldsymbol{\mu}$ and outputs public parameters mpk and master secret key msk.

$\mathsf{Derive}(mpk, sk_{\boldsymbol{v}}, \boldsymbol{v}_t)$ Takes as input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1}) \in \Sigma_{|t-1}$, and a vector $\boldsymbol{v}_t \in \Sigma_t$, and outputs a secret key $sk_{\boldsymbol{v}'}$ for the vector $\boldsymbol{v'} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1}, \boldsymbol{v}_t)$. If $\boldsymbol{v}$ is the empty vector then $sk_{\boldsymbol{v}}$ is the master secret key msk.

$\mathsf{Enc}(mpk, \mathfrak{m}, \boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t) \in \Sigma_{|t})$ Takes as input public parameters $\mathfrak{m}$ in some associated message space, public parameters mpk and an attribute vector $\boldsymbol{w}$ and outputs a ciphertext Ct.

$\mathsf{Dec}(mpk, \mathsf{Ct}, sk_{\boldsymbol{v}})$ Takes as input public parameters mpk, ciphertext Ct and secret key $sk_{\boldsymbol{v}}$ and outputs the message $\mathfrak{m}$.

Correctness. We make the following consistency requirement. Suppose ciphertext Ct is obtained by running Enc on input mpk, message $\mathfrak{m}$ and attribute vector $\boldsymbol{w}$ and that $sk_{\boldsymbol{v}}$ is a secret key for attribute vector $\boldsymbol{v}$ obtained through a sequence of Derive calls using the same mpk. Then Dec, on input mpk, Ct and $sk_{\boldsymbol{v}}$, returns $\mathfrak{m}$, except with negligible probability, if and only if $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$.

5.1.2 Security Definition

The notion we achieve is the selective weakly attribute-hiding indistinguishability under-chosen message attack (wAH-sIND-CPA, for short) and can be formalized by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Game wAH-sIND-CPA$_{\mathcal{A}}^{\mathsf{HIPE}}(1^\lambda)$, where λ is the security parameter, is defined as follows.

Init: $\mathcal{A}$ is given hierarchical format $\boldsymbol{\mu}$ of depth d and outputs challenge vectors $\boldsymbol{w}_0, \boldsymbol{w}_1 \in \Sigma_{|h}$.

Setup: $\mathcal{C}$ generates master public key and secret key by invoking the setup algorithm on input the security parameter λ given in unary. Specifically, $(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda)$. Then, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input mpk.

Query Phase 1: $\mathcal{A}$ is given oracle access to $\mathsf{Derive}(mpk, msk, \cdot)$. Then, $\mathcal{A}$ can delegate secret keys directly by invoking the Derive algorithm.

Challenge: $\mathcal{A}$ gives a pair of message $(\mathfrak{m}_0, \mathfrak{m}_1)$ to $\mathcal{C}$. Then $\mathcal{C}$ chooses random $\eta \leftarrow \{0, 1\}$, encrypts $\mathfrak{m}_\eta$ under $\boldsymbol{w}_\eta$ and sends the resulting ciphertext to $\mathcal{A}$.

Query Phase 2: It is the same as Query Phase 1.

Guess: Eventually, $\mathcal{A}$ submits its guess η'.

Winning Condition: $\mathcal{A}$ wins the game if the following conditions are satisfied:

1. $\eta = \eta'$.
2. for any vector $\boldsymbol{v}$ for which the relative secret key has been obtained through derivation from one of the revealed secret key, it holds that $f_{\boldsymbol{v}}(\boldsymbol{w}_0) = f_{\boldsymbol{v}}(\boldsymbol{x}_1) = 0$.

The *advantage* of $\mathcal{A}$ in the above game is defined as

$$\mathbf{Adv}_{\text{wAH-sIND-CPA}}^{\mathsf{HIPE},\mathcal{A}}(\lambda) = \text{Prob}[\text{wAH-sIND-CPA}_{\mathcal{A}}^{\mathsf{HIPE}}(1^\lambda) = 1] - 1/2 \ .$$

Definition 5.1.2 An hierarchical inner-product encryption scheme HIPE is wAH-sIND-CPA-secure if any polynomial-time adversaries making at most a polynomial number of queries to the key derivation oracle only has a negligible advantage in the wAH-sIND-CPA game described above, meaning that $\mathbf{Adv}_{\text{wAH-sIND-CPA}}^{\mathsf{HIPE},\mathcal{A}}(\lambda)$ is negligible function of λ.

5.2 An HIPE Construction

We start by giving some intuitions on how we have extended the scheme by Agrawal *et al.* [AFV11] to the hierarchical setting.

Public Parameters. For hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$, the public parameters will contain random matrices $(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\})$ in $\mathbb{Z}_q^{n\times m}$. The master secret key is a trapdoor $\boldsymbol{T_A}$ for $\boldsymbol{A}$. To generate a secret key for vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ at depth $t \leq d$ we use the matrix:

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A} \| \sum_{j\in[\mu_1]} \sum_{\gamma=0}^{k} v_{1,j,\gamma} \cdot \boldsymbol{A}_{1,j,\gamma} \| \ldots \| \sum_{j\in[\mu_t]} \sum_{\gamma=0}^{k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma}\right) \in \mathbb{Z}_q^{n\times(t+1)m} \tag{5.1}$$

where each $v_{i,j}$ is r-decomposed for a certain fixed r and $k = \lfloor \log_r q \rfloor$.

Secret Keys. Then the secret key for $\boldsymbol{v}$ is a short basis for the lattice $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$. By using the short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$, it is possible to generate a random short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}\|\boldsymbol{v}_{t+1}})$. This provides the delegation mechanism.

Simulation. For challenge vector $\boldsymbol{w}^\star = (\boldsymbol{w}_1^\star, \ldots, \boldsymbol{w}_{t^\star}^\star)$, the simulator chooses the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ uniformly at random in $\mathbb{Z}_q^{n\times m}$ and construct the matrices $\boldsymbol{A}_{i,j,\gamma}$ as follows:

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}_{i,j,\gamma} - r^\gamma w_{i,j}^\star \boldsymbol{B}$$

where $\boldsymbol{R}_{i,j,\gamma} \in \{-1,1\}^{m\times m}$. Since the matrices $\boldsymbol{A}, \{\boldsymbol{R}_{i,j,\gamma}\}$ are uniform and independent in $\mathbb{Z}_q^{n\times m}$, we have that the $\boldsymbol{A}_{i,j,\gamma}$'s are uniform in $\mathbb{Z}_q^{n\times m}$ as in the real system. Moreover the simulator has a trapdoor $\boldsymbol{T_B}$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ but no trapdoor for $\Lambda_q^{\perp}(\boldsymbol{A})$. To generate a secret key for vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, the simulator must produce a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ where

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A} \| \boldsymbol{A}\overline{\boldsymbol{R}_1} - \langle \boldsymbol{v}_1, \boldsymbol{w}_1^\star\rangle \boldsymbol{B} \| \ldots \| \boldsymbol{A}\overline{\boldsymbol{R}_t} - \langle \boldsymbol{v}_t, \boldsymbol{w}_t^\star\rangle \boldsymbol{B}\right)$$

Then let

$$\begin{aligned}
\overline{\boldsymbol{R}_i} &= \sum_{j\in[\mu_i]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{R}_{i,j,\gamma} \quad && \in \mathbb{Z}_q^{m\times m} \\
\overline{\boldsymbol{R}} &= \left[\overline{\boldsymbol{R}_1} \| \ldots \| \overline{\boldsymbol{R}_t}\right] && \in \mathbb{Z}_q^{m\times t\cdot m} \\
\boldsymbol{B}_{\boldsymbol{v}} &= \left[-\langle \boldsymbol{v}_1, \boldsymbol{w}_1^\star\rangle \boldsymbol{B} \| \ldots \| - \langle \boldsymbol{v}_t, \boldsymbol{w}_t^\star\rangle \boldsymbol{B}\right] && \in \mathbb{Z}_q^{n\times t\cdot m}
\end{aligned}$$

Thus $F_{\boldsymbol{v}}$ can be written as:

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A} \| \boldsymbol{A}\overline{\boldsymbol{R}} + \boldsymbol{B}_{\boldsymbol{v}}\right) \quad \in \mathbb{Z}_q^{n\times(t+1)m} . \tag{5.2}$$

When $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}$ meaning that there exists an index i such that $\langle \boldsymbol{v}_i, \boldsymbol{w}_i^\star\rangle \neq 0$, the simulator can then extend $\boldsymbol{T_B}$ to a short basis for the entire lattice $\Lambda_q^{\perp}(\boldsymbol{B}_{\boldsymbol{v}})$. The simulator can now generate short vectors in $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ using algorithm SampleRight, which is sufficient for constructing a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$, as required. When $\boldsymbol{v}$ is a prefix of $\boldsymbol{w}$, meaning that for each $i = 1, \ldots, t$, $\langle \boldsymbol{v}_i, \boldsymbol{w}_i^\star\rangle = 0$, then the matrix $F_{\boldsymbol{v}}$ no longer depends on $\boldsymbol{B}$ and the simulator's trapdoor disappears. Consequently, the simulator can generate secret keys for all vectors other than prefixes of $\boldsymbol{w}^\star$. As we will see, for $\boldsymbol{w}^\star$ the simulator can produce a challenge ciphertext that helps it solve the given LWE challenge.

5.2.1 Sampling a random basis

To realize the above delegation mechanism and for the simulation we will need the following algorithms. Following, almost verbatim, [ABB10, CHKP10], let Λ be an m-dimensional lattice and let $\mathcal{O}(\Lambda, \sigma)$ be an algorithm that generates independent samples from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma}$. The following algorithm called $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ uses $\mathcal{O}$ to generate a basis T of Λ in the following way:

1. For $i = 1, \ldots, m$, generate $v \leftarrow \mathcal{O}(\Lambda, \sigma)$, if v is independent of $\{v_1, \ldots, v_{i-1}\}$, set $v_i \leftarrow v$, if not, repeat.
2. Convert the set of independent vectors $v_1, \ldots, v_m$ to a basis T using Lemma 3.3.8 (and using some canonical basis of Λ) and output T.

The following theorem summarizes properties of this algorithm.

Lemma 5.2.1 For $\sigma > \tilde{bl}(\Lambda)\omega(\sqrt{\log m})$ algorithm $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ satisfies the following properties:

1. Step 1 requires at most $O(m \log m)$ w.h.p and $2m$ samples in expectation.
2. With overwhelming probability $\|\widetilde{T}\| \leq \|T\| \leq \sigma\sqrt{m}$.
3. Up to a statistical distance, the distribution of T does not depend on the implementation of $\mathcal{O}$. That is, the random variable $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ is statistically close to $\mathsf{SampleBasis}^{\mathcal{O}'}(\Lambda, \sigma)$ for any algorithm $\mathcal{O}'$ that samples from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma}$.

By using the above SampleBasis algorithm, we define the following two algorithms. The SampleBasisLeft algorithm will be used in the real scheme, instead the SampleBasisRight will be used in our proof of security. Specifically:

Algorithm SampleBasisLeft**.** Given the lattice $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ where $F_{\boldsymbol{v}}$ is defined as in Equation 5.1 for $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, write $F_{\boldsymbol{v}} = (\boldsymbol{A}|\boldsymbol{M})$ for some matrices $\boldsymbol{A}$ and $\boldsymbol{M}$.

Then given a short basis $\boldsymbol{T_A}$ for $\Lambda_q^{\perp}(\boldsymbol{A})$ we can implement algorithm $\mathcal{O}(F_{\boldsymbol{v}}, \sigma)$ as

$$\mathcal{O}(F_{\boldsymbol{v}}, \sigma) := \mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{T_A}, 0, \sigma) \; .$$

When $\sigma > \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega(\sqrt{\log((t+1)m)})$, Theorem 3.3.10 shows that the resulting vector is distributed statistically close to $\mathcal{D}_{\Lambda_q^{\perp}(F_{\boldsymbol{v}}),\sigma}$ as required for SampleBasis. Using the above algorithm in algorithm SampleBasis leads to an algorithm to sample a random basis of $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ given a short basis of $\boldsymbol{A}$. We refer to this algorithm as SampleBasisLeft and summarize its properties in the following corollary.

Corollary 5.2.2 Algorithm $\mathsf{SampleBasisLeft}(\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{T_A}, \sigma)$ outputs a basis of $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ satisfying the three properties in Lemma 5.2.1 provided that $\boldsymbol{A}$ is rank n and $\sigma > \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega\left(\sqrt{\log((t+1)m)}\right)$.

Algorithm SampleBasisRight**.** In the simulation, the matrix $F_{\boldsymbol{v}}$ is defined as in Equation 5.2. In this case, given a short basis $\boldsymbol{T_B}$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ we can implement algorithm $\mathcal{O}(F_{\boldsymbol{v}}, \sigma)$ as follows:

1. Using Theorem 3.3.9, extend basis $\boldsymbol{T_B}$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ to a basis $\boldsymbol{T_{B_v}}$ for $\Lambda_q^{\perp}(\boldsymbol{B_v})$ such $\|\widetilde{\boldsymbol{T_{B_v}}}\| = \|\widetilde{\boldsymbol{T_B}}\|$.
2. Then run $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B_v}, \overline{\boldsymbol{R}}, \boldsymbol{T_{B_v}}, 0, \sigma)$ and output the result. When $\boldsymbol{B_v}$ is rank n and $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}^{\star}$ the matrix $\boldsymbol{B_v}$ is rank n as required for SampleRight.

Let $s_{\overline{\boldsymbol{R}}} := \|\overline{\boldsymbol{R}}\|$ be the norm of the matrix $\overline{\boldsymbol{R}}$. Then, when $\sigma > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)})$, Theorem 3.3.11 shows that the resulting vector is distributed statistically close to $\mathcal{D}_{\Lambda_q^{\perp}(F_{\boldsymbol{v}}),\sigma}$ as required for SampleBasis. Using the above algorithm in algorithm SampleBasis leads to an algorithm to sample a random basis of $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ for $F_{\boldsymbol{v}}$ defined in (5.2) given a short basis of $\boldsymbol{B}$. We refer to this algorithm as SampleBasisRight and summarize its properties in the following corollary.

Corollary 5.2.3 Algorithm SampleBasisRight$(\boldsymbol{A}, \boldsymbol{B_v}, \overline{\boldsymbol{R}}, \boldsymbol{T_B}, \sigma)$ outputs a basis of $\mathcal{D}_{\Lambda_q^{\perp}(F_{\boldsymbol{v}}),\sigma}$ satisfying the three properties in Lemma 5.2.1 provided that $\boldsymbol{B}$ is rank n, that $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}^{\star}$ and $\sigma > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)})$.

5.2.2 Our Construction

Let $n > 0$ be the security parameter and $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ the hierarchical format. Let $q = q(n, \boldsymbol{\mu})$ and $m = m(n, \boldsymbol{\mu})$ be positive integers. Let $r = r(n, \boldsymbol{\mu}) \geq 2$ be and integer and define $k = k(n, \boldsymbol{\mu}) := \lfloor \log_r q \rfloor$. Our *hierarchical inner-product encryption* for hierarchical format $\boldsymbol{\mu}$ consists of the following algorithms.

Setup$(1^n, \boldsymbol{\mu})$: On input a security parameter n, and an hierarchical format of depth d $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$, the algorithm generates public and secret parameters as follows: Use algorithm TrapGen(q, n, m) to select a uniformly random $n \times m$-matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ with a basis $\boldsymbol{T_A} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\widetilde{\boldsymbol{T_A}}\| \leq O(\sqrt{n \log q})$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, choose uniformly random matrices $\boldsymbol{A}_{i,j,\gamma} \in \mathbb{Z}_q^{n \times m}$. Select a uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Output

$$mpk = (\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \boldsymbol{u}) ,$$

and

$$msk = \boldsymbol{T_A} .$$

Derive$(mpk, sk_{\boldsymbol{v}}, \boldsymbol{v}_t)$: On input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1})$, and the vector $\boldsymbol{v}_t$, the algorithm generates a secret key for the vector $\boldsymbol{v'} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ as follows: Construct short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v'}})$ by invoking

$$\boldsymbol{S} \leftarrow \text{SampleBasisLeft}\left(F_{\boldsymbol{v}}, \sum_{j \in [\mu_i]} \sum_{\gamma \in k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma}, sk_{\boldsymbol{v}}, \sigma_t\right) ,$$

where

$$F_{\boldsymbol{v'}} = \left[F_{\boldsymbol{v}} \| \sum_{j \in [\mu_i]} \sum_{\gamma=0}^{k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma} \right] ,$$

and $sk_{\boldsymbol{v}}$ is a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$. Then, output $sk_{\boldsymbol{v'}} = \boldsymbol{S}$.

By Corollary 5.2.2, when $\sigma_t > \|\widetilde{sk_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{\log((t+1)m)})$ then $\|\widetilde{sk_{\boldsymbol{v'}}}\| \leq \|sk_{\boldsymbol{v'}}\| \leq \sigma_t \cdot \sqrt{(t+1)m}$.

Notice that, for the special case of the first level secret keys when $\boldsymbol{v}$ is the empty vector ϵ, we define $F_\epsilon := \boldsymbol{A}$ and $sk_{\boldsymbol{v}} = msk$.

$\mathsf{Enc}(mpk, \boldsymbol{w}, \mathfrak{m})$: On input the master public key mpk, the vector $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$, and the message $\mathfrak{m} \in \{0,1\}$, the algorithm generates a ciphertext Ct as follows: The algorithm chooses a uniformly random matrix $\boldsymbol{B} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, a noise vector $\boldsymbol{x} \leftarrow \overline{\Psi}_{\alpha_t}^m$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$. Then, the algorithm computes

$$\boldsymbol{c}_0 = \boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m \ .$$

For $i \in [t]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, the algorithm chooses a random matrix $\boldsymbol{R}_{i,j,\gamma} \in \{-1,1\}^{m\times m}$ and computes

$$\boldsymbol{c}_{i,j,\gamma} = (\boldsymbol{A}_{i,j,\gamma} + r^\gamma w_{i,j}\boldsymbol{B})^\top \boldsymbol{s} + \boldsymbol{R}_{i,j,\gamma}^\top \boldsymbol{x} \in \mathbb{Z}_q^m \ .$$

Finally, the algorithm computes

$$c_1 = \boldsymbol{u}^\top \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q \ ,$$

and outputs $\mathsf{Ct} = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c_1)$.

$\mathsf{Dec}(mpk, sk_{\boldsymbol{v}}, C)$: On input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, and a ciphertext $\mathsf{Ct} = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c_1)$, the algorithm does the following: For $i \in [t]$, define the r-ary expansion of the vector $\boldsymbol{v}_i$, then the algoirthm computes

$$\boldsymbol{c}_{\boldsymbol{v}_i} = \sum_{j\in[\mu_i]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{c}_{i,j,\gamma} \ .$$

Let $\boldsymbol{c} = [\boldsymbol{c}_0 \| \boldsymbol{c}_{\boldsymbol{v}_1} \| \ldots \| \boldsymbol{c}_{\boldsymbol{v}_t}]$, then the algorithm sets $\tau_t := \sigma_t \cdot \sqrt{(t+1)m} \cdot \omega(\sqrt{(t+1)m})$ and $\tau_t \geq \|\widetilde{sk_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{(t+1)m})$, and computes

$$\boldsymbol{e}_{\boldsymbol{v}} = \mathsf{SamplePre}(F_{\boldsymbol{v}}, sk_{\boldsymbol{v}}, \boldsymbol{u}, \tau_t) \ .$$

Finally, the algorithm computes

$$z = c_1 - \boldsymbol{e}_{\boldsymbol{v}}^\top \cdot \boldsymbol{c} \ ,$$

and interpreters z as in integer in $(-q/2, q/2]$. Then the algorithm outputs 0 if $|z| < q/4$ and 1 otherwise.

With the following lemma, we show the correctness of our scheme.

Lemma 5.2.4 For hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ of depth d, suppose the parameters q and α_t, for each $t \in [d]$, are such that

$$q/\log q = \Omega\left(\sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m^{3/2}\right) \text{ and } \alpha_t \leq \left(t \cdot \log q \cdot \sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m \cdot \omega(\sqrt{\log m})\right)^{-1} ,$$

where $\mu = \max_{i\in[d]} \mu_i$. Moreover, for vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ and $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h)$, let $sk_{\boldsymbol{v}}$ be the basis of the lattice $F_{\boldsymbol{v}}$ obtained through a sequence of calls to the Derive algorithm, let $\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk, \boldsymbol{w}, \mathfrak{m})$ and $\mathfrak{m}' \leftarrow \mathsf{Dec}(mpk, sk_{\boldsymbol{v}}, C)$.

Then, if $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$, namely $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0 \pmod q$ for each $i \in [t]$, then with overwhelming probability we have $\mathfrak{m}' = \mathfrak{m}$.

Proof: Just for notational convenience, let

$$\widetilde{\boldsymbol{A}_i} = \left(\sum_{j\in[\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{A}_{i,j,\gamma} \right) \quad \text{and} \quad \widetilde{\boldsymbol{R}_i} = \sum_{j\in[\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma}^{\top} \boldsymbol{x} \ .$$

Then, for each $i \in [t]$, the decryption algorithm computes:

$$\begin{aligned} \boldsymbol{c}_{\boldsymbol{v}_i} &= \sum_{j\in[\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{c}_{i,j,\gamma} \\ &= \sum_{j\in[\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \left[(\boldsymbol{A}_{i,j,\gamma} + r^{\gamma} w_{i,j} \boldsymbol{B})^{\top} \boldsymbol{s} + \boldsymbol{R}_{i,j,\gamma}^{\top} \boldsymbol{x} \right] \\ &= \widetilde{\boldsymbol{A}_i}^{\top} \boldsymbol{s} + \underbrace{\left(\sum_{j\in[\mu]} \sum_{\gamma=0}^{k} r^{\gamma} v_{i,j,\gamma} w_{i,j} \right)}_{\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle} \boldsymbol{B}^{\top} \boldsymbol{s} + \widetilde{\boldsymbol{R}_i} \ . \end{aligned}$$

If $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0$ then we have:

$$\boldsymbol{c}_{\boldsymbol{v}_i} = \widetilde{\boldsymbol{A}_i}^{\top} \boldsymbol{s} + \widetilde{\boldsymbol{R}_i} \pmod{q} \ .$$

Thus, if for each $i \in [t]$, $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0$ then $\boldsymbol{c}$ can be written as:

$$\begin{aligned} \boldsymbol{c} =& [\boldsymbol{c}_0 \| \boldsymbol{c}_{\boldsymbol{v}_1} \| \dots \| \boldsymbol{c}_{\boldsymbol{v}_t}] \\ =& \left[\boldsymbol{A} \| \widetilde{\boldsymbol{A}_1} \| \dots \| \widetilde{\boldsymbol{A}_t} \right]^{\top} \boldsymbol{s} \ + \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod{q} \\ =& F_{\boldsymbol{v}}^{\top} \cdot \boldsymbol{s} + \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod{q} \ . \end{aligned}$$

At this point, the short vector $\boldsymbol{e}_{\boldsymbol{v}} = \mathsf{SampleLeft}(F_{\boldsymbol{v}}, sk_{\boldsymbol{v}}, \boldsymbol{u}, \tau_t)$ is computed by the decryption algorithm such that by Theorem 3.3.10, $F_{\boldsymbol{v}} \cdot \boldsymbol{e}_{\boldsymbol{v}} = \boldsymbol{u} \pmod{q}$. It follows that

$$\boldsymbol{e}_{\boldsymbol{v}}^{\top} \boldsymbol{c} = \boldsymbol{u}^{\top} \boldsymbol{s} + \boldsymbol{e}_{\boldsymbol{v}}^{\top} \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod{q} \ .$$

Finally, the decryption algorithm computes:

$$\begin{aligned} z &= c_1 - \boldsymbol{e}_{\boldsymbol{v}}^{\top} \boldsymbol{c} \pmod{q} \\ &= \left(\boldsymbol{u}^{\top} \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil \right) - \boldsymbol{u}^{\top} \boldsymbol{s} - \boldsymbol{e}_{\boldsymbol{v}}^{\top} \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod{q} \\ &= \mathfrak{m} \cdot \lfloor q/2 \rceil + \underbrace{\left(x - \boldsymbol{e}_{\boldsymbol{v}}^{\top} \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \right)}_{\text{noise term}} \pmod{q} \ . \end{aligned}$$

Thus, to have a successful decryption, it suffices to set the parameters so that with overwhelming probability,

$$\left| x - \boldsymbol{e}_{\boldsymbol{v}}^{\top} \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \dots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \right| < \frac{q}{4} \ .$$

Let us write $\boldsymbol{e_v} = [\boldsymbol{e}_{\boldsymbol{v},0} \| \boldsymbol{e}_{\boldsymbol{v},1} \| \ldots \| \boldsymbol{e}_{\boldsymbol{v},t}]$ with $\boldsymbol{e}_{\boldsymbol{v},i} \in \mathbb{Z}^m$ for $i = 0, \ldots, t$. Then the noise term can be rewritten as

$$x - \left(\boldsymbol{e}_{\boldsymbol{v},0} + \sum_{i \in [t]} \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma} \boldsymbol{e}_{\boldsymbol{v},i} \right)^{\top} \boldsymbol{x} \ .$$

By Lemma 3.3.1, we have $\|\boldsymbol{e_v}\| < \tau_t \sqrt{(t+1)m}$ with overwhelming probability. Moreover by Lemma 3.3.2, we have $\|\boldsymbol{R}_{i,j,\gamma} \cdot \boldsymbol{e}_{\boldsymbol{v},i}\| \le 12\sqrt{2m} \cdot \|\boldsymbol{e}_{\boldsymbol{v},i}\|$ with overwhelming probability, and since $v_{i,j,\gamma} \in [0, r-1]$ it follows that

$$\left\| \boldsymbol{e}_{\boldsymbol{v},0} + \sum_{i \in [t]} \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma} \boldsymbol{e}_{\boldsymbol{v},i} \right\| = O(t \cdot \mu \cdot k \cdot r \cdot \sigma_t \cdot m) \ .$$

Finally, by Lemma 3.3.16, the error term has absolute value at most:

$$\left(q\alpha_t \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2 \right) \cdot O\left(t \cdot \mu \cdot k \cdot r \cdot \sigma_t \cdot m\right) \ .$$

∎

5.3 Proofs of Security

In this section we prove the following theorem.

Theorem 5.3.1 If the decision-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible (Definition 3.3.12), then our HIPE scheme is wAH-sIND-CPA-secure.

Following [ABB10, AFV11], we define additional algorithms. These will not be used in the real scheme, but we need them in our proofs.

$\mathsf{Sim.Setup}(1^n, \boldsymbol{\mu}, \boldsymbol{w}^\star)$: The main difference with the normal setup algorithm is that now the simulator does not generate any more a trapdoor for the matrix $\boldsymbol{A}$. Instead, a trapdoor for a matrix $\boldsymbol{B}$, that will be embedded into the $\boldsymbol{A}_{i,j,\gamma}$ matrices, is generated. Furthermore, the simulator embeds the challenge vector $\boldsymbol{w}^\star$ in the public parameters in such a way it is still possible to generate a ciphertext for $\boldsymbol{w}^\star$ but no secret key can be generate for vectors $\boldsymbol{v}$ such that $f_{\boldsymbol{v}}(\boldsymbol{w}^\star) = 1$.

Specifically, the algorithm chooses random $\boldsymbol{A}$, $\boldsymbol{R}^\star_{i,j,k}$ and $\boldsymbol{u}$, it uses $\mathsf{TrapGen}$ to generate $\boldsymbol{B}^\star$ and defines

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}^\star_{i,j\gamma} - r^\gamma w^\star_{i,j} \boldsymbol{B}^\star \ .$$

Specifically, on input a security parameter n, an hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ of depth d , and a challenge vector $\boldsymbol{w}^\star = (\boldsymbol{w}^\star_1, \ldots, \boldsymbol{w}^\star_d)$, the algorithm generates public and secret parameters as follows: Choose random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, choose uniformly random matrices $\boldsymbol{R}^\star_{i,j,\gamma} \in \mathbb{Z}_q^{n \times m}$. Select a uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Use algorithm $\mathsf{TrapGen}(q, n, m)$ to select a uniformly random $n \times m$-matrix $\boldsymbol{B}^\star \in \mathbb{Z}_q^{n \times m}$ with a basis $\boldsymbol{T}_{\boldsymbol{B}^\star} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^\perp(\boldsymbol{B}^\star)$ such that $\|\widetilde{\boldsymbol{T}_{\boldsymbol{B}^\star}}\| \le O(\sqrt{n \log q})$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, set

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j} \boldsymbol{B}^\star \ .$$

Output $mpk = (\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \boldsymbol{u})$ and $msk = (\{\boldsymbol{R}^\star_{i,j,\gamma}\}, \boldsymbol{B}^\star, \boldsymbol{T}_{\boldsymbol{B}^\star})$.

$\mathsf{Sim.Derive}(mpk, sk_{\boldsymbol{v}}, \boldsymbol{v}_t)$: Secret keys are now created by using the trapdoor $\boldsymbol{T}_{\boldsymbol{B}^\star}$, sampled by Sim.Setup, and the SampleBasisRight algorithm.

Specifically, on input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_{t-1})$, and the vector $\boldsymbol{v}_t$, the algorithm generates a secret key for the vector $\boldsymbol{v'} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_t)$ by constructing a short basis for $\Lambda_q^\perp(F_{\boldsymbol{v}'})$, as defined by Equation 5.2, by invoking

$$\boldsymbol{S} \leftarrow \mathsf{SampleBasisRight}(\boldsymbol{A}, \boldsymbol{B}^\star_{\boldsymbol{v}'}, \overline{\boldsymbol{R}^\star}, \boldsymbol{T}_{\boldsymbol{B}^\star}, \sigma_t) \; .$$

Output $sk_{\boldsymbol{v}'} = \boldsymbol{S}$.

Notice that if the simulator tries to generate a secret key for a vector $\boldsymbol{v}$ such that $f_{\boldsymbol{v}}(\boldsymbol{w}^\star) = 1$ then the simulator's trapdoor disappears meaning that the simulator cannot generate such a secret key.

$\mathsf{Sim.Enc}(mpk, \boldsymbol{w}, \mathfrak{m})$: The algorithm differs from Enc in the sense that it uses matrices $\boldsymbol{R}^\star_{i,j,\gamma}$ and $\boldsymbol{B}^\star$ instead of matrices $\boldsymbol{R}_{i,j,\gamma}$ and $\boldsymbol{B}$.

Specifically, on input master public key mpk, vector $\boldsymbol{w} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_t)$, and message $\mathfrak{m} \in \{0,1\}$, the algorithm generates a ciphertext Ct as follows: The algorithm chooses a uniformly random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, a noise vector $\boldsymbol{x} \leftarrow \overline{\Psi}^m_{\alpha_t}$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$. Then, the algorithm computes

$$\boldsymbol{c}_0 = \boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m \; ,$$

and, for $i \in [t]$, $j \in [\mu_i]$ and $\gamma = 0, \dots, k$,

$$\boldsymbol{c}_{i,j,\gamma} = (\boldsymbol{A}_{i,j,\gamma} + r^\gamma w_{i,j} \boldsymbol{B}^\star)^\top \boldsymbol{s} + \boldsymbol{R}^{\star\top}_{i,j,\gamma} \boldsymbol{x} \in \mathbb{Z}_q^m \; .$$

Finally, the algorithm computes

$$c_1 = \boldsymbol{u}^\top \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q \; ,$$

and outputs $\mathsf{Ct} = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c_1)$.

We now turn on the proof of security that follows the same line of that of Agrawal *et al.* [AFV11] adapted to new hierarchical setting.

Specifically, for a probabilistic polynomial-time adversary $\mathcal{A}$, our proof of security will consist of the following sequence of 6 games between $\mathcal{A}$ and $\mathcal{C}$ showing that the encryption of the pair $(\boldsymbol{w}_0, \mathfrak{m}_0)$ is computationally indistinguishable from the encryption of the pair $(\boldsymbol{w}_1, \mathfrak{m}_1)$.

The six games are defined as follows:

$\mathbf{Game_0}$: $\mathcal{C}$ runs the Setup algorithm, answers $\mathcal{A}$'s secret key queries using the Derive algorithm, and generates the challenge ciphertext for vector $\boldsymbol{w}_0$ and message $\mathfrak{m}_0$ by using the Enc algorithm.

$\mathbf{Game_1}$: In this game $\mathcal{C}$ uses the simulation algorithms. Specifically, $\mathcal{C}$ runs the Sim.Setup algorithm with $\boldsymbol{w}^\star = \boldsymbol{w}_0$, answers $\mathcal{A}$'s secret key queries using the Sim.Derive algorithm, and generates the challenge ciphertext for vector $\boldsymbol{w}_0$ and message $\mathfrak{m}_0$ by using the Sim.Enc algorithm .

$\mathbf{Game_2}$: It is the same as the $\mathbf{Game_1}$ except that the challenge ciphertext is randomly chosen from the ciphertext space.

$\mathbf{Game_3}$: It is the same as the $\mathbf{Game_2}$ except that $\mathcal{C}$ runs the Sim.Setup algorithm with $\boldsymbol{w}^\star = \boldsymbol{w}_1$.

$\mathbf{Game_4}$: It is the same as the $\mathbf{Game_3}$ except that $\mathcal{C}$ generates the challenge ciphertext for vector $\boldsymbol{w}_1$ and message $\mathfrak{m}_1$ by using the Sim.Enc algorithm

$\mathbf{Game_5}$: $\mathcal{C}$ runs the Setup algorithm, answers $\mathcal{A}$'s secret key queries using the Derive algorithm, and generates the challenge ciphertext for vector $\boldsymbol{w}_1$ and message $\mathfrak{m}_1$ by using the Enc algorithm.

We give now formal proofs of the above statements.

Lemma 5.3.2 The view of the adversary $\mathcal{A}$ in $\mathbf{Game_0}$ (resp. $\mathbf{Game_4}$) is statistically close to the view of $\mathcal{A}$ in $\mathbf{Game_1}$ (resp. $\mathbf{Game_5}$).

Proof: We will show that $\mathbf{Game_0}$ is statically close to $\mathbf{Game_1}$. The same proof also applies to $\mathbf{Game_4}$ and $\mathbf{Game_5}$.

Setup: In $\mathbf{Game_0}$, matrix $\boldsymbol{A}$ is generated by TrapGen and, for each $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, matrix $\boldsymbol{A}_{i,j,\gamma}$ is uniformly random in $\mathbb{Z}_q^{n \times m}$.

Instead, in $\mathbf{Game_1}$ $\boldsymbol{A}$ is chosen uniformly at random and, for each $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, we have $\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star$, where $\boldsymbol{B}^\star$ is generated by TrapGen and the matrices $\boldsymbol{R}^\star_{i,j,\gamma}$ are uniformly and independently chosen at random in $\{-1,1\}^{m \times m}$.

Moreover, in both the games the vector $\boldsymbol{u}$ is chosen at random in $\mathbb{Z}_q^n$. Notice that, by Theorem 3.3.8, since $m \geq 6n \log q$, the matrixes $\boldsymbol{A}$ and $\boldsymbol{B}^\star$ output by TrapGen are statistically indistinguishable from a uniformly random matrix.

Secret keys: Assuming, for each level $t \in [d]$, a sufficiently large σ_t, then in $\mathbf{Game_0}$, the secret key for vector $\boldsymbol{v}$, $sk_{\boldsymbol{v}}$, is a basis of $\Lambda_q^\perp(F_{\boldsymbol{v}})$, sampled by using the SampleBasisLeft algorithm, satisfying the properties in Lemma 5.2.1. The same happens in $\mathbf{Game_1}$ by using SampleBasisRight algorithm this time. Moreover, Lemma 5.2.1, guarantees the independence from the derivation path. Thus, the secret keys have the same distribution in both games.

Challenge: In $\mathbf{Game_0}$, the challenge ciphertext components $\boldsymbol{c}_{i,j,\gamma}$ are computed as follow:

$$\boldsymbol{c}_{i,j,\gamma} = \left(\boldsymbol{A}_{i,j,\gamma} + r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\right)^\top \boldsymbol{s} + {\boldsymbol{R}^\star_{i,j,\gamma}}^\top \boldsymbol{x} \in \mathbb{Z}_q^m ,$$

where $\boldsymbol{B}^\star$ is uniformly random in $\mathbb{Z}_q^{n \times m}$ and the matrices $\boldsymbol{R}^\star_{i,j,\gamma}$ are uniformly and independently chosen at random in $\{-1,1\}^{m \times m}$.

On the other side, in $\mathbf{Game_1}$, we have:

$$\begin{aligned}\boldsymbol{c}_{i,j,\gamma} &= \left(\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star + r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\right)^\top \boldsymbol{s} + {\boldsymbol{R}^\star_{i,j,\gamma}}^\top \boldsymbol{x} \\ &= {\boldsymbol{R}^\star_{i,j,\gamma}}^\top \left(\boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x}\right)\end{aligned} ,$$

where, this time, the matrices $\boldsymbol{R}^\star_{i,j,\gamma}$ are the same used to construct the public parameters.

Let us now analyze the join distribution of the public parameters and the challenge ciphertext in $\mathbf{Game_0}$ and $\mathbf{Game_1}$. We will show that the distributions of $(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \{\boldsymbol{c}_{i,j,\gamma}\})$ in $\mathbf{Game_0}$ and in $\mathbf{Game_1}$ are statistical indistinguishable.

First notice that by [ABB10, Lemma 13] we have that the following two distributions are statistically indistinguishable for every fixed matrix $\boldsymbol{B}^\star$, every $\boldsymbol{w}^\star$ and every vector $\boldsymbol{x} \in \mathbb{Z}_q^m$:

$$\left(\boldsymbol{A}, \boldsymbol{A}_{i,j,\gamma}, {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right) \approx_s \left(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star, {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right) .$$

Since each $\boldsymbol{R}^\star_{i,j,\gamma}$ is chosen independently for every i, j and γ, then the joint distribution of them are statistically close:

$$\left(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \left\{{\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right) \approx_s \left(\boldsymbol{A}, \{\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\}, \left\{{\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right)$$

Since $(\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star)^\top \boldsymbol{s}$ is statistically close to $\boldsymbol{A}_{i,j,\gamma}^\top \boldsymbol{s}$, it is possible to add each term to one side of the equation:

$$\begin{gathered}\left(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \left\{\boldsymbol{A}_{i,j,\gamma}^\top \boldsymbol{s} + {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right) \approx_s \\ \left(\boldsymbol{A}, \{\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\}, \left\{\left(\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\right)^\top \boldsymbol{s} + {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right)\end{gathered} .$$

Finally, we add $\left(r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\right)^\top \boldsymbol{s}$ to both sides:

$$\begin{gathered}\left(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \left\{\left(\boldsymbol{A}_{i,j,\gamma} + r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\right)^\top \boldsymbol{s} + {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right) \approx_s \\ \left(\boldsymbol{A}, \{\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star\}, \left\{\left(\boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma}\right)^\top \boldsymbol{s} + {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x}\right\}\right)\end{gathered} .$$

To conclude, observe that the distribution on the left hand side is that of the public parameters and the challenge ciphertext in $\mathbf{Game_0}$, while that on the right hand side is the distribution in $\mathbf{Game_1}$. ∎

Lemma 5.3.3 The view of the adversary $\mathcal{A}$ in $\mathbf{Game_1}$ (resp. $\mathbf{Game_3}$) is computationally indistinguishable to the view of $\mathcal{A}$ in $\mathbf{Game_2}$ (resp. $\mathbf{Game_4}$) under the decision-LWE problem.

Proof: We will show that $\mathbf{Game_1}$ is computationally indistinguishable from $\mathbf{Game_2}$ by giving a reduction from the decision-LWE problem. The same proof applies also to $\mathbf{Game_3}$ and $\mathbf{Game_4}$.

Suppose $\mathcal{A}$ can distinguish between $\mathbf{Game_1}$ and $\mathbf{Game_2}$ with non-negligible advantage. Then, it is possible to use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ to solve the decision-LWE problem.

Init: $\mathcal{B}$ is given $m+1$ LWE challenge pairs $(\boldsymbol{a}_j, y_j) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where either $y_j = \langle \boldsymbol{a}_j, s\rangle + x_j$ for a random $s \in \mathbb{Z}_q^n$ and a noise term $x_j \leftarrow \Psi_\alpha$ or y_j is uniformly random in $\mathbb{Z}_q$.

Setup: The public parameters are constructed using the vectors of the pairs $(\boldsymbol{a}_j, y_j)$. The i-th column of matrix $\boldsymbol{A}$ will be the vector $\boldsymbol{a}_i, 1 \leq i \leq m$ and vector $\boldsymbol{u}$ will be $\boldsymbol{a}_0$. The matrices $\boldsymbol{A}_{i,j,\gamma}$ are still computed as in Sim.Setup on input $\boldsymbol{w}^\star = \boldsymbol{w}_0$, i.e.,

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}^\star_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star .$$

Secret keys: All private-key extraction queries are answered using Sim.Derive.

Challenge: The ciphertext $\mathsf{Ct} = (\boldsymbol{c}_0, \{c_{i,j,\gamma}\}, c_1)$ is constructed using the LWE challenge pairs $(\boldsymbol{a}_j, y_j)$, as follows:

$$\boldsymbol{c}_0 = (y_1, \dots, y_m)\ ,$$

$$\boldsymbol{c}_{i,j,\gamma} = \boldsymbol{R}^{\star\top}_{i,j,\gamma}\boldsymbol{c}_0\ ,$$

and

$$c_1 = y_0 + \mathfrak{m} \cdot \lfloor q/2 \rceil\ .$$

Then, If $y_j = \langle \boldsymbol{a}_j, s\rangle + x_j$, we have that

$$\begin{aligned}\boldsymbol{c}_0 &= (y_1, \dots, y_m) \\ &= (\langle \boldsymbol{a}_1, s\rangle + x_1, \dots, \langle \boldsymbol{a}_m, s\rangle + x_m) = \boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x}\end{aligned}\ ,$$

$$\begin{aligned}\boldsymbol{c}_{i,j,\gamma} &= \boldsymbol{R}^{\star\top}_{i,j,\gamma}\boldsymbol{c}_0 \\ &= \boldsymbol{R}^{\star\top}_{i,j,\gamma}(\boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x}) \\ &= (\boldsymbol{A}\boldsymbol{R}^{\star}_{i,j,\gamma})^\top \boldsymbol{s} + \boldsymbol{R}^{\star\top}_{i,j,\gamma}\boldsymbol{x} \\ &= (\boldsymbol{A}\boldsymbol{R}^{\star}_{i,j,\gamma} - r^\gamma w^\star_{i,j}\boldsymbol{B}^\star + r^\gamma w^\star_{i,j}\boldsymbol{B}^\star)^\top \boldsymbol{s} + \boldsymbol{R}^{\star\top}_{i,j,\gamma}\boldsymbol{x} \\ &= (\boldsymbol{A}_{i,j,\gamma} + r^\gamma w^\star_{i,j}\boldsymbol{B})^\top \boldsymbol{s} + \boldsymbol{R}^{\star\top}_{i,j,\gamma}\boldsymbol{x}\end{aligned}\ ,$$

and

$$\begin{aligned}c_1 &= y_0 + \mathfrak{m}\lfloor q/2 \rceil \\ &= \boldsymbol{u}^\top \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil\end{aligned}\ .$$

Therefore the challenge ciphertext is distributed exactly as in $\mathbf{Game_1}$.

Instead, if y_j is uniformly random in $\mathbb{Z}_q$ then the challenge ciphertext is

$$(\boldsymbol{c}_0, \overline{\boldsymbol{R}}^{\star\top}\boldsymbol{c}_0, c_1)\ ,$$

where $\overline{\boldsymbol{R}}^{\star\top}$ is the concatenation of all the $\boldsymbol{R}^{\star}_{i,j,\gamma}$ matrices, and by the leftover hash lemma [Sho08, Theorem 8.38], $\boldsymbol{A}\overline{\boldsymbol{R}}^{\star}$ and $\overline{\boldsymbol{R}}^{\star\top}\boldsymbol{c}_0$ are independent uniformly random samples.

Therefore, the ciphertext is uniformly random, as the ciphertext generated by $\mathbf{Game_2}$.

At the end $\mathcal{A}$ must guess if it is interacting with $\mathbf{Game_1}$ or $\mathbf{Game_2}$. Notice that, the answer to this guess is also the answer to the LWE challenge. We showed that if y_j is uniformly random in $\mathbb{Z}_q$, than $\mathcal{A}$'s view is the same as in $\mathbf{Game_2}$ and if $y_j = \langle \boldsymbol{a}_j, s\rangle + x_j$, than $\mathcal{A}$'s view is the same as in $\mathbf{Game_1}$. Therefore, the $\mathcal{B}$'s advantage in solving LWE is the same as $\mathcal{A}$'s advantage in distinguish the views of $\mathbf{Game_1}$ and $\mathbf{Game_2}$. ▌

Them, it remains to show that $\mathbf{Game_2}$ is indistinguishability from $\mathbf{Game_3}$.

Lemma 5.3.4 The view of the adversary $\mathcal{A}$ in $\mathbf{Game_2}$ is statistically indistinguishability to the view of $\mathcal{A}$ in $\mathbf{Game_3}$

Proof: We will show that $\mathbf{Game_2}$ is statically close to $\mathbf{Game_3}$. The main difference between the two games is in the way the Sim.Setup algorithm is invoked. Specifically, in $\mathbf{Game_2}$, Sim.Setup is invoked on input $\boldsymbol{w}^\star = \boldsymbol{w}_0$. In $\mathbf{Game_3}$, $\boldsymbol{w}^\star = \boldsymbol{w}_1$ is used instead.

Notice that, all public parameters are randomly generated in both games, except for the matrices $\boldsymbol{A}_{i,j,\gamma}$. Therefore, the indistinguishability of $\mathbf{Game_2}$ and $\mathbf{Game_3}$ only depends on the indistinguishability of the $\boldsymbol{A}_{i,j,\gamma}$ matrices that in both games they are generated as

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}^{\star}_{i,j,\gamma} - r^{\gamma}\boldsymbol{w}^{\star}_{i,j}\boldsymbol{B}^{\star} .$$

By Lemma 3.3.4, $(\boldsymbol{A}, \boldsymbol{A}\overline{\boldsymbol{R}}^{\star})$ is statistically indistinguishable from $(\boldsymbol{A}, \boldsymbol{A}')$ where $\overline{\boldsymbol{R}}^{\star}$ is the concatenation of all the $\boldsymbol{R}^{\star}_{i,j,\gamma}$ matrices. Moreover, for any fixed $\boldsymbol{X}$ and uniformly random $\boldsymbol{C}$, the variable $\boldsymbol{C} - \boldsymbol{X}$ is also uniformly random. Therefore the distributions of $\boldsymbol{A}_{i,j,\gamma}$ in both games are statistically indistinguishable. ▌

Wrapping up. We have proved by the previous lemmata that our hierarchical inner-product encryption is weak attribute hiding-selective attribute secure assuming decision-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible. To finalize we extract the parameters required for correctness and security of the system. Specifically:

- We need to ensure that for each $t \in [d]$, correctness holds. Specifically, Lemma 5.2.4 requires
$$q/\log q = \Omega(\sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m^{3/2})$$
and
$$\alpha_t \le (t \cdot \log q \cdot \sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m \cdot \omega(\sqrt{\log m}))^{-1} .$$

- By Theorem 3.3.8, algorithm $\mathsf{TrapGen}$ requires
$$q > 2 \text{ and } m > 6n \lg q .$$

- By Corollary 5.2.2, to have algorithm $\mathsf{SampleBasisLeft}$ working correctly in the Derive algorithm, we need for each $t \in [d]$,
$$\sigma_t > \|\widetilde{sk_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{\log((t+1)m)}) .$$
Thus, we have
$$\sigma_t > \sigma_{\mathrm{TG}} \cdot \omega((\log m)^{t/2}) .$$

- By Corollary 5.2.3, to have algorithm $\mathsf{SampleBasisRight}$ working correctly in the $\mathsf{Sim.Derive}$ algorithm, we need for each $t \in [d]$,
$$\sigma_t > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)}) .$$
Thus, by Theorem 3.3.8,
$$\|\widetilde{\boldsymbol{T_B}}\| < \sigma_{\mathrm{TG}}$$
and, by Lemma 3.3.2,
$$s_{\overline{\boldsymbol{R}}} = \|\overline{\boldsymbol{R}}\| = O(t \cdot \mu \cdot (\log_r q + 1) \cdot \sqrt{(t+1)m})$$
due the particular structure of $\overline{\boldsymbol{R}}$, where $\mu = \max_{i\in[d]} \mu_i$. Thus,
$$\sigma_t > O(\sqrt{n \log q}) \cdot O(\mu \cdot (\log_r q + 1) \cdot \sqrt{(t+1)m}) \cdot \omega(\sqrt{(t+1)m}) .$$

- Finally, Regev's reduction must apply. Thus
$$q > 2\sqrt{n}/\alpha_t .$$

Chapter 6

Adaptive Simulation-Based Secure Constructions for Functional Encryption

In this chapter we consider the problem of designing simulation-based secure functional encryption schemes.

Specifically, we give a general transformation that takes a game-based secure functional encryption scheme for all polynomial-size circuits and constructs a functional encryption scheme for the same functionality that is simulation-based secure against adaptive adversaries that can ask for a bounded number of tokens and can see one ciphertext. Our transformation is both black-box and for the standard model and is inspired by the work of Feige, Lapidot and Shamir [FLS90]. We remark that by the recent impossibility results of [AGVW12], the restriction to bounded number of tokens is necessary.

We then investigate functional encryption schemes for Hidden-Vector Encryption [BW07] which is readily seen to be a generalization of (Anonymous) Identity-Based Encryption. Therefore the impossibility result for IBE of [BSW11] applies also to HVE. The current state of the art [BSW11, O'N10, AGVW12] tells us that in general for functional encryption the game-based and simulation-based definitions of security are not equivalent. Despite this, the work of O'Neill [O'N10] provides us with an avenue for constructing simulation-based secure functional encryption schemes from game-based secure ones for non-adaptive adversaries (these are adversaries that cannot ask token queries after having seen the ciphertext). Specifically, for a *pre-image sampleable* functionality (see later for a definition) any game-based secure functional encryption scheme is also non-adaptive simulation-based secure. We show that if HVE is pre-image sampleable then it is possible to decide $\mathbb{NP}$ in probabilistic polynomial time. Therefore it is unlikely that pre-image samplability can be used to construct simulation-based secure HVE.

Motivated by this result, we then show how to construct a HVE scheme whose simulation-based security can be proved under standard assumptions in the bilinear pairing setting in the standard model. Our construction is shown secure against *adaptive* adversaries obtaining *one* ciphertext and asking an *unbounded* number of tokens. Again, this is the best one can hope for in the standard model, given the impossibility results of [BSW11, BO12] for IBE. The only previous simulation-based construction for IBE secure against adaptive adversary was given in [BSW11] and is in the programmable random oracle model and imposes no bound on the number of tokens and ciphertexts obtained by the adversary.

We remark that a functional encryption scheme for HVE can be derived from the general result of [GVW12]. However, our construction allows the adversary to obtain an unbounded number of tokens (whereas the construction of [GVW12] imposes an upper bound on the number of tokens that can be seen by an adversary) and is more efficient (this is due to the fact that our construction is tailored for HVE whereas the one of [GVW12] is for a general class). Finally, we remark that the impossibility result of Agrawal *et al.* [AGVW12] does not apply to HVE. Indeed, the impossibility results uses the fact that polynomial size circuits can compute *incompressible* functionalities (i.e., weak pseudo-random functions). It is easy to see that HVE is not incompressible.

6.1 A General Transformation

In this section we show how to transform any game-based secure functional encryption scheme for the $\mathsf{Circuit}$ functionality (see Definition 2.0.2) into one for the same functionality that is simulation-based secure against adaptive adversaries that can ask a *bounded* number of tokens and can see one ciphertext. The transformation has some similarities with the FLS paradigm introduced by Feige, Lapidot and Shamir [FLS90] to obtain zero-knowledge proof systems from witness indistinguishable proof systems.

Notice, that by the impossibility result of [AGVW12], in the full model, there exists no functional encryption scheme for the functionality of all circuits that is simulation-based secure against adversaries that ask an *unbounded* number of tokens (even if the adversary is non-adaptive).

Let us start by showing how to transform a given n-input Boolean circuit C in a new randomized circuit over $n\cdot(q+1)$ Boolean inputs C^r such that C^r evaluates the same function of C and provides additional slots in the message that will be used in a critical way by our simulator. Later we will set q as the bound on the number of tokens the adversary can ask for, and r as a random n-bit Boolean string.

Definition 6.1.1 [The transformed circuit C^r] Fix the security parameter λ and let $n = \mathrm{poly}(\lambda)$. For any string $r \in \{0,1\}^n$ and any n-bit input circuit C, we define $(n\cdot(q+1))$-input Boolean circuit C^r as follows:

$$C^r(m, m_1, \ldots, m_q) = \begin{cases} 1, & \text{if } m_i = r \text{ for some } i \in [q]; \\ 0, & \text{if } m = \perp \text{ and } m_i \neq r, \text{for all } i \in [q]; \\ C(m), & \text{otherwise.} \end{cases}$$

Moreover, we let $\perp$ be a special message not contained in X_n.

Now, we are ready to show how to transform a functional encryption scheme IndFE for the $\mathsf{Circuit}$ functionality in an another functional encryption scheme SimFE for the same functionality by using C^r. We call this transformation $\mathsf{Trans}_{\mathsf{Bound}}$.

Definition 6.1.2 [The transformation $\mathsf{Trans}_{\mathsf{Bound}}$] Let IndFE be a functional encryption scheme for the functionality Circuit defined by the algorithm $=$ ($\mathsf{IndFE.Setup}$, $\mathsf{IndFE.Enc}$, $\mathsf{IndFE.KeyGen}$, $\mathsf{IndFE.Eval}$), and let $q > 0$ be a fixed integer. We define a new functional encryption scheme $\mathsf{SimFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ for Circuit as follows.

- $\mathsf{Setup}(1^\lambda, 1^n)$: the algorithm returns the output of $\mathsf{IndFE.Setup}(1^\lambda, 1^{n\cdot(q+1)})$ as its own output.
- $\mathsf{Enc}(mpk, m)$: on input mpk and $m \in \{0,1\}^n$, the algorithm chooses q random n-bit Boolean strings $m_1, \ldots, m_q \in \{0,1\}^n$ and sets $m' = (m, m_1, \ldots, m_q)$ and returns the output of $\mathsf{IndFE.Enc}(mpk, m')$ as its own output.

- $\mathsf{KeyGen}(msk, C)$: on input msk and a n-input Boolean circuit C, the algorithm chooses random $r \in \{0,1\}^n$ and returns the pair (r, sk) where $sk = \mathsf{IndFE.KeyGen}(msk, C^r)$.

- $\mathsf{Eval}(mpk, \mathsf{Ct}, sk)$: on input mpk, Ct and sk, the algorithm returns the output $\mathsf{IndFE.Eval}(mpk, \mathsf{Ct}, sk)$ as its own output.

Correctness. Suppose that Eval is invoked on a ciphertext Ct for $m \in \{0,1\}^n$ and sk_C for circuit C. Then if $m_i \neq r$ for all $i \in [q]$, Eval returns the correct evaluation, namely $C(m)$. Thus, the probability that Eval fails is at most $q/2^n$ that is negligible in n.

Before proving the q-SIM-Security of SimFE, we mention that SimFE is also IND-Secure. Notice that it is not implied by q-SIM-Security. The lemma follows from a simple reduction to the IND-Security of IndFE and we state it without proof.

Lemma 6.1.3 If IndFE is IND-Secure then SimFE is IND-Secure.

We now prove the main result of this section, namely, that SimFE is q-SIM-Secure.

Theorem 6.1.4 If there exists an IND-Secure functional encryption scheme for circuits, then, for all constant q, there exists an q-SIM-Secure functional encryption scheme for the same functionality.

Proof: Let $\mathsf{IndFE} = (\mathsf{IndFE.Setup}, \mathsf{IndFE.Enc}, \mathsf{IndFE.KeyGen}, \mathsf{IndFE.Eval})$ be an IND-Secure functional encryption scheme for the functionality Circuit, and let $\mathsf{SimFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval})$ be the functional encryption scheme for the same functionality obtained applying $\mathsf{Trans}_{\mathsf{Bound}}$. Then, we show that SimFE is q-SIM-Secure by constructing a simulator $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for any p.p.t. adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ and $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$ are computationally indistinguishable, and reducing the q-SIM security to the IND security of IndFE.

Specifically, the simulator works as follows:

- Sim_0 on input the security parameters invokes the $\mathsf{IndFE.Setup}$ algorithm to generate public parameters and master secret key.

- Sim_1 on input the public parameter mpk, token queries $C_1, \ldots, C_{q_1}$ made by $\mathcal{A}_0$, the Boolean values $z_1 = C_1(m), \ldots, z_{q_1} = C_{q_1}(m)$ where m is the challenge message output by the adversary, and the tokens $sk_1, \ldots, sk_{q_1}$ generated by KeyGen to answers A_0's token queries, does the following:

  Sim_0 chooses values $\tilde{m}_1, \ldots, \tilde{m}_q \in \{0,1\}^n$ as follows. For $i = 1, \ldots, q_1$, if $z_i = 1$, Sim_0 sets $\tilde{m}_i = r_i$; otherwise, if $z_i = 0$, Sim_0 picks $\tilde{m}_i$ at random in $\{0,1\}^n$. Furthermore for $i = q_1 + 1, \ldots, q$, Sim_0 picks $\tilde{m}_i$ at random in $\{0,1\}^n$.

  Finally, Sim_0 sets $\tilde{m} = (\perp, \tilde{m}_1, \ldots, \tilde{m}_q)$ and computes $\tilde{\mathsf{Ct}}$ as $\tilde{\mathsf{Ct}} = \mathsf{IndFE.Enc}(mpk, \tilde{m})$ and stores $\tilde{m}_1, \ldots, \tilde{m}_q$ in the state.

- Sim_2 on input the master secret key msk, Boolean circuit C_i and $z_i = C_i(m)$, for $q_1 < i \leq q$, does the following:

  If $z_i = 0$ then Sim_2 picks random $r_i \in \{0,1\}^n$; otherwise, if $z_i = 1$, Sim_1 sets $r_i = \tilde{m}_i$. Sim_2 then sets sk_i as the pair $(r_i, \mathsf{IndFE.KeyGen}(msk, C_i^{r_i}))$.

We now prove that Sim is a good simulator meaning that for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ and $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$ are computationally indistinguishable.

For sake of contradiction, suppose there exists a distinguisher $\mathcal{D}$ that distinguishes with non-negligible probability the output distribution of a certain $\mathcal{A}$ in experiment $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ and in experiment $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$. Then, $\mathcal{A}$ and $\mathcal{D}$ can be used to construct a successful IND adversary $\mathcal{B}$ for IndFE. Specifically, $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$ does the following.

- $\mathcal{B}_0$ on input the public parameters mpk generated by $\mathsf{IndFE.Setup}$, runs $\mathcal{A}_0$ on input mpk.

  Then, $\mathcal{B}_0$ answers $\mathcal{A}_0$'s query C_i for $i \in [q_1]$ by using its own oracle $\mathsf{IndFE.KeyGen}(msk, \cdot)$ as follows: $\mathcal{B}_0$ chooses a random $r_i \in \{0,1\}^n$ and outputs the pair (r_i, sk_i) where $sk_i = \mathsf{IndFE.KeyGen}(msk, C_i^{r_i})$.

  Eventually, $\mathcal{A}_0$ outputs a message m and the state $\mathtt{aux}$. $\mathcal{B}_0$ sets $x = (m, m_1, \ldots, m_q)$ and $x' = (\perp, m'_1, \ldots, m'_q)$ where the values $m_1, \ldots, m_q$ are chosen at random in $\{0,1\}^n$ and the values $m'_1, \ldots, m'_q$ are defined as follows. For $i = 1$ to q_1:

  - If $C_i(m) = 0$, picks m'_i at random in $\{0,1\}^n$,
  - if $C_i(m) = 1$, sets $m'_i = r_i$.

  Furthermore for $q_1 < i \le q$, Sim_0 pick m'_i at random in $\{0,1\}^n$.

  Finally, $\mathcal{B}_0$ outputs $(x, x', \mathtt{st})$; i.e., x and x' are the two challenge plaintexts and $\mathtt{st}$ is $\mathcal{B}$'s state.

- $\mathcal{B}_1$ on input the public parameters mpk, a ciphertext Ct and the state $\mathtt{st}$, runs $\mathcal{A}_1$ on the same inputs.

  Then, $\mathcal{B}_1$ answers $\mathcal{A}_1$'s queries by using its own oracle $\mathsf{IndFE.KeyGen}(msk, \cdot)$ as follows: Given a query C_i, for $i = q_1 + 1, \ldots, q$,

  - If $C_i(m) = 0$, $\mathcal{B}_1$ picks random $r_i \in \{0,1\}^n$,
  - if $C_i(m) = 1$, sets $r_i = m'_i$.

  Then, $\mathcal{B}_1$ outputs (r_i, sk_i) where $sk_i = \mathsf{IndFE.KeyGen}(msk, C_i^{r_i})$.

  Eventually, $\mathcal{A}_1$ outputs α, then $\mathcal{B}_1$ invokes $\mathcal{D}$ on input (mpk, x, α) and returns $\mathcal{D}$'s guess as its own guess.

We now show that, except with negligible probability, $x = (m, m_1, \ldots, m_q)$ and $x' = (\perp, m'_1, \ldots, m'_q)$ are such that, for all $i = 1, \ldots, q$, $C_i^{r_1}(x) = C_i^{r_1}(x')$ and thus $\mathcal{B}$ is a valid IND adversary.

Recall that $C_i^{r_i}(m, m_1, \ldots, m_q) = 1$ iff $r_i = m_j$ or $C_i(m) = 1$ for some $j \in [q]$, and that $m \neq \perp$. For each $i \in [q]$, we distinguish two mutually exclusive cases.

- $C_i^{r_i}(x) = 0$. In this case, we know that $C_i(m) = 0$ since $m \neq \perp$. On the other side, since $m'_1, \ldots, m'_q$ are randomly chosen and independent from r_i, from the definition of $C_i^{r_i}$, it follows that, except with negligible probability, $C_i^{r_i}(x') = 0$.

- $C_i^{r_i}(x) = 1$. Suppose toward a contradiction that with non-negligible probability $C_i(m) = 0$. Thus $C_i^{r_i}(x) = 1$ because $r_i = m_j$ for some $j \in [q]$ given that $m \neq \perp$. But when $C_i(m) = 0$ then r_i is randomly and independently chosen. Thus, with negligible probability, $r_i \neq m_j$ for all $j \in [q]$. Thus, $C_i^{r_i}(x) = 0$. Contradiction. Then, it must be that $C_i(m) = 1$. Thus, it holds that $r_i = m'_j$ for some $j \in [q]$, so that $C_i^{r_i}(x') = 1$ too.

Now notice that if the challenger of the IND-Security game encrypts x then $\mathcal{B}$ is simulating $\mathcal{A}$'s output distribution in the $\mathsf{RealExp}^{\mathsf{SimFE},\mathcal{A}}$ experiment. Instead if x' is encrypted then $\mathcal{B}$ is simulating Sim's output distribution in the $\mathsf{IdealExp}^{\mathsf{SimFE},\mathcal{A}}_{\mathsf{Sim}}$ experiment.

Finally, for $\mathcal{A}$ to be successful it is important that $x \neq x'$, which holds with non-negligible probability. By the hypothesis on $\mathcal{D}$, this contradicts our initial assumption that IndFE is IND-Secure. ▌

Comparison with the construction of Gorbunov, Vaikuntanathan, and Wee [GVW12]. Gorbunov, Vaikuntanathan, and Wee [GVW12], building on the construction of Sahai and Seyalioglu [SS10], gave a functional encryption scheme for all polynomial size circuits that is simulation-based secure against adaptive adversaries that have access to a *bounded* number of tokens. We stress that they assume a (one-query) SIM-Secure functional encryption scheme whereas we rely only on the existence of IND-Secure functional encryption schemes. Furthermore, their construction results in a scheme with a ciphertext of size super linear (at least *quadratic* even in the case of $\mathsf{NC1}$ circuits) in q whereas our construction has ciphertext of size *linear* in q.

6.2 Adaptive Security for HVE

In this section, we show how to construct a HVE scheme whose semantic security can be proved under static assumptions in the bilinear pairing setting in the standard model. We show our construction secure against *adaptive* adversaries obtaining *one* ciphertext and asking an *unbounded* number of tokens.

One possible avenue for obtaining (non-adaptively) simulation-based secure functional encryption scheme for HVE could be via the notion of pre-image samplability introduced by O'Neill [O'N10]. In Section 6.2.1 we prove that HVE is unlikely to be pre-image sampleable. We then embark into the construction of an adaptively secure simulation-based functional encryption scheme for HVE.

6.2.1 Pre-Image Samplability

In this section we prove that if HVE is pre-image sampleable then we can decide 3SAT in polynomial time. We start by reviewing the notion of a *pre-image sampleable* functionality by O'Neill [O'N10].

Definition 6.2.1 [O'N10] Functionality $F : K \times M \to \{0, 1\}$ is *pre-image sampleable* (PS, for short) if there exists a *sampler* algorithm Sam such that for all *PPT adversaries* $\mathcal{A}$,

$$\text{Prob}\left[\left(m, (k_i)_{i=1}^{\ell=\text{poly}(\lambda)}\right) \leftarrow \mathcal{A}(1^\lambda);\, m' \leftarrow \mathsf{Sam}(1^\lambda, |m|, (k_i, F(k_i, m))_{i=1}^{\ell}) : \\ F(k_i, m) = F(k_i, m') \text{ for } i = 1, \ldots, \ell\right] = 1 - \text{negl}(\lambda)\,.$$

For $q > 0$, we say that F is *q-pre-image sampleable* (q-PS, for short) if the sampler algorithm Sam is guaranteed to work only with respect to adversaries $\mathcal{A}$ that output at most q pairs (k_i, b_i).

The following theorem is from [O'N10].

Theorem 6.2.2 If functionality F is PS then any IND-Secure functional encryption scheme $\mathcal{FE}$ for F is also NA-SIM-Secure.

Next we prove that the fact that the HVE functionality is PS has unexpected complexity-theoretic consequences.

Theorem 6.2.3 If HVE is PS then SAT can be decided in probabilistic polynomial time.

Proof: Let Sam be a sampler algorithm for HVE and consider the following algorithm $\mathcal{B}$ that, on input a Boolean formula $\Phi = \phi_1 \wedge \ldots \wedge \phi_c$ in CNF with c clauses and ℓ variables $x_1, \ldots, x_\ell$, performs the following computation. In the description of $\mathcal{B}$, we will identify ℓ-bit strings with truth assignment to variables $x_1, \ldots, x_\ell$.

For each clause ϕ_i of Φ, $\mathcal{B}$ computes key $k_i = (k_{1,i}, \ldots, k_{\ell,i}) \in \{0, 1, \star\}^\ell$ in the following way. Let $h \in \{0,1\}^\ell$ be a truth assignment that does not satisfy the i-th clause ϕ_i. For $j = 1, \ldots, \ell$, $\mathcal{B}$ sets $k_{j,i}$ in the following way

$$k_{j,i} = \begin{cases} h_j, & \text{if } x_j \text{ appears in } \phi_i; \\ 1 - h_j, & \text{if } \bar{x}_j \text{ appears in } \phi_i; \\ \star, & \text{otherwise.} \end{cases}$$

Keys $k_1, \ldots, k_c$ enjoy the following (easy to verify) property. Let $m \in \{0,1\}^\ell$ be a truth assignment over the variables $x_1, \ldots, x_\ell$. Then, m satisfies Φ if and only if $\mathsf{HVE}(m, k_i) = 0$ for $i = 1, \ldots, c$.

$\mathcal{B}$ then runs algorithm Sam on input $((k_1, 0), \ldots, (k_c, 0))$ and let m be Sam's output. If m is a satisfying truth assignment for Φ then $\mathcal{B}$ decides that Φ is satisfiable. Otherwise, $\mathcal{B}$ decides that Φ is not satisfiable.

Let us now prove that $\mathcal{B}$'s output is correct with high probability. Notice that Sam is only guaranteed to work if it is given in input a sequence (k_i, b_i) for which there exists an $m \in \{0,1\}^\ell$ such that $b_i = \mathsf{HVE}(m, k_i)$. So we distinguish two cases. In the first case, we assume that Φ is satisfiable. Then the input of Sam is exactly as required by Definition 6.2.1 and thus, except with negligible probability, Sam outputs m such that $\mathsf{HVE}(m, k_i) = 0$ for $i = 1, \ldots, c$. By our previous observation such an m is a satisfying assignment for Φ and $\mathcal{B}$ is correct.

Suppose instead that Φ is not satisfiable. Then Sam will not output a satisfying assignment and $\mathcal{B}$ is correct. ▌

6.2.2 Complexity Assumptions

In our construction we will make hardness assumptions for bilinear settings whose order N is product of five distinct primes each of length $\Theta(\lambda)$.

Assumption 1 **(General Diffie-Hellman for composite order groups).** The assumption is a kind of Diffie-Hellman assumption in the bilinear setting of composite order. Specifically, for a given composite-order bilinear setting generator CBSGen, define the following distribution: Pick a random bilinear setting $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 5)$ and then pick

$$A_1 \leftarrow \mathbb{G}_{p_1},\ A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4},\ A_5 \leftarrow \mathbb{G}_{p_5},\ \alpha, \beta \leftarrow \mathbb{Z}_N\ ,$$

$$T_0 = A_1^{\alpha\beta} \cdot D_4,\ T_1 \leftarrow \mathbb{G}_{p_1 p_4}\ ,$$

and set $D = (N, A_1, A_2, A_3, A_4, A_5, A_1^\alpha \cdot B_4, A_1^\beta \cdot C_4)$. We define the advantage of any $\mathcal{A}$ in breaking Assumption 1 to be

$$\mathbf{Adv}_1^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_0) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_1) = 1]|\ .$$

Definition 6.2.4 We say that the Assumption 1 holds for generator CBSGen if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathbf{Adv}_1^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

6.2.3 The Scheme

In this section we describe our HVE scheme. To make our description and proofs simpler, we add to all vectors $\vec{x}$ and $\vec{y}$ two dummy components and set both of them equal to 0. We can thus assume that all vectors have at least two non-star positions.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm randomly chooses a description of composite-order bilinear setting $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 5)$ with known factorization, and random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, and, for $i \in [\ell]$ and $b \in \{0,1\}$, random $t_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b} \ .$$

The public key is $mpk = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}]$, and the master secret key is $msk = [g_{12}, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$. The algorithm returns (mpk, msk).

$\mathsf{KeyGen}(msk, \vec{y})$: Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. The key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ under the constraint that $\sum_{i \in S_{\vec{y}}} a_i = 0$. For $i \in S_{\vec{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i \ .$$

The algorithm returns ciphertext $\mathsf{Ct} = (Y_i)_{i \in S_{\vec{y}}}$.

Here we use the fact that $S_{\vec{y}}$ has size at least 2.

$\mathsf{Enc}(mpk, \vec{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets

$$X_i = T_{i,x_i}^s \cdot Z_i \ ,$$

and returns the token $sk_{\vec{y}} = (X_i)_{i\in[\ell]}$.

$\mathsf{Eval}(mpk, \mathsf{Ct}, sk_{\vec{y}})$: The test algorithm computes

$$T = \prod_{i \in S_{\vec{y}}} \hat{e}(X_i, Y_i) \ .$$

It returns TRUE if $T = 1$, FALSE otherwise.

Correctness. It easy to see that the scheme is correct.

Remark 6.2.5 In our construction the computation of the HVE predicate is performed in the $\mathbb{G}_{p_1}$ subgroup. All the other subgroups are used during the proof of security and let us able to change the distribution of the real experiment to one that is computationally indistinguishable and easy to simulate. Notice that at this stage the $\mathbb{G}_{p_2}$ subgroup can be removed from our construction at the expense of introducing another game in the security proof. To simplify the proof we have decided to include that subgroup directly.

6.2.4 Proof of Security

In this section we prove the following theorem.

Theorem 6.2.6 Under the General Subgroup Decision Assumption and Assumption 1, the HVE scheme described in Section 6.2.3 is SIM-Secure in the sense of Definition 2.2.2.

We next give an overview of the proof of the theorem and break down the proof into 4 lemmata.

Our simulator Sim consists of three algorithms: Sim.Setup, Sim.Enc, and Sim.KeyGen that share a common state. Informally,

Sim.Setup produces two public keys mpk and mpk'. mpk is given to the adversary (that may use it to generate his own ciphertexts) and mpk' instead is used to generate the simulated ciphertext.

Sim.Enc is used to compute the simulated ciphertext and takes as input the public key mpk' and the sequence $(\vec{y}_k, z_k)_{k=1}^{q_1}$, where $z_k = \mathsf{HVE}(\vec{y}_k, \vec{x})$, $\vec{x}$ is the plaintext output by the adversary and q_1 is the number of queries asked by the adversary in the first stage.

Sim.KeyGen instead is used to answer the adaptive queries asked by the adversary after seeing the simulated ciphertext. When the adversary asks to see a token for $\vec{y}$, Sim.KeyGen is invoked with the master secret key msk, the value $\vec{y}$ and the value $\mathsf{HVE}(\vec{x}, \vec{y})$, where $\vec{x}$ is the challenge ciphertext output by the adversary. We stress that the simulator does not have access to $\vec{x}$. Thus we have to prove indistinguishability of the following two experiments.

Therefore, to prove Theorem 6.2.6 it is enough to show that the following two experiments are computational indistinguishable.

$\mathsf{RealExp}^{\mathcal{A}}(1^\lambda, 1^\ell)$	$\mathsf{IdealExp}^{\mathcal{A}}_{\mathsf{Sim}}(1^\lambda, 1^\ell)$
$(mpk, msk) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$;	$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;	$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk, \vec{x})$;	$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(mpk', (\vec{y}_k, \mathsf{HVE}(\vec{y}_k, \vec{x}))_{k=1}^{q_1})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(msk,\cdot)}(mpk, \mathsf{Ct}, \mathsf{aux})$;	$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
Output: $(mpk, \vec{x}, \alpha)$	**Output:** $(mpk, \vec{x}, \alpha)$

More formally, we next describe the three algorithms used by Sim.

Algorithm Sim.Setup constructs two public keys mpk (that is given to the adversary) and mpk' (that is used to produce the simulated ciphertext) and a master secret key msk (that is used to answer the queries of the adversary). In public key mpk the $t_{i,b}$'s are encoded in the $\mathbb{G}_{p_2}$ part of the $T_{i,b}$'s (instead of the $\mathbb{G}_{p_1}$ part as in normal public key) whereas mpk' is generated correctly. Then, notice that the simulated ciphertext is independent from mpk, and this gives us enough freedom to manipulate the $\mathbb{G}_{p_1}$ part of the simulated ciphertext and tokens the adversary sees in the second stage. The subgroup $\mathbb{G}_{p_3}$ will help us to hide this transition.

Specifically, Sim.Setup randomly chooses a description of a composite-order bilinear group $(N, \mathbb{G}, g, \mathbb{G}_T, \hat{e}) \leftarrow \mathsf{CBSGen}(1^\lambda, 5)$ with known factorization, and random $g_1 \in$

$\mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$, $g_5 \in \mathbb{G}_{p_5}$. Then, for $i \in [\ell]$ and $b \in \{0, 1\}$, Sim.Setup picks random $t_{i,b} \in Z_N$, $R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets

$$T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}, \quad T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b} .$$

The public keys are

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad mpk' = [N, g_3, g_5, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and the master secret key is

$$msk = [g_{12}, g_4, g_5, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

where $g_{12} = g_1 \cdot g_2$. Sim.Setup returns (mpk, mpk', msk).

Algorithm Sim.Enc simulates the encryption of the challenge plaintext $\vec{x}$ provided by the adversary. It receives on input the queries $\vec{y}_k$, $k = 1, \ldots, q_1$, asked by the adversary in the first stage (the non-adaptive one) and the values $z_k = \mathsf{HVE}(\vec{y}_k, \vec{x})$, for $k = 1, \ldots, q_1$.

Sim.Enc does not have access to $\vec{x}$ and by Theorem 6.2.3 the pre-image samplability cannot be used to sample an $\vec{x}'$ such that $\mathsf{HVE}(\vec{y}_k, \vec{x}) = \mathsf{HVE}(\vec{y}_k, \vec{x}')$ for all k. But still some information about $\vec{x}$ can be derived from the inputs. Indeed, observe that if $\mathsf{HVE}(\vec{y}, \vec{x}) = 1$, then $\vec{y}$ and $\vec{x}$ coincide in all non-star entries of $\vec{y}$. For these positions, Sim.Enc computes the ciphertext Ct just like the encryption algorithm; for all remaining positions, the ciphertext output by Sim.Enc has a random $\mathbb{G}_{p_1}$ part.

Notice that such a simulated ciphertext is then compatible with all the tokens the adversary sees in first stage in the sense that $\mathsf{Eval}(mpk, \mathsf{Ct}, sk_{\vec{y}_k}) = \mathsf{HVE}(\vec{y}_k, \vec{x})$, for $k = 1, \ldots, q_1$. Unfortunately the simulation is not perfect. In the second stage, the adversary could ask to see a token $sk_{\vec{y}}$ for a vector $\vec{y}$ such that $\mathsf{HVE}(\vec{y}, \vec{x}) = 1$ but $\mathsf{Eval}(mpk, \mathsf{Ct}, sk_{\vec{y}}) = 0$. This is because the $\mathbb{G}_{p_1}$ part of simulated ciphertext Ct and the token $sk_{\vec{y}}$ does not cancel out correctly upon decryption. A possible solution is to remove the $\mathbb{G}_{p_1}$ part from the matching tokens the adversary sees in the second stage. This is where $\mathbb{G}_{p_5}$ comes to help us. $\mathbb{G}_{p_5}$ will be introduced in the simulated ciphertext and matching second stage tokens in such a way it will cancel out upon decryption and will provide us enough entropy to remove the $\mathbb{G}_{p_1}$ part from the adaptive tokens. Thus, each component of the ciphertext computed by Sim.Enc contains also a random $\mathbb{G}_{p_5}$ part that will be used for constructing the answers to the adaptive queries of the adversary.

More formally, for a sequence $Q = (\vec{y}_k, z_k)_{k=1}^{q_1}$ with $z_k = \mathsf{HVE}(\vec{y}_k, \vec{x})$, for $k = 1, \ldots, q_1$, we let MPos denote the set of indices $1 \le i \le \ell$ for which there exists $k \in \{1, \ldots, q_1\}$ such that $z_k = 1$ and $y_{k,i} \neq \star$. Notice that, by the observation above, for all $i \in \mathsf{MPos}$, the value x_i can be derived from the sequence Q. Then, $\mathsf{Sim.Enc}(mpk', (\vec{y}_k, z_k)_{k=1}^{q_1})$ can be described as follows. The algorithm parses mpk' as $mpk' = [N, g_3, g_5, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and, for all $i \in \mathsf{MPos}$, derives the value x_i from the input sequence. Then, the algorithm chooses random $s \in \mathbb{Z}_N$ and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$, and, for each $i \in [\ell]$:

- if $i \notin \mathsf{MPos}$, randomly select $r_i \in \mathbb{Z}_N$ and set $X_i = T_{i,0}^{r_i} \cdot g_5^{s_i} \cdot Z_i$.
- if $i \in \mathsf{MPos}$, set $X_i = T_{i,x_i}^{s} \cdot g_5^{s_i} \cdot Z_i$.

Sim.Enc returns one output, the challenge ciphertext $\mathsf{Ct} = (X_i)_{i\in[\ell]}$, and keeps the vector $(s_i)_{i\in[\ell]}$ in the state so that it can be used by algorithm Sim.KeyGen.

Real	H_1	H_2	H_3	Ideal
Setup	Sim.Setup	Sim.Setup	Sim.Setup	Sim.Setup
Enc	Enc	Sim.A.Enc	Sim.A.Enc	Sim.Enc
KeyGen	KeyGen	KeyGen	Sim.KeyGen	Sim.KeyGen

Figure 6.1: The hybrids used in the proof of security.

Algorithm Sim.KeyGen simulates the answer to the second stage queries of the adversary. It receives as input the master secret key msk, the vector $\vec{y}$ for which the token has to be simulated and the value $z = \mathsf{HVE}(\vec{y}, \vec{x})$, where $\vec{x}$ is the challenge plaintext.

For each $j \in S_{\vec{y}}$, Sim.KeyGen selects random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$. Then, it distinguishes two cases.

- $z = 0$: In this case, the algorithm generates a token with a random $\mathbb{G}_{p_1}$ part. Specifically, for each $j \in S_{\vec{y}}$, Sim.KeyGen chooses random $c_j \in \mathbb{Z}_N$ and sets

$$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

- $z = 1$: In this case, the algorithm generates a token without the $\mathbb{G}_{p_1}$ part and with a $\mathbb{G}_{p_5}$ part that will cancel out against the simulated ciphertext upon decryption. Specifically, for each $j \in S_{\vec{y}}$, Sim.KeyGen sets

$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

We remind the reader that the vector $(s_j)_{j \in [\ell]}$ is the same vector used by Sim.Enc (and is stored in the state of the simulator). Also, note that the key output by Sim.KeyGen behaves correctly when matched against the ciphertext output by Sim.Enc.

To prove that the real and the ideal experiments are indistinguishable we use three intermediate hybrids that we call H_1, H_2 and H_3. Then, the proof consists of four main steps (see Figure 6.2.4 for a quick reference).

The first step of our proof consists in constructing the public key by means of Sim.Setup. This has the effect of projecting the public key (and thus the ciphertexts the adversary constructs by himself) to a different subgroup from the one of the challenge ciphertext. Specifically:

```
$H_1^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(msk,\cdot)}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, \vec{x}, \alpha)$
```

The first step of the proof consists in proving that the outputs of $\mathsf{RealExp}^{\mathcal{A}}$ and $H_1^{\mathcal{A}}$ are indistinguishable for all PPT adversaries $\mathcal{A}$.

The second step modifies the simulated ciphertext by adding a $\mathbb{G}_{p_5}$ part. This will be used in successive experiments to answer to the adversary's adaptive token queries. Specifically:

```
$H_2^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathtt{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{KeyGen}(msk,\cdot)}(mpk, \mathsf{Ct}, \mathtt{aux})$;
**Output:** $(mpk, m, \alpha)$
```

where algorithm Sim.A.Enc proceeds as follows.

Algorithm Sim.A.Enc on input public key $mpk' = [N, g_3, g_5, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, the algorithm chooses random $s \in \mathbb{Z}_N$. Then, for $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$, sets

$$X_i = {T'_{i,x_i}}^{s} \cdot g_5^{s_i} \cdot Z_i \ ,$$

returns the tuple $(X_i)_{i\in[\ell]}$ and stores the vector $(s_i)_{i\in[\ell]}$ in the state so that it can be used by Sim.KeyGen.

The second step of the proof consists in proving that the outputs of $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$ are indistinguishable for all PPT adversaries $\mathcal{A}$.

The third step computes the simulated answers using the Sim.KeyGen algorithm that adds a $\mathbb{G}_{p_5}$ part that cancels out with the one added in the simulated ciphertext.

```
$H_3^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathtt{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathtt{aux})$;
**Output:** $(mpk, \vec{x}, \alpha)$
```

Thus the third step of the proof consists in the defining hybrid experiment $H_3^{\mathcal{A}}$ and in proving that it is indistinguishable from $H_2^{\mathcal{A}}$ for all PPT adversaries $\mathcal{A}$.

The fourth step consists in proving that hybrid $H_3^{\mathcal{A}}$ is indistinguishable from the ideal experiment for all PPT adversaries $\mathcal{A}$.

This concludes the proof of Theorem 6.2.6. In the next sections we give formal details for each step of the proof.

6.2.5 The first step of the proof

In this section we prove that RealGame and PKGame are indistinguishable.

Lemma 6.2.7 If the GSD Assumption holds, then for all PPT adversaries $\mathcal{A}$, $\mathsf{RealGame}^{\mathcal{A}} \approx_c \mathsf{PKGame}^{\mathcal{A}}$.

PROOF. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{RealGame}^{\mathcal{A}}$ and $\mathsf{PKGame}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ which receives the following instance of the GSD Assumption, $(N, T, \{g_3, g_4, g_{13}, g_{12}\}) \leftarrow \mathsf{GSDGen}(1^\lambda, \{1,3\}, \{2,3\}, \{3\}, \{4\}, \{1,3\}, \{1,2\})$ and, depending on the nature of T, simulates $\mathsf{RealGame}^{\mathcal{A}}$ or $\mathsf{PKGame}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets

$$T_{i,b} = T^{t_{i,b}} \quad \text{and} \quad T'_{i,b} = g_{13}^{t_{i,b}} .$$

Then $\mathcal{B}$ sets

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \quad \text{and} \quad mpk' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and

$$msk = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and starts the interaction with $\mathcal{A}$ on input mpk.

Token Queries: $\mathcal{B}$ uses msk to answer the token queries of $\mathcal{A}$.

Ciphertext: $\mathcal{B}$ computes $\mathsf{Ct} = \mathsf{Enc}(mpk', \vec{x})$.

This concludes the description of algorithm $\mathcal{B}$.

The view of $\mathcal{A}$ consists of mpk, Ct and the answer of the queries. The answers of the queries are distributed, independently from the nature of the challenge T, as in $\mathsf{RealGame}^{\mathcal{A}}$ and as in $\mathsf{PKGame}^{\mathcal{A}}$.

If $T \in \mathbb{G}_{p_1p_3}$ then mpk is constructed exactly as in $\mathsf{RealGame}^{\mathcal{A}}$. Moreover, the ciphertext Ct, even though is constructed using mpk', has the same distribution of a ciphertext constructed using mpk and thus the view of $\mathcal{A}$ is the same as in $\mathsf{RealGame}^{\mathcal{A}}$.

On the other hand, when $T \in \mathbb{G}_{p_2p_3}$ then mpk and Ct are distributed as in $\mathsf{PKGame}^{\mathcal{A}}$. □

6.2.6 The second step of the proof

In this section we prove that PKGame is indistinguishable from TypeACtGame.

Lemma 6.2.8 If GSD Assumption holds then, for all PPT adversaries $\mathcal{A}$, $\mathsf{PKGame}^{\mathcal{A}} \approx_c \mathsf{TypeACtGame}^{\mathcal{A}}$.

PROOF. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{PKGame}^{\mathcal{A}}$ and $\mathsf{TypeACtGame}^{\mathcal{A}}$ are not indistinguishable. Then, we show a PPT algorithm $\mathcal{B}$ which receives the following instance of the GSD Assumption, $(N, T, \{g_1, g_2, g_3, g_4\}) \leftarrow \mathsf{GSDGen}(1^\lambda, \{3\}, \{3,5\}, \{1\}, \{2\}, \{3\}, \{4\})$ and, depending on the nature of T, simulates $\mathsf{PKGame}^{\mathcal{A}}$ or $\mathsf{TypeACtGame}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{B}$ sets

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and

$$msk = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and starts the interaction with $\mathcal{A}$ on input mpk.

Token Queries: $\mathcal{B}$ runs algorithm KeyGen on input msk to answer token queries.

Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for $\vec{x}$ as follows. $\mathcal{B}$ chooses, for $i \in [\ell]$, random $r_i \in \mathbb{Z}_N$ and random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_1^{t_{i,x_i}} \cdot T^{r_i} \cdot Z_i$$

This concludes the description of algorithm $\mathcal{B}$.

Finally it is easy to see that if $T \in \mathbb{G}_{p_3}$ then Ct is distributed as in $\mathsf{PKGame}^{\mathcal{A}}$ and if $T \in \mathbb{G}_{p_3p_5}$ then Ct is distributed as in $\mathsf{TypeACtGame}^{\mathcal{A}}$. □

6.2.7 The third step of the proof

In this section we prove that H_2 is indistinguishable from H_3.

Lemma 6.2.9 Under the GSD Asummption and Assumption 5, for all PPT adversaries $\mathcal{A}$, $H_2^{\mathcal{A}} \approx_c H_3^{\mathcal{A}}$.

The proof of the lemma above consists of three sub-steps.

First Substep. In the *first substep*, we define a new experiment called TypeAKeysGame which differs from TypeACtGame in the way the token queries of the second stage are answered. More specifically, the token for vector $\vec{y}$ such that $\mathsf{HVE}(\vec{x}, \vec{y}) = 1$ contains a $\mathbb{G}_{p_5}$ part that is related with the one of the simulated ciphertext. The remaining tokens are like those in TypeACtGame. We will prove that, under Assumption 3, TypeACtGame $\approx_c$ TypeAKeysGame.

Second Substep. In the *second substep*, we define a new experiment called TypeBKeysGame which differs from TypeAKeysGame in the way the token for matching queries of the second stage are generated. More specifically, if the adversary asks for the token for a vector $\vec{y}$ such that $\mathsf{HVE}(\vec{x}, \vec{y}) = 1$, then the answer does not contain a $\mathbb{G}_{p_1}$ part. The tokens for $\vec{y}$ such that $\mathsf{HVE}(\vec{x}, \vec{y}) = 0$ instead are computed as in TypeAKeysGame. We will prove that, under Assumption 4, TypeAKeysGame $\approx_c$ TypeBKeysGame.

Third Substep. Let q_2 be the number of token queries made by the adversary in the second stage. In the *third substep* we define, for $k = 0, \ldots, q_2$, a new experiment called $H_{3,k}$ which differs from TypeBKeysGame in the way the the second stage token queries are answered. More specifically, the first k tokens asked by the adversary are modified in the following way. The token for $\vec{y}$ such that $\mathsf{HVE}(\vec{x}, \vec{y}) = 0$ contains a random $\mathbb{G}_{p_1}$ part. Instead if $\mathsf{HVE}(\vec{x}, \vec{y}) = 1$ then the token is computed as in TypeBKeysGame. The tokens for the remaining $q_2 - k$ queries are computed like in experiment TypeBKeysGame. We observe that TypeBKeysGame $= H_{3,0}$ and that $H_{3,q_2} = H_3$.

We will prove that, under Assumption 5, $H_{3,k-1} \approx_c H_{3,k}$.

The first substep

Let us start by formally defining experiment TypeAKeysGame in terms of the algorithm Sim.A.KeyGen used to answer the token queries of the second stage.

Algorithm Sim.A.KeyGen receives in input msk, $\vec{y}$ and $z = \mathsf{HVE}(\vec{y}, \vec{x})$ where $\vec{x}$ is the challenge plaintext provided by the adversary and distinguishes two cases.

- $z = 0$: The answer to the query is computed by running KeyGen on input msk and $\vec{y}$.
- $z = 1$: For each $j \in S_{\vec{y}}$, the algorithm chooses random $W_j \in \mathbb{G}_{p_4}$, random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}_i}} a_j = 0$, and sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \;.$$

Notice that the vector $(s_j)_{j\in[\ell]}$ is the vector stored in the state by Sim.A.Enc.

We define hybrid experiment TypeAKeysGame$^{\mathcal{A}}$ as follows.

```
$\mathsf{TypeAKeysGame}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.A.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, m, \alpha)$
```

Lemma 6.2.10 If the GSD Assumption holds then, for all PPT $\mathcal{A}$, $H_2^{\mathcal{A}} \approx_c \mathsf{TypeAKeysGame}^{\mathcal{A}}$.

PROOF. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{TypeACtGame}^{\mathcal{A}}$ and $\mathsf{TypeAKeysGame}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ that receives the following instance of the GSD Assumption, $(N, T, \{g_1, g_2, g_3, g_4, g_{15}\}) \leftarrow \mathsf{GSDGen}(1^\lambda, \{1\}, \{1,5\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1,5\})$ and, depending on the nature of T, simulates $\mathsf{TypeACtGame}^{\mathcal{A}}$ or $\mathsf{TypeAKeysGame}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{B}$ sets

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}] ,$$

and

$$msk = [g_{12}, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}] ,$$

and starts the interaction with $\mathcal{A}$ on input mpk.

First Stage Token Queries: $\mathcal{B}$ computes the token for $\vec{y}$ by running KeyGen on input msk and $\vec{y}$.

Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for vector $\vec{x}$ as follows. $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_{15}^{s \cdot t_{i,x_i}} \cdot Z_i$$

Second Stage Token Queries: $\mathcal{B}$ generates the token for $\vec{y}$ in the following way.

If $\mathsf{HVE}(\vec{x}, \vec{y}) = 0$, the token is computed by running KeyGen on input msk and $\vec{y}$.

If $\mathsf{HVE}(\vec{x}, \vec{y}) = 1$, the token is computed as follows. For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j\in S_{\vec{y}}} a_j = 0$. Then $\mathcal{B}$, for each $j \in S_{\vec{y}}$, sets:

$$Y_j = T^{a_j/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j .$$

This concludes the description of algorithm $\mathcal{B}$.

Now we observe that the output of $\mathcal{B}$'s setup is distributed like the output of algorithm $\mathsf{Sim.Setup}$ and thus like in H_2 and $\mathsf{TypeAKeysGame}$. Similarly, the simulated ciphertext is distributed like the output of $\mathsf{A.Enc}$ on input mpk' and $\vec{x}$ and thus exactly as in H_2 and $\mathsf{TypeAKeysGame}$. Finally, let us consider the answers to the second stage queries. If $T \in \mathbb{G}_{p_1}$ then the answer to the query for $\vec{y}$ has the same distribution as the output of KeyGen on input msk and $\vec{y}$ and thus it is distributed as in H_2. On the other hand, if $T \in \mathbb{G}_{p_1p_5}$ then the answer to the query for $\vec{y}$ has the same distribution as the output of $\mathsf{Sim.A.KeyGen}$ on input $msk, \vec{y}$ and $\mathsf{HVE}(\vec{x}, \vec{y})$ just like in $\mathsf{TypeAKeysGame}$. □

The second substep

Let us start by formally defining experiment $\mathsf{TypeBKeysGame}$ in terms of the algorithm $\mathsf{Sim.B.KeyGen}$ that is used to answer the token queries of the second stage.

Algorithm $\mathsf{Sim.B.KeyGen}$ receives in input msk, $\vec{y}$ and $z = \mathsf{HVE}(\vec{y}, \vec{x})$, where $\vec{x}$ is the challenge plaintext provided by the adversary and distinguishes two cases.

- $z = 0$: The answer to the query is computed by running KeyGen on input msk and $\vec{y}$.
- $z = 1$: For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$ and sets

$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

Notice that vector $(s_j)_{j\in[\ell]}$ is the vector stored in the state by $\mathsf{Sim.A.Enc}$.

We define hybrid experiment $\mathsf{TypeBKeysGame}^{\mathcal{A}}$ as follows.

```
$\mathsf{TypeBKeysGame}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.B.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, m, \alpha)$
```

Lemma 6.2.11 If the GSD Assumption holds then, for all PPT $\mathcal{A}$, $\mathsf{TypeAKeysGame}^{\mathcal{A}} \approx_c \mathsf{TypeBKeysGame}^{\mathcal{A}}$.

PROOF. Suppose there exists an adversary $\mathcal{A}$ for which $\mathsf{TypeAKeysGame}^{\mathcal{A}}$ and $\mathsf{TypeBKeysGame}^{\mathcal{A}}$ are distinguishable. Then, we show a PPT algorithm $\mathcal{B}$ that receives the following instance of the GSD Assumption, $(N, T, \{g_2, g_3, g_4, g_{14}, g_{15}\}) \leftarrow \mathsf{GSDGen}(1^\lambda, \{1,4,5\}, \{4,5\}, \{2\}, \{3\}, \{4\}, \{1,4\}, \{1,5\})$ and, depending on the nature of T, simulates $\mathsf{TypeAKeysGame}^{\mathcal{A}}$ or $\mathsf{TypeBKeysGame}^{\mathcal{A}}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{B}$ sets

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}] \ ,$$

and

$$msk = [\bot, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}] \ ,$$

and starts the interaction with $\mathcal{A}$ on input mpk. Notice that $\mathcal{B}$ is unable to compute a complete msk as it does not have access to an element from $\mathbb{G}_{p_1p_2}$.

First Stage Token Queries: $\mathcal{B}$ generates the token for $\vec{y}$ in the following way. For each $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_i \in \mathbb{G}_{p_4}$, random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = 0$ and then sets

$$Y_i = g_{14}^{a_i/t_{i,y_i}} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i \ .$$

Ciphertext: $\mathcal{B}$ generates the simulated ciphertext for vector $\vec{x}$ as follows. $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_{15}^{s \cdot t_{i,x_i}} \cdot Z_i$$

Second Stage Token Queries: On input $msk, \vec{y}$ and $z = \mathsf{HVE}(\vec{x}, \vec{y})$, $\mathcal{B}$ generates the token for vector $\vec{y}$ in the following way.

For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_y} a_j = 0$. Then $\mathcal{B}$ distinguishes two cases.

$z = 0$**:** For each $j \in S_{\vec{y}}$, $\mathcal{B}$ sets

$$Y_j = g_{14}^{a_j/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

$z = 1$**:** For each $j \in S_{\vec{y}}$, $\mathcal{B}$ sets

$$Y_j = T^{a_j/t_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

This concludes the description of algorithm $\mathcal{B}$.

Let us now look at the answers to the token queries of the second stage. It is easy to see that queries for $\vec{y}$ such that $\mathsf{HVE}(\vec{x}, \vec{y}) = 0$ are distributed as the output of KeyGen. Suppose that $T \in \mathbb{G}_{p_1p_4p_5}$. Then the answers to matching queries are distributed like in the output of algorithm $\mathsf{Sim.A.KeyGen}$ and thus like in $\mathsf{TypeAKeysGame}$. On the other hand, if $T \in \mathbb{G}_{p_4p_5}$ then the answers to matching queries are distributed like in the output of algorithm $\mathsf{Sim.B.KeyGen}$ and thus like in $\mathsf{TypeBKeysGame}$.

$\square$

The third substep

In this section we prove that, for any PPT adversary $\mathcal{A}$, $\mathsf{TypeBKeysGame}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ are indistinguishable, under Assumption 5.

For any adversary $\mathcal{A}$ that makes q_2 queries in the second stage, we define a sequence of hybrid experiments $H_{3,k}^{\mathcal{A}}$, for $k = 0, \ldots, q_2$, such that

$$H_{3,0}^{\mathcal{A}} = \mathsf{TypeBKeysGame}^{\mathcal{A}} \quad \text{and} \quad H_{3,q_2}^{\mathcal{A}} = H_3^{\mathcal{A}} \ .$$

Then if $\mathsf{TypeBKeysGame}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ can be distinguished then there must exist $k \in \{1, \ldots, q_2\}$ such that hybrid experiments $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ are indistinguishable. Moreover, from the definition of experiment $H_{3,k}$, it is clear that if k-th token query of the second stage is a matching query (that is, $\mathsf{HVE}(\vec{x}, \vec{y}) = 1$) then $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ coincide. Therefore, if $\mathcal{A}$ is such that $\mathsf{TypeBKeysGame}^{\mathcal{A}}$ and $H_3^{\mathcal{A}}$ can be distinguished there must exist k such that $H_{3,k-1}^{\mathcal{A}}$ and $H_{3,k}^{\mathcal{A}}$ can be distinguished and the k-th token query of $\mathcal{A}$ in the second stage is with non-negligible probability a non-matching query (that is, $\mathsf{HVE}(\vec{x}, \vec{y}) = 0$).

Let us now define experiment $H_{3,k}$. In $H_{3,k}$ second stage token queries are answered by running a parametrized version of algorithm $\mathsf{Sim.KeyGen}$ that, with a slight abuse of notation, we also call $\mathsf{Sim.KeyGen}$.

Algorithm $\mathsf{Sim.KeyGen}$ receives as input the master secret key msk, the vector $\vec{y}$ for which the token must be computed and the value $z = \mathsf{HVE}(\vec{x}, \vec{y})$. In addition, $\mathsf{Sim.KeyGen}$ receives the number i of the query and the value k.

For each $j \in S_{\vec{y}}$, $\mathsf{Sim.KeyGen}$ selects random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$. Then, it distinguishes two cases.

- $z = 0$: The depending on the query index i, $\mathcal{B}$ does the following.
  - If $i \le k$: For each $j \in S_{\vec{y}}$, $\mathsf{Sim.KeyGen}$ chooses random $c_j \in \mathbb{Z}_N$ and sets
  $$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$
  - If $i > k$: For each $j \in S_{\vec{y}}$, $\mathsf{Sim.KeyGen}$ sets
  $$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$
- $z = 1$: For each $j \in S_{\vec{y}}$, $\mathsf{Sim.KeyGen}$ sets
$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \ .$$

  We remind the reader that the vector $(s_j)_{j\in[\ell]}$ is the vector stored in the state of the simulator.

We observe that for algorithm $\mathsf{Sim.B.KeyGen}$ is the parametrized version of $\mathsf{Sim.KeyGen}$ with $k = 0$. On the other hand, algorithm $\mathsf{Sim.KeyGen}$ is the parametrized version of $\mathsf{Sim.KeyGen}$ with $k = q_2$.

Next we define hybrid experiment $H_{3,k}^{\mathcal{A}}$, for an adversary $\mathcal{A}$ that ask q_2 token queries in the second stage and for $k = 0, \ldots, q_2$,

```
$H_{3,k}^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}(mpk', \vec{x})$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot),\cdot,k)}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, \vec{x}, \alpha)$
```

Notice that $H_{3,0} = \mathsf{TypeBKeysGame}$ and $H_{3,q_2} = H_3$.

Lemma 6.2.12 If Assumption 1 holds then, for any PPT adversary $\mathcal{A}$, $\mathsf{TypeBKeysGame}^{\mathcal{A}} \approx_c H_3^{\mathcal{A}}$

PROOF. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{TypeBKeysGame}$ and H_3 are distinguishable. We show a PPT algorithm $\mathcal{B}$ that receives an instance of Assumption 1, consisting of $(N, A_1, A_2, A_3, A_4, A_5, A_1^\alpha \cdot B_4, A_1^\beta \cdot C_4)$ and challenge T, and, depending on the nature of T, simulates $H_{3,k-1}^{\mathcal{A}}$ or $H_{3,k}^{\mathcal{A}}$ with some non-negligible probability for a random $k \in \{1, \ldots, q_2\}$. This suffices to prove the Lemma.

Setup: $\mathcal{B}$ randomly chooses position $\mathsf{j} \in [\ell]$ and $b_\mathsf{j} \in \{0, 1\}$.

$\mathcal{B}$ sets $g_1 = A_1, g_2 = A_2, g_{12} = A_1 \cdot A_2, g_3 = A_3, g_4 = A_4, g_5 = A_5$ and, for each $i \in [\ell]$ and $b \in \{0, 1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b}, R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Finally, $\mathcal{B}$ sets
$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] \ ,$$
and
$$msk = [g_{12}, g_4, , (t_{i,b})_{(i,b)\in([\ell]\times\{0,1\})\setminus\{(\mathsf{j},b_\mathsf{j})\}}]$$
and starts the interaction with $\mathcal{A}$ on input mpk.

First Stage Token Queries: In the first stage, $\mathcal{B}$ generates token for $\vec{y}$ in the following way. For each $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_i \in \mathbb{G}_{p_4}$, random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_{\vec{y}}} a_i = 0$ and then sets

$$Y_i = g_{12}^{a_i/t_{i,y_i}} \cdot W_i \ .$$

However, if $y_{\mathrm{j}} = b_{\mathrm{j}}$ then $\mathcal{B}$ sets

$$Y_{\mathrm{j}} = (A_1^{\beta} \cdot C_4)^{a_i} \cdot g_2^{a_{\mathrm{j}}/t_{\mathrm{j},y_{\mathrm{j}}}} \cdot W_{\mathrm{j}} \ .$$

This last setting has the effect of implicitly setting $t_{\mathrm{j},y_{\mathrm{j}}} \equiv 1/\beta \pmod{p_1}$.

Ciphertext: $\mathcal{B}$ generate the challenge ciphertext for vector $\vec{x}$ in the following way. If $x_{\mathrm{j}} = b_{\mathrm{j}}$ then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ chooses random $s \in \mathbb{Z}_N$ and, for $i \in [\ell]$, random $\mathbb{Z}_i \in \mathbb{G}_{p_3}$, and sets

$$X_i = g_1^{s \cdot t_{i,x_i}} \cdot g_5^{s \cdot t_{i,x_i}} \cdot Z_i.$$

Second Stage Token Queries: At the start of the second stage of token queries, $\mathcal{B}$ picks a random $k \in \{1, \ldots, q_2\}$.

The i-th token query of the second stage for vector $\vec{y}$ is answered by $\mathcal{B}$ in the following way by distinguishing the following two cases.

HVE$(\vec{x}, \vec{y}) = 0$**:** We distinguish between the following three cases

$i < k$**:** For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$, $c_j \in \mathbb{Z}_N$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$ and sets

$$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \ .$$

$i = k$**:** If $y_{\mathrm{j}} \neq b_{\mathrm{j}}$ then $\mathcal{B}$ aborts.
Otherwise let $h \in [\ell]$ be such that $h \neq \mathrm{j}$ and $y_h \neq \star$ and set $S = S_{\vec{y}} \setminus \{\mathrm{j}, h\}$, Notice that such an h always exists since we assumed that each query contains at least two non-$\star$ entries.
Then, for each $j \in S$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and random $a_j \in \mathbb{Z}_N$ and sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$

Then, for position j, $\mathcal{B}$ chooses random $W_{\mathrm{j}} \in \mathbb{G}_{p_4}$ and random $a_{\mathrm{j}} \in \mathbb{Z}_N$ and sets

$$Y_{\mathrm{j}} = T \cdot g_2^{a_{\mathrm{j}}/t_{\mathrm{j},y_{\mathrm{j}}}} \cdot W_{\mathrm{j}}.$$

Finally, for position h, $\mathcal{B}$ sets

$$Y_h = (A_1^{\alpha} B_4)^{-1/t_{h,y_h}} \cdot g_1^{-s/t_{h,y_h}} \cdot g_2^{-(s+a_{\mathrm{j}})/t_{h,y_h}} \cdot W_h,$$

where $s = \sum_{j \in S} a_j$.

$i > k$**:** For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$ and sets

$$Y_j = g_{12}^{a_j/t_{j,y_j}} \cdot W_j \ .$$

In the special case that $y_{\mathrm{j}} = b_{\mathrm{j}}$, then $\mathcal{B}$ sets

$$Y_{\mathrm{j}} = (A_1^{\beta} \cdot C_4)^{a_{\mathrm{j}}} \cdot g_2^{a_{\mathrm{j}}/t_{\mathrm{j},y_{\mathrm{j}}}} \cdot W_{\mathrm{j}} \ .$$

$\mathsf{HVE}(\vec{x}, \vec{y_i}) = 1$**:** For each $j \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $W_j \in \mathbb{G}_{p_4}$ and $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_{\vec{y}}} a_j = 0$ and sets

$$Y_j = g_2^{a_j/t_{j,y_j}} \cdot g_5^{a_j/t_{j,y_j}} \cdot W_j.$$

This concludes the description of algorithm $\mathcal{B}$.

Let us now prove that the probability that $\mathcal{B}$ does not abort while interacting with $\mathcal{A}$ is non-negligible. First observe that the probability that $\mathcal{B}$ aborts while constructing the simulated ciphertext is $1/2$. Indeed, the view of $\mathcal{A}$ up to this point is independent from j and b_{j} and thus, since b_{j} is chosen at random by $\mathcal{B}$, the probability that $x_{\mathrm{j}} = b_{\mathrm{j}}$ is $1/2$. Let us now look at the probability that $\mathcal{B}$ aborts while answering the k-th token query of the second stage of $\mathcal{A}$, given that it has not aborted in the construction of the simulated ciphertext. In this case the view of $\mathcal{A}$ is independent from j and also remember that the probability that the k-th query of $\mathcal{A}$ is non-matching is non-negligible. If this is the case then there must exist one position j such that $y_j \neq \star$ and $y_j \neq x_j$. If $j = \mathrm{j}$ (which happens with non-negligible probability since $\mathcal{A}$'s view is independent from the value of j) then $\mathcal{B}$ does not abort. Indeed, in this case we would have $y_{\mathrm{j}} \neq x_j \neq b_j$ which implies that $y_{\mathrm{j}} = b_j$.

Let us now look at the view of $\mathcal{A}$ while interacting with $\mathcal{B}$. We observe that mpk has the same distribution of the corresponding output of $\mathsf{Sim.Setup}$. Even though $\mathcal{B}$ does not have a complete msk as it misses $t_{\mathrm{j},b_{\mathrm{j}}}$, the answer of the first stage queries are distributed as the output of algorithm KeyGen on input msk in which $t_{\mathrm{j},b_{\mathrm{j}}} \equiv 1/\beta \pmod{p_1}$. Given that $\mathcal{B}$ does not abort, it is straightforward to see that Ct constructed by $\mathcal{B}$ has the same distribution as the output of $\mathsf{Ct} \leftarrow \mathsf{Sim.A.Enc}$ on input mpk' and $\vec{x}$. Let us now look at the answers of the second stage queries. The matching queries and the first $k-1$ non-matching queries have the same distribution of the output of algorithms $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k)$ and $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k - 1)$. Notice that, for both types of queries, the missing element of msk plays no role. Similarly, the last $q_2 - k$ non-matching queries of the second stage are distributed as the output $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k)$ and $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k - 1)$ in which the missing element of msk is set equal to $t_{\mathrm{j},b_{\mathrm{j}}} \equiv 1/\beta \pmod{p_1}$. Let us finally look at the answer to the k-th query. If $T = A_1^{\alpha\beta} \cdot D_4$ then the answer of the query is distributed according to the output of $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k - 1)$. Notice that if B does not abort the missing element of msk plays no role. On the other hand, if T is random $\mathbb{G}_{p_1 p_4}$ then the answer of the query is distributed according $\mathsf{Sim.KeyGen}(msk, \cdot, \mathsf{HVE}(\vec{x}, \cdot), \cdot, k)$. □

6.2.8 The fourth step of the proof

In this section we prove that H_3 is indistinguishable from $\mathsf{IdealExp}$.

Lemma 6.2.13 If Assumption 1 holds then, for all PPT adversaries $\mathcal{A}$, $H_3^{\mathcal{A}} \approx_c \mathsf{IdealExp}^{\mathcal{A}}$.

We start by defining $\ell + 1$ intermediate hybrid experiments $I_1, \dots, I_{\ell+1}$ such that $I_1 = H_3$ and $I_{\ell+1} = \mathsf{IdealExp}$ and show that, for $f = 1, \dots, \ell + 1$, I_f and I_{f+1} are indistinguishable, under Assumption 1.

To define I_f, we introduce a parametrized version of algorithm $\mathsf{Sim.Enc}$ that, with a slight abuse of notation, we also call $\mathsf{Sim.Enc}$.

The parametrized version of $\mathsf{Sim.Enc}$ takes as input public key $mpk' = [N, g_3, g_5, (T'_{i,b})_{i \in [\ell], b \in \{0,1\}}]$, the challenge ciphertext $\vec{x}$, the sequence $(\vec{y}_k, z_k)_{k=1}^{q_1}$ of the q_1 queries asked by the adversary in the first stage along with $z_k = \mathsf{HVE}(\vec{y}_k, \vec{x})$. In addition we let $\mathsf{Sim.Enc}$ take parameter $0 \leq f \leq \ell$. $\mathsf{Sim.Enc}$ returns a simulated ciphertext in which the $\mathbb{G}_{p_1}$ part of positions $i \leq f$ that do not belong to MPos is random. The remaining positions are well formed.

More formally, $\mathsf{Sim.Enc}$ chooses random $s \in \mathbb{Z}_N$, and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$. Then for each $i \in [\ell]$, $\mathsf{Sim.Enc}$ distinguishes the following cases.

- if $i < f$ and $i \notin \mathsf{MPos}$, $\mathsf{Sim.Enc}$ randomly selects $r_i \in \mathbb{Z}_N$ and sets
$$X_i = T'^{r_i}_{i,0} \cdot g_5^{s_i} \cdot Z_i.$$
- if $i < f$ and $i \in \mathsf{MPos}$, $\mathsf{Sim.Enc}$ sets
$$X_i = T'^{s}_{i,x_i} \cdot g_5^{s_i} \cdot Z_i.$$
- if $i \geq f$ $\mathsf{Sim.Enc}$ sets
$$X_i = T'^{s}_{i,x_i} \cdot g_5^{s_i} \cdot Z_i.$$

$\mathsf{Sim.Enc}$ returns the simulated ciphertext $\mathsf{Ct} = (X_i)_{i\in[\ell]}$ and stores the vector $(s_i)_{i\in[\ell]}$ in the state. Notice that if $f = \ell + 1$ the input $\vec{x}$ is not used by the algorithm and we obtain algorithm $\mathsf{Sim.Enc}$. On the other hand, if $f = 1$, we obtain algorithm $\mathsf{Sim.A.Enc}$.

Next, we define, for $1 \leq f \leq \ell + 1$, experiment $I_f^{\mathcal{A}}$ as follows.

$I_f^{\mathcal{A}}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{KeyGen}(msk,\cdot)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(mpk', \vec{x}, (\vec{y}_k, \mathsf{HVE}(\vec{y}_k, \vec{x}))_{k=1}^{q_1}, f)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
Output: $(mpk, \vec{x}, \alpha)$

Clearly, for all PPT adversaries $\mathcal{A}$, $I_1^{\mathcal{A}} = H_3$ and $I_{\ell+1}^{\mathcal{A}} = \mathsf{IdealExp}^{\mathcal{A}}$. Therefore to prove Lemma 6.2.13, it is enough to prove the following lemma.

Lemma 6.2.14 If Assumption 1 holds, then for all PPT adversaries $\mathcal{A}$, and for $f = 1, \ldots, \ell$, $I_f^{\mathcal{A}} \approx_c I_{f+1}^{\mathcal{A}}$.

To prove the above lemma, we introduce another sequence of intermediate games and make the following observation.

Observation 6.2.15 *If $f \in \mathsf{MPos}$, the output distributions of $\mathsf{Sim.Enc}(mpk, \vec{x}, (\vec{y}_k, \mathit{HVE}(\vec{y}_k, \vec{x})_{k=1}^{q},$ $f)$ and $\mathsf{Sim.Enc}(mpk, \vec{x}, (\vec{y}_k, \mathit{HVE}(\vec{y}_k, \vec{x})_{k=1}^{q}, f + 1)$ coincide.*

Therefore if $I_f^{\mathcal{A}}$ and $I_{f+1}^{\mathcal{A}}$ are distinguishable then it must be the case that $\mathcal{A}$ has a non-negligible probability of outputting a challenge plaintext $\vec{x}$ such that $f \notin \mathsf{MPos}$. For an adversary $\mathcal{A}$ that makes q_1 first stage token queries we introduce $2\cdot(q+1)$ intermediate hybrid experiments $L_{f,0}^{\mathcal{A}}, \ldots, L_{f,q_1}^{\mathcal{A}}$ and $M_{f,q_1}^{\mathcal{A}}, \ldots, M_{f,0}^{\mathcal{A}}$ which differ in the way in which first stage token queries are answered. Specifically, first stage queries are answered by running the following algorithm.

Algorithm $\mathsf{Sim.C.KeyGen}$ takes as input the master secret key msk, the query $\vec{y}$ for which a token has to be computed, the number $1 \leq i \leq q_1$ of the query, integer $1 \leq f \leq \ell+1$ and integer $0 \leq k \leq q_1$. The algorithm distinguishes the following cases.

- $i \leq k$ and $y_f \neq \star$:
  The $\mathbb{G}_{p_1}$ part of the token is random.
  For each $j \in S_{\vec{y}}$, the algorithm chooses random $W_j \in \mathbb{G}_{p_4}$, random $c_j \in \mathbb{Z}_N$ and random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j \in S_y} a_j = 0$ and sets
  $$Y_j = g_1^{c_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j.$$
- $i \leq k$ and $y_f = \star$: the algorithm returns the output of $\mathsf{KeyGen}(msk, \vec{y})$.
- $i > k$: the algorithm returns the output of $\mathsf{KeyGen}(msk, \vec{y})$.

We are now ready to describe experiments $L^{\mathcal{A}}_{f,k}$ and $M^{\mathcal{A}}_{f,k}$ for $f = 1, \ldots, \ell + 1$ and $k = 0, \ldots, q_1$.

```
$L^{\mathcal{A}}_{f,k}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{Sim.C.KeyGen}(msk,\cdot,\cdot,f,k)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(mpk', \vec{x}, (\vec{y}_k, \mathsf{HVE}(\vec{y}_k, \vec{x}))_{k=1}^q, f - 1)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, m, \alpha)$
```

```
$M^{\mathcal{A}}_{f,k}(1^\lambda, 1^\ell)$
$(mpk, mpk', msk) \leftarrow \mathsf{Sim.Setup}(1^\lambda, 1^\ell)$;
$(\vec{x}, \mathsf{aux}) \leftarrow \mathcal{A}_0^{\mathsf{Sim.C.KeyGen}(msk,\cdot,\cdot,f,k)}(mpk)$;
$\mathsf{Ct} \leftarrow \mathsf{Sim.Enc}(mpk', \vec{x}, (\vec{y}_k, \mathsf{HVE}(\vec{y}_k, \vec{x}))_{k=1}^q, f)$;
$\alpha \leftarrow \mathcal{A}_1^{\mathsf{Sim.KeyGen}(msk,\cdot,\mathsf{HVE}(\vec{x},\cdot))}(mpk, \mathsf{Ct}, \mathsf{aux})$;
**Output:** $(mpk, m, \alpha)$
```

Before continuing, we observe the following:

Observation 6.2.16 *For all PPT adversaries $\mathcal{A}$, $I^{\mathcal{A}}_f = L_{f,0}$.*
Directly from the definition of the experiments.

Observation 6.2.17 *For all PPT adversaries $\mathcal{A}$, $L^{\mathcal{A}}_{f,q_1} = M^{\mathcal{A}}_{f,q_1}$ for $f = 1, \ldots, \ell$, where q_1 is the number of first stage queries made by $\mathcal{A}$.*
From the definitions of the two experiments, it is clear that all the token queries are answered in the same way in both the experiments and all components X_i for $i \neq f$ of the challenge ciphertext are computed in the same way. Let us now look at X_f and more precisely to its $\mathbb{G}_{p_1}$ part. In L_{f,q_1}, the $\mathbb{G}_{p_1}$ part of X_f is computed as T'^{s}_{f,x_f} which is exactly how it is computed in M_{f,q_1} when $f \in \mathsf{MPos}$. On the other hand, when $f \notin \mathsf{MPos}$, the $\mathbb{G}_{p_1}$ part of X_f is chosen at random. However, observe that exponents $t_{f,0} \mod p_1$ and $t_{f,1} \mod p_1$ have not appeared in the answers to key queries since every query has either a $\star$ in position f (in which case position f of the answer is empty) or a non-$\star$ value in position f (in which case the $\mathbb{G}_{p_1}$ part of the element in position f of the answer is random). Therefore, we can conclude that the $\mathbb{G}_{p_1}$ part of the component X_f of the answer to the challenge query is also random in $\mathbb{G}_{p_1}$.

Observation 6.2.18 *For all PPT adversaries $\mathcal{A}$ and for $f = 1, \ldots, \ell - 1$, $M^{\mathcal{A}}_{f,0} = L^{\mathcal{A}}_{f+1,0}$.*
Indeed, in both experiments all key queries are answered correctly, and the challenge query in $M_{f,0}$ is by definition answered in the same way as in $L_{f+1,0}$.

By the above observations, for proving Lemma 6.2.14 it suffices to prove that $L_{f,k-1}$ and $L_{f,k}$ are indistinguishable and that $M_{f,k-1}$ and $M_{f,k}$ are indistinguishable, for $k = 1, \ldots, q_1$. In the next section we prove that, under Assumption 1, $L_{f,k-1}$ and $L_{f,k}$ are indistinguishable. The proof that $M_{f,k-1}$ and $M_{f,k}$ are indistinguishable is similar and omitted.

Indistinguishability of $L_{f,k}$ and $L_{f,k-1}$

We start by describing an algorithm B that takes as input f and k and an instance of Assumption 1 and interacts with an adversary $\mathcal{A}$. Then, provided that $\mathcal{A}$ outputs a challenge such that $f \notin \mathsf{MPos}$, B simulates with some non-negligible probability $L_{f,k}$ or $L_{f,k-1}$ depending on the nature of the challenge. This suffices to prove that the two hybrids are indistinguishable.

Description of algorithm B

Input: Integers $1 \le f \le \ell+1$ and $0 \le k \le q$, and a randomly chosen instance of Assumption 1 consisting of $D = (N, A_1, A_2, A_3, A_4, A_5, A_1^{\alpha}B_4, A_1^{\beta}C_4)$ and challenge T which is either $T = A_1^{\alpha\beta}D_4$ or random $\mathbb{G}_{p_1p_4}$.

Setup: B starts by constructing public parameters mpk. B sets $g_1 = A_1, g_2 = A_2, g_{12} = A_1 \cdot A_2$, $g_3 = A_3, g_4 = A_4, g_5 = A_5$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, B chooses random $t_{i,b} \in \mathbb{Z}_N, R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then B sets

$$mpk = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}] ,$$

and starts the interaction with $\mathcal{A}$ on input mpk.

Now, B guesses a position $\mathsf{j} \in [\ell]$ and a bit $b_{\mathsf{j}} \in \{0,1\}$ and chooses the following randomness that will be used in the $\mathbb{G}_{p_1}$ subgroup. Specifically, B chooses, for each $i \in [\ell]$ and $i \neq \mathsf{j}$, and $b \in \{0,1\}$, random $t'_{i,j} \in \mathbb{Z}_N$. Moreover, B chooses random $t'_{\mathsf{j},c_{\mathsf{j}}} \in \mathbb{Z}_N$ where $c_{\mathsf{j}} = 1 - b_{\mathsf{j}}$. Notice that the value $t'_{\mathsf{j},b_{\mathsf{j}}}$ is unknown to B. It will be provided by the assumption as β at the exponent.

First Stage Secret Tokens: To simulate the output of $\mathsf{Sim.C.KeyGen}(\mathsf{Sk}, \vec{y}_i, f, k)$, B does the following way:

$i \le k$: We have the following mutually exclusive cases.

Case A.1: $y_f \neq \star$. In this case, B outputs a key whose $\mathbb{G}_{p_1}$ part is random.

Specifically, for each $j \in S_{\vec{y}_i}$, B chooses random a_j such that $\sum_{j\in S_{\vec{y}_j}} a_i = 0$, random $r_j \in \mathbb{Z}_N$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\vec{y}_i}$, B sets

$$Y_j = g_1^{r_j} \cdot g_2^{a_j/v_{i,y_j}} \cdot W_j .$$

Case A.2: $y_f = \star$. In this case, B outputs a key with a well-formed $\mathbb{G}_{p_1}$ part.

Specifically, B, for each $j \in S_{\vec{y}_i}$, chooses random $W_j \in \mathbb{G}_{p_4}$, random $a_j \in \mathbb{Z}_N$ under the constraint that $\sum_{j\in S_{\vec{y}_i}} a_j = 0$ and then sets

$$Y_j = g_1^{a_j/t'_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j .$$

In the special case that $y_{\mathsf{j}} = b_{\mathsf{j}}$, then B sets

$$Y_{\mathsf{j}} = (A_1^{\beta} \cdot C_4)^{a_i} \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}} .$$

$i = k$: We distinguish between the following cases:

Case B.1: $y_f = \star$. B performs the same steps of Case A.2.

Case B.2: $y_f \neq \star$ **and** $y_{\mathsf{j}} \neq b_{\mathsf{j}}$. In this case, B aborts.

Case B.3: $y_f \neq \star$ **and** $y_{\mathsf{j}} = b_{\mathsf{j}}$. B mounts T, the challenge of the assumption, in position j.

Specifically, let $S = S_{\vec{y}_i} \setminus \{\mathsf{j}, h\}$, where h is an index such that $y_{k,h} \neq \star$. Such an index h always exists since we assumed that each query contains at least two non-$\star$ entries. Then, for each $j \in S$, B sets

$$Y_j = g_{12}^{a_j/t'_{j,y_j}} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j \; .$$

Then, for position j, B sets

$$Y_{\mathsf{j}} = T \cdot g_2^{a_{\mathsf{j}}/t_{\mathsf{j},y_{\mathsf{j}}}} \cdot W_{\mathsf{j}} \; ,$$

and for position h, B sets

$$Y_h = (A_1^{\alpha} B_4)^{-1/t'_{h,y_h}} \cdot g_1^{-s/t'_{h,y_h}} \cdot g_2^{-(s+a_j)/t_{h,y_h}} \cdot W_h,$$

where $s = \sum_{j \in S} a_j$.

$i > k$: B handles these queries as in Case A.2, independently from whether $y_f = \star$ or $y_f \neq \star$.

Ciphertext: B receives $\vec{x}$ and has to compute a ciphertext. We distinguishes the following two cases:

Case C.1: $x_{\mathsf{j}} = b_{\mathsf{j}}$. In this case, B aborts.

Case C.2: $x_{\mathsf{j}} \neq b_{\mathsf{j}}$. In this case, B computes sequence $(\vec{y}_i, \mathsf{HVE}(\vec{y}_i, \vec{x}))_{i=1}^{q_1}$ for all queries $\vec{y}_i$ that it has received from $\mathcal{A}$ to induce the set MPos.

Then, B chooses random $s \in \mathbb{Z}_N$, and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$ and random $s_i \in \mathbb{Z}_N$. Then, for each $i \in [\ell]$, if $i < f$ and $i \notin \mathsf{MPos}$, the algorithm randomly select $r_i \in \mathbb{Z}_N$ and sets

$$X_i = g_1^{r_i} \cdot g_5^{s_i} \cdot Z_i \; ,$$

otherwise, the algorithm sets

$$X_i = g_1^{s \cdot t'_{i,x_i}} \cdot g_5^{s_i} \cdot Z_i \; .$$

Second Stage Secret Tokens: To simulate the output of $\mathsf{Sim.KeyGen}(msk, \vec{y}_i, \mathsf{HVE}(\vec{x}, \vec{y}_i))$, B does the following way:

$\mathsf{HVE}(\vec{x}, \vec{y}_i) = 0$: B generates a secret key with a random $\mathbb{G}_{p_1}$ part and without the $\mathbb{G}_{p_5}$ part.

Specifically, for each $j \in S_{\vec{y}_i}$, B chooses random a_j such that $\sum_{j \in S_{\vec{y}_j}} a_i = 0$, random $r_j \in \mathbb{Z}_N$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\vec{y}_i}$, B sets

$$Y_j = g_1^{r_j} \cdot g_2^{a_j/t_{i,y_j}} \cdot W_j \; .$$

$\mathsf{HVE}(\vec{x}, \vec{y}_i) = 1$: B generates a secret key without the $\mathbb{G}_{p_1}$ part and with a $\mathbb{G}_{p_5}$ part correlated with that of the ciphertext.

Specifically, for each $j \in S_{\vec{y}_i}$, B chooses random a_j such that $\sum_{j \in S_{\vec{y}_j}} a_i = 0$, and random $W_j \in \mathbb{G}_{p_4}$. Then, for each $j \in S_{\vec{y}_i}$, B sets

$$Y_j = g_2^{a_j/t_{i,y_j}} \cdot g_5^{a_j/s_j} \cdot W_j \; .$$

This ends the description of B.

The algorithm B will be used to prove properties of experiments L. We can modify B so that, on input f and k, the challenge ciphertext is constructed by randomizing the $\mathbb{G}_{p_1}$ part also of the f-th component. The so modified algorithm, that we call B_2, closely simulates the work of experiments M and will be used to prove properties of these experiments.

Let us define the following two events.

$\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$: denotes the event that B does not abort while computing the answer to the k-th query in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that $y_{k,f} = \star$ or $y_{k,\mathsf{j}} = b_{\mathsf{j}}$.

$\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{B}}(f,k)$: denotes the event that B does not abort while computing the ciphertext in an interaction with $\mathcal{A}$ on input f and k. This is equivalent to the event that adversary $\mathcal{A}$ outputs vector $\vec{x}$ such that $x_{\mathsf{j}} = c_{\mathsf{j}} = 1 - b_{\mathsf{j}}$.

We can modify, experiments PKGame, $L(f,k)$ and $M(f,k)$ so that j and b_{j} are chosen just like B does. This modification makes the definitions of events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathbf{Exp}}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathbf{Exp}}$ meaningful also for these experiments. We write $\mathsf{NotAbort}^{\mathcal{A}}_{2}$ as a shorthand for $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{PKGame}}$.

Lemma 6.2.19 For all f, k and $\mathcal{A}$, $\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)] \geq \frac{1}{\ell}$.

PROOF. The probability of $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$ is at least the probability that $y_{k,\mathsf{j}} = b_{\mathsf{j}}$. Moreover, the view of $\mathcal{A}$ up to the k-th key query is independent from b_{j} and j. Now observe that the $\vec{y}_k$ has at least two non-star entry and, provided that j is one of these (which happens with probability at least $2/\ell$), the probability that $y_{k,\mathsf{j}} = b_{\mathsf{j}}$ is $1/2$. □

Lemma 6.2.20 For all f, k and $\mathcal{A}$, $\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathbf{Exp}}(f,k)] \geq \frac{1}{2\ell}$ for $\mathbf{Exp} = L(f,k)$.

PROOF. $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathbf{Exp}}(f,k)$ is the event that $y_{\vec{k},j} \neq x_{\mathsf{j}}$ in the experiment $\mathbf{Exp}$. It is easy to see that the probability that j and b_{j} are correctly guessed such that $x_{\mathsf{j}} = c_{\mathsf{j}} = 1 - b_{\mathsf{j}}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. □

Lemma 6.2.21 Suppose event $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$ occurs. If $T = T_1$ then $\mathcal{A}$'s view up to the challenge ciphertext in the interaction with B running on input (f,k) is the same as in $L_{f,k-1}$. If instead $T = T_2$ then $\mathcal{A}$'s view up to the challenge ciphertext in the interaction with B running on input (f,k) is the same as in $L_{f,k}$.

Moreover, suppose events $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{B}}(f,k)$ occur. If $T = T_1$ then $\mathcal{A}$'s total view in the interaction with B running on input (f,k) is the same as in $L_{f,k-1}$. If instead $T = T_2$ then $\mathcal{A}$'s total view in the interaction with B running on input (f,k) is the same as in $L_{f,k}$.

PROOF. First observe that mpk has the same distribution as the public parameters seen by $\mathcal{A}$ in both experiments. The same holds for the answers to the first $(k-1)$ key queries and to the last $(q_1 - k)$. Let us now focus on the answer to the k-th key query. We have two cases:

Case 1: $y_f = \star$**.** Then the view of $\mathcal{A}$ in the interaction with B is independent from T (see Case B.1) and, on the other hand, by definition, the two experiments coincide. Therefore the lemma holds in this case.

Case 2: $y_f \neq \star$**.** Suppose $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ and that $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$ occurs. Therefore, $y_{\mathrm{j}} = b_{\mathrm{j}}$ and B's answer to the k-th key query has the same distributions as in $L(f, k-1)$.

On the other hand if T is random in $\mathbb{G}_{p_1p_4}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k)$ occurs, the $\mathbb{G}_{p_1}$ parts of the Y_j's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in $L_{f,k}$.

For the second part of the lemma, we observe that the challenge ciphertext has the same distribution in both experiments and that, if $\mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{B}}(f,k)$ occurs, B properly constructs the challenge ciphertext. □

Let us now analyze the probability that $\mathcal{A}$ does output a challenge such that $f \notin \mathsf{MPos}$, this is crucial for B to successfully simulate with some non-negligible probability $L(f,k)$ or $L(f,k-1)$.

We start by introducing some notation.

$E^{\mathcal{A}}_{f,\mathbf{Exp}}$ is defined as the event that in experiment **Exp** the adversary $\mathcal{A}$ declare a challenge vector such that $f \notin \mathsf{MPos}$. When the adversary $\mathcal{A}$ is clear from the context we will simply write $E_{f,\mathbf{Exp}}$.

$E^{\mathcal{A}}_{f}$: is defined as the event that in experiment PKGame, the adversary $\mathcal{A}$ declares a challenge vector such that $f \notin \mathsf{MPos}$. When the adversary $\mathcal{A}$ is clear from the context we will simply write E_f.

$E^{\mathcal{A}}_{f,\mathsf{B}}(f',k)$: we extend the definition of $E_{f,\mathbf{Exp}}$ to include the experiment played by $\mathcal{A}$ against the algorithm B. Thus we denote by the event that in the interaction between $\mathcal{A}$ and B on input f' and k, B does not abort and $\mathcal{A}$ declares a challenge vector such that $f \notin \mathsf{MPos}$. If $\mathcal{A}$, f' and k are clear from the context, we will simply write $E_{f,\mathsf{B}}$.

$E^{\mathcal{A}}_{f,f'}$: is defined as the event that during the execution of $I_{f'}$ adversary $\mathcal{A}$ outputs a challenge vector such that $f \notin \mathsf{MPos}$.

Observation 6.2.22 *For all PPT adversaries $\mathcal{A}$ and distinguisher $\mathcal{D}$ and all $1 \leq f \leq \ell$, we have that* $\mathrm{Prob}[\mathcal{D}(I(f)^{\mathcal{A}}) = 1 | \neg E_{f,f}] = \mathrm{Prob}[\mathcal{D}(I(f+1)^{\mathcal{A}}) = 1 | \neg E_{f,f+1}]$.

PROOF. By definition of I, if the challenge vector is such that $f \in \mathsf{MPos}$, then $\mathcal{A}$'s view in I_f and I_{f+1} is the same. □

Observation 6.2.23 *For all PPT adversaries $\mathcal{A}$ and all $1 \leq f \leq \ell$, we have that* $\mathrm{Prob}[E^{\mathcal{A}}_{f,f}] = \mathrm{Prob}[E^{\mathcal{A}}_{f,f+1}]$.

PROOF. The view of $\mathcal{A}$ in I_f up to the challenge ciphertext is independent from f. □

Therefore we can set $\mathrm{Prob}[E^{\mathcal{A}}_{f,f}] = \mathrm{Prob}[E^{\mathcal{A}}_{f,1}] = \mathrm{Prob}[E^{\mathcal{A}}_{f}]$.

Lemma 6.2.24 If Assumption 1 holds, then for $k = 1, \ldots, q$ and $f = 1, \ldots, \ell$, and for all PPT adversaries $\mathcal{A}$, $\left|\mathrm{Prob}[E^{\mathcal{A}}_{f,G}] - \mathrm{Prob}[E^{\mathcal{A}}_{f,H}]\right|$ and $\left|\mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,G}] - \mathrm{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}]\right|$ are negligible functions of λ, for experiments $G = L(f, k-1)$ and $H = L(f,k)$.

PROOF. We prove the lemma for $E_{f,G}$ and $E_{f,H}$. A similar reasoning holds for $\mathsf{NotAbort}^{\mathcal{A}}_{2,G}$ and $\mathsf{NotAbort}^{\mathcal{A}}_{2,H}$. For the sake of contradiction, suppose that $\mathrm{Prob}[E^{\mathcal{A}}_{f,G}] \geq \mathrm{Prob}[E^{\mathcal{A}}_{f,H}] + \epsilon$ for some non-negligible ϵ. Then we can modify algorithm B into algorithm $\mathcal{B}$ with a non-negligible advantage in breaking Assumption 1. Algorithm $\mathcal{B}$ simply execute B's code. By

Lemma 6.2.19 event $\mathsf{NotAbort}_{1,\mathsf{B}}$ occurs with probability at least $1/\ell$ and in this case $\mathcal{B}$ can continue the execution of B's code and receive the challenge vector from $\mathcal{A}$. At this point, $\mathcal{B}$ checks whether $f \notin \mathsf{MPos}$. If it is the case, $\mathcal{B}$ outputs 1; else $\mathcal{B}$ outputs 0. It is easy to see that, by Lemma 6.2.21, the above algorithm has a non-negligible advantage in breaking Assumption 1. □

The proof of the following corollary is straightforward from Lemma 6.2.24 and Observations 6.2.17-6.2.18.

Corollary 6.2.25 For all $f = 1, \dots, \ell+1$ and $k = 0, \dots, q$, and all PPT adversaries $\mathcal{A}$, we have that, for $H = L_{f,k}$ $\left|\text{Prob}[E^{\mathcal{A}}_{f,H}] - \text{Prob}[E^{\mathcal{A}}_{f}]\right|$ and $\left|\text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2,H}] - \text{Prob}[\mathsf{NotAbort}^{\mathcal{A}}_{2}]\right|$ are negligible.

We are now ready to prove that $L_{f,k-1} \approx_c L_{f,k}$. To do this let us define the event $\mathsf{Succ}^{\mathcal{A}}(f,k)$ as

$$\mathsf{Succ}^{\mathcal{A}}(f,k) := \mathsf{NotAbort}^{\mathcal{A}}_{1,\mathsf{B}}(f,k) \wedge \mathsf{NotAbort}^{\mathcal{A}}_{2,\mathsf{B}}(f,k) \wedge E^{\mathcal{A}}_{f,\mathsf{B}}(f,k). \tag{6.1}$$

When $\mathcal{A}$ is clear from the context we use the shortcut $\mathsf{Succ}(f,k)$.

We are now ready to prove Lemma 6.2.26.

Lemma 6.2.26 Suppose there exists an adversary $\mathcal{A}$, a distinguisher $\mathcal{D}$ and integers $1 \le f \le \ell$ and $1 \le k \le q$ such that $\left|\text{Prob}[\mathcal{D}(G^{\mathcal{A}}) = 1] - \text{Prob}[\mathcal{D}(H^{\mathcal{A}}) = 1]\right| \ge \epsilon$, where $G = L(f,k-1)$, $H = L(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathbf{Adv}^{\mathcal{B}}_{1} \ge \text{Prob}[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.

PROOF. Assume without loss of generality that $\text{Prob}[\mathcal{D}(G^{\mathcal{A}}) = 1] \ge \text{Prob}[\mathcal{D}(H^{\mathcal{A}}) = 1] + \epsilon$ and consider the following algorithm $\mathcal{B}$. $\mathcal{B}$ uses algorithm B as a subroutine and interacts with $\mathcal{A}$ on input integers f and k for which the above inequality holds, and an instance (D,T) of Assumption 1. If event $\mathsf{Succ}(f,k)$ does not occur, $\mathcal{B}$ outputs $\perp$. Otherwise, $\mathcal{B}$ outputs $\mathcal{D}$'s output. Therefore we have

$$\begin{aligned}\text{Prob}[\mathcal{B}\text{ outputs }1|T = T_1] &= \text{Prob}[\mathcal{B}\text{ outputs }1|T = T_1 \wedge \mathsf{Succ}(f,k)] \cdot \\ &\quad \text{Prob}[\mathsf{Succ}(f,k)|T = T_1]\end{aligned} \tag{6.2}$$

By definition of $\mathsf{Succ}(f,k)$ we have

$$\begin{aligned}\text{Prob}[\mathsf{Succ}(f,k)|T = T_1] &= \text{Prob}[E_{f,\mathsf{B}} \wedge \mathsf{NotAbort}_{1,\mathsf{B}} \wedge \mathsf{NotAbort}_{2,\mathsf{B}}|T = T_1] \\ &= \text{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}|T = T_1]\cdot \\ &\qquad \text{Prob}[E_{f,\mathsf{B}} \wedge \mathsf{NotAbort}_{2,\mathsf{B}}|\mathsf{NotAbort}_{1,\mathsf{B}} \wedge T = T_1].\end{aligned}$$

Now observe that event $\mathsf{NotAbort}_{1,\mathsf{B}}$ is determined before B uses T and thus

$$\text{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}|T = T_1] = \text{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}].$$

Moreover, by Lemma 6.2.21, if event $\mathsf{NotAbort}_{1,\mathsf{B}}$ occurs and $T = T_1$, the view of $\mathcal{A}$ up to Challenge Query is equal to the view of $\mathcal{A}$ in experiment G and thus

$$\text{Prob}[E_{f,\mathsf{B}} \wedge \mathsf{NotAbort}_{2,\mathsf{B}}|\mathsf{NotAbort}_{1,\mathsf{B}} \wedge T = T_1] = \text{Prob}[E_{f,G} \wedge \mathsf{NotAbort}_{2,G}]$$

whence

$$\begin{aligned}\text{Prob}[\mathsf{Succ}^{\mathcal{A}}(f,k)|T = T_1] &= \text{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G} \wedge E_{f,G}] \\ &= \text{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}] \cdot \text{Prob}[\mathsf{NotAbort}_{2,G}] \cdot \text{Prob}[E_{f,G}]\end{aligned},$$

where $\mathsf{NotAbort}_{2,G}$ and $E_{f,G}$ are independent. Finally, if $T = T_1$ and $\mathsf{Succ}^{\mathcal{A}}(f,k)$ occurs, then, by Lemma 6.2.21, $\mathcal{A}$'s view is exactly as in experiment G, and thus the probability that $\mathcal{B}$ outputs 1 is equal to the probability that $\mathcal{D}$ output 1. We can thus rewrite Eq. 6.2 as

$$\begin{aligned}\mathrm{Prob}[\mathcal{B}\text{ outputs }1|T=T_1] = \mathrm{Prob}[\mathcal{D}(G^{\mathcal{A}})=1]\cdot \\ \mathrm{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}]\cdot\mathrm{Prob}[\mathsf{NotAbort}_{2,G}]\cdot\mathrm{Prob}[E_{f,G}]\end{aligned}$$

A similar reasoning yields

$$\begin{aligned}\mathrm{Prob}[\mathcal{B}\text{ outputs }1|T=T_2] = \mathrm{Prob}[\mathcal{D}(H^{\mathcal{A}})=1]\cdot \\ \mathrm{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}]\cdot\mathrm{Prob}[\mathsf{NotAbort}_{2,H}]\cdot\mathrm{Prob}[E_{f,H}]\end{aligned}$$

By using Corollary 6.2.25, Lemma 6.2.19 and Lemma 6.2.20, we can conclude that there exists a negligible function ν such that we have

$$\begin{aligned}\mathbf{Adv}_1^{\mathcal{B}} &= \mathrm{Prob}[\mathsf{NotAbort}_{1,\mathsf{B}}]\cdot\mathrm{Prob}[\mathsf{NotAbort}_2]\cdot\mathrm{Prob}[E_f]\cdot \\ &\quad \big(\mathrm{Prob}[\mathcal{D}(G^{\mathcal{A}})=1]-\mathrm{Prob}[\mathcal{D}(H^{\mathcal{A}})=1]\big)-\nu(\lambda) \\ &\geq \frac{\epsilon}{2\ell^2}\cdot\mathrm{Prob}[E_f]-\nu(\lambda)\end{aligned}$$

□

The following Lemma can be proved by referring to algorithm B_2. We omit further details since the proof is essentially the same as the one of Lemma 6.2.26.

Lemma 6.2.27 Suppose there exists an adversary $\mathcal{A}$, a distinguisher $\mathcal{D}$ and integers $1 \leq f \leq \ell+1$ and $1 \leq k \leq q$ such that $\big|\mathrm{Prob}[\mathcal{D}(G^{\mathcal{A}})=1]-\mathrm{Prob}[\mathcal{D}(H^{\mathcal{A}})=1]\big| \geq \epsilon$, where $G = M_{f,k-1}$, $H = M_{f,k}$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathbf{Adv}_1^{\mathcal{B}} \geq \mathrm{Prob}[E_f]\cdot\epsilon/(2\cdot\ell^2)-\nu(\lambda)$, for a negligible function ν.

We are finally ready to prove Lemma 6.2.14.

Lemma 6.2.14. If Assumption 1 holds, $I_f \approx_c I_{f+1}$.

PROOF. Suppose that for some adversary $\mathcal{A}$, distinguisher $\mathcal{D}$ and $f \in [\ell]$

$$\big|\mathrm{Prob}[\mathcal{D}(I_f^{\mathcal{A}})=1]-\mathrm{Prob}[\mathcal{D}(I_{f+1}^{\mathcal{A}})=1]\big| \geq \epsilon\,. \tag{6.3}$$

Now recall that $I_f = L_{f,0}$ and $I_{f+1} = M_{f,0}$. Thus, there exists $1 \leq k \leq q$ such that:

$$\big|\mathrm{Prob}[\mathcal{D}(G^{\mathcal{A}})=1]-\mathrm{Prob}[\mathcal{D}(H^{\mathcal{A}})=1]\big| \geq \epsilon/(2q)\,,$$

where $G = L_{f,k}$ and $H = L_{f,k-1}$ or where $G = M_{f,k}$ and $H = M_{f,k-1}$. Then by Lemma 6.2.26, in the former case, and by Lemma 6.2.27 in the latter, we can construct an adversary $\mathcal{B}$ against Assumption 1, such that

$$\mathbf{Adv}_1^{\mathcal{B}} \geq \frac{\epsilon}{4q\ell^2}\cdot\mathrm{Prob}[E_f]-\nu(\lambda)$$

Now it remains to estimate $\mathrm{Prob}[E_f]$. Notice that we can write

$$\begin{aligned}\mathrm{Prob}[\mathcal{D}(I_f^{\mathcal{A}})=1] = \mathrm{Prob}[E_{f,f}]\cdot\mathrm{Prob}[\mathcal{D}(I_f^{\mathcal{A}})=1|E_{f,f}]+ \\ \mathrm{Prob}[\neg E_{f,f}]\cdot\mathrm{Prob}[\mathcal{D}(I_f^{\mathcal{A}})=1|\neg E_{f,f}]\,,\end{aligned}$$

and

$$\begin{aligned}\text{Prob}[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1] = \text{Prob}[E_{f,f+1}] \cdot \text{Prob}[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1|E_{f,f+1}] + \\ \text{Prob}[\neg E_{f,f+1}] \cdot \text{Prob}[\mathcal{D}(I^{\mathcal{A}}_{f}) = 1|\neg E_{f,f+1}]\end{aligned} ,$$

and by combining Equation 6.3 and Observations 6.2.22 and 6.2.23, we obtain

$$\text{Prob}[E_f] \cdot \left|\text{Prob}[\mathcal{D}(I^{\mathcal{A}}_{f}) = 1|E_{f,f}] - \text{Prob}[\mathcal{D}(I^{\mathcal{A}}_{f+1}) = 1|E_{f,f+1}]\right| \geq \epsilon.$$

Thus, we can conclude that

$$\text{Prob}[E_f] \geq \epsilon \, ,$$

and thus $\mathcal{B}$ has advantage

$$\mathbf{Adv}^{\mathcal{B}}_1 \geq \frac{\epsilon^2}{4q\ell^2} - \nu(\lambda) \, .$$

□

Chapter 7

Summary of Contributions

We briefly summarize our main contributions.

Generalized Key Delegation for Wildcarded Identity-Based and Inner-Product Encryption. Even though the WIBE and WKD-IBE constructions in [ACD$^+$06, AKN07] are very practical, they had two significant shortcomings. First, their security proofs only hold in cases where the maximum hierarchy depth L is a constant due to the fact that they are not tight and lose a factor which is exponential in L. As a result, these schemes can only be used in scenarios where such a restriction is acceptable. In particular, when using WKD-IBE schemes to build identity-based broadcast encryption schemes, such a limitation on the maximum hierarchy depth will have a direct impact on the maximum size of the target group. Second, their solutions do not hide the pattern associated with the ciphertext. Hence, their schemes cannot be used in any application where the anonymity of the recipient needs to be preserved.

In Chapter 4 we showed how to overcome the limitations of existing WKD-IBE and WIBE schemes to obtain the first fully secure anonymous WW-IBE scheme.

Lattice-based Hierarchical Inner Product Encryption. In Chapter 5, we considered the problem of constructing hierarchical inner-product encryption scheme based on lattices assumptions. To achieve this goal, we extended the lattice-based IPE scheme by Agrawal *et al.* [AFV11] to the hierarchical setting by employing basis delegation technics by Peikert *et al.* [CHKP10] and by Agrawal *et al.* [ABB10].

Adaptive Simulation-Based Secure Constructions for Functional Encryption. In Chapter 6, we considered the problem of designing simulation-based secure functional encryption schemes. We looked both at general functionalities corresponding to the class of poly-sized circuits and at a more specialized functionality, specifically HVE.

Bibliography

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, May 2010. (Cited on pages 4, 17, 18, 19, 20, 45, 47, 52, 55 and 87.)

[ABC$^+$08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, July 2008. (Cited on page 2.)

[ACD$^+$06] Michel Abdalla, Dario Catalano, Alex Dent, John Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311. Springer, July 2006. (Cited on pages 3, 21, 22, 23, 26 and 87.)

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, December 2011. (Cited on pages 3, 4, 17, 45, 46, 52, 53 and 87.)

[AGVW12] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. *IACR Cryptology ePrint Archive*, 2012:468, 2012. (Cited on pages 3, 4, 10, 59 and 60.)

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 99–108. ACM Press, May 1996. (Cited on page 19.)

[AKN07] Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. In Joachim Biskup and Javier López, editors, *ESORICS 2007: 12th European Symposium on Research in Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 139–154. Springer, September 2007. (Cited on pages 4, 21, 22, 23 and 87.)

[AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *In STACS (2009)*, 2009. (Cited on pages 17 and 19.)

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004. (Cited on pages 2, 9, 21 and 22.)

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, May 2005. (Cited on pages 2 and 22.)

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004. (Cited on page 38.)

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, May 2004. (Cited on page 2.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001. (Cited on page 14.)

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. (Cited on page 2.)

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, February 2005. (Cited on pages 14, 21 and 26.)

[BL95] Dan Boneh and Richard J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 424–437. Springer, August 1995. (Cited on page 17.)

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, December 2001. (Cited on page 14.)

[BO12] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. Cryptology ePrint Archive, Report 2012/515, 2012. `http://eprint.iacr.org/`. (Cited on pages 3, 4 and 59.)

[Bon07] Dan Boneh. Bilinear groups of composite order (invited talk). In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007: 1st International Conference on Pairing-based Cryptography*, volume 4575 of *Lecture Notes in Computer Science*, page 1. Springer, July 2007. (Cited on page 14.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. (Cited on page 2.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, March 2011. (Cited on pages 3, 4, 5, 8, 10 and 59.)

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–

307. Springer, August 2006. (Cited on page 2.)

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, February 2007. (Cited on pages 2, 7 and 59.)

[BWY11] Mihir Bellare, Brent Waters, and Scott Yilek. Identity-based encryption secure against selective opening attack. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 235–252. Springer, March 2011. (Cited on page 15.)

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, May 2003. (Cited on pages 2 and 9.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552. Springer, May 2010. (Cited on pages 4, 17, 19, 45, 47 and 87.)

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, December 2001. (Cited on page 2.)

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, May 2004. (Cited on page 18.)

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 308–317. IEEE Computer Society, 1990. (Cited on pages 4, 59 and 60.)

[FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999. (Cited on page 14.)

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62(206):865–865, 1994. (Cited on page 14.)

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on pages 3 and 8.)

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 2.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008. (Cited on page 17.)

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, December 2002. (Cited on page 2.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology – CRYPTO 2012*, Lecture Notes in Computer Science, pages 162–179. Springer, August 2012. (Cited on pages 3, 59, 60 and 63.)

[Hes03] Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, August 2003. (Cited on page 14.)

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, April / May 2002. (Cited on pages 2, 7 and 14.)

[KH04] Kaoru Kurosawa and Swee-Huay Heng. From digital signature to ID-based identification/signature. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 248–261. Springer, March 2004. (Cited on page 14.)

[KH06] Kaoru Kurosawa and Swee-Huay Heng. The power of identification schemes. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 364–377. Springer, April 2006. (Cited on page 14.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, April 2008. (Cited on pages 2, 8, 22 and 33.)

[LOS$^+$10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, May 2010. (Cited on pages 8, 22, 33, 34, 37, 38 and 41.)

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, February 2010. (Cited on pages 21, 22, 26, 28 and 40.)

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *Advances in Cryptology – CRYPTO 2012*, Lecture Notes in Computer Science, pages 180–198. Springer, August 2012. (Cited on page 22.)

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002. (Cited on page 19.)

[MOV91] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *23rd Annual ACM Symposium on Theory of Computing*, pages 80–89. ACM Press, May 1991. (Cited on page 14.)

[MR04] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, October 2004. (Cited on page 17.)

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, August 2002. (Cited on page 3.)

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/`. (Cited on pages 3, 8, 10, 59 and 63.)

[OT08] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 57–74. Springer, September 2008. (Cited on page 16.)

[OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 214–231. Springer, December 2009. (Cited on pages 2, 8, 16, 22 and 34.)

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, August 2010. (Cited on page 38.)

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 333–342. ACM Press, May / June 2009. (Cited on page 20.)

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM Press, May 2005. (Cited on pages 3, 17 and 20.)

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, August 1985. (Cited on page 1.)

[Sho96] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26:1484–1509, 1996. (Cited on page 17.)

[Sho08] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2008. (Cited on page 56.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 463–472. ACM Press, October 2010. (Cited on pages 3 and 63.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald

Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, May 2005. (Cited on page 2.)

[SW08] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578. Springer, July 2008. (Cited on pages 24 and 34.)

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, August 2009. (Cited on pages 21, 22 and 26.)