Predicate Encryption Systems
No Query Left Unanswered

by

Vincenzo Iovino
iovino@dia.unisa.it
Dipartimento di Informatica,
Università di Salerno, 84084 Fisciano (SA), Italia.

A thesis
presented to the Università di Salerno
in fullfillment of the
thesis requirements for the degree of
Doctor of Philosophy
in
Computer Science

Supervisor: Giuseppe Persiano

Graduate Group Chair: Margherita Napoli

Fisciano (SA), Italia, 2011

April 2011

I hereby declare that I am the sole author of this thesis. Different versions of this thesis distributed by the author could contain minor revisions or corrections.

I agree that my thesis can be made avaiable to the public.

Vincenzo Iovino, April 2011

Abstract

Predicate encryption is an important cryptographic primitive (see [7, 14, 28]) that enables fine-grained control on the decryption keys. Let Π be a class of binary predicates. Roughly speaking, in a predicate encryption scheme for Π the owner of the master secret key Msk can derive secret key Sk_P, for any predicate $P \in \Pi$. In encrypting a message M, the sender can specify an *attribute* $\vec{x}$ and the resulting ciphertext $\tilde{X}$ can be decrypted only by using keys $\mathsf{Sk}_{\vec{y}}$ such that $P(\vec{x}) = 1$. Our main contribution is the *first* construction of a predicate encryption scheme that can be proved *fully* secure against *unrestricted* queries by probabilistic polynomial-time adversaries under non-interactive constant sized (that is, independent of the length ℓ of the attribute vectors) hardness assumptions on bilinear groups.

Specifically, we consider *Hidden Vector Encryption* (HVE for short), a notable case of predicate encryption introduced by Boneh and Waters [14]. In a HVE scheme, the ciphertext attributes are vectors $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ of length ℓ over alphabet Σ, keys are associated with vectors $\vec{y} = \langle y_1, \ldots, y_\ell \rangle$ of length ℓ over alphabet $\Sigma \cup \{\star\}$ and we consider the $\mathsf{Match}(\vec{x}, \vec{y})$ predicate which is true if and only if, for all i, $y_i \neq \star$ implies $x_i = y_i$. Previous constructions limited the proof of security to *restricted* adversaries that could ask only *non-matching* queries; that is, for challenge attribute vectors $\vec{x}_0$ and $\vec{x}_1$, the adversary could ask only keys for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 0$. Generally speaking, restricted adversaries can ask only queries that do not satisfy neither of the challenge attributes. At time of writing, the construction of schemes secure against unrestricted adversaries was an open problem, not just for HVE, but for any non-trivial[1] predicate encryption system and a candidate solution for HVE is presented in this thesis. Beyond that, we will also discuss other kinds of predicate encryption systems, their security notions and applications.

[1]For some specific cases of Predicate Encryption Systems like Anonymous IBE the adversary can ask only queries for predicates that do not satisfy neither of the challenges, so that for these systems the security against 'restricted' adversaries is the best we can guarantee.

Acknowledgements

I am grateful to my co-authors Carlo Blundo, Angelo De Caro and Giuseppe Persiano. I appreciated much to conduct research with them. I am grateful to my advisor Giuseppe Persiano for introducing me to the science and the art of the cryptography and for his deep patience in assisting me. I also thank Giuseppe Ateniese to allow me to visit the department of Computer Science of the University of Baltimore. There, I got the possiblity to learn new subjects of study and to have interesting and fruitful exchange of ideas with other researchers. Thanks to my family, friends (including my lovely cats!) and collegaues for the beautiful and amusing moments spent together during my life and Ph.D. time. They contributed to keep me happy and sunny even when the research was poor. The list would be too long but I especially mention my best friend Angelo Avino. I conclude with a tribute to the Magica Roma and especiallly to Francesco Totti. As well as for research, his deeds teach to not ever give up and that the King inside you does not ever die.

Contents

Chapter 1

Preface

Predicate encryption is a young area of research that generalizes traditional encryption. In classical encryption the decryption key allows to decrypt any ciphertext, that is, it implements a all-or-nothing policy. Predicate encryption (henceforth, PE) offers a mechanism to implement more sophisticated policies. In PE you can decrypt a ciphertext only if you own a special key which satisfies a relationship with such ciphertext. In PE systems, ciphertexts are associated to some attributes x, and the system can generate keys for some boolean predicates. The owner of a key for predicate P can decrypt a ciphertext associated with attribute x iff $P(x) = 1$. Consider the case of Identity-Based Enryption (IBE). In a IBE system, you can encrypt a message for an user id without knowing his/her public-key, but simply specifying his/her identity. The user id has a decryption key for identity id by means of which can decrypt ciphertexts associated with id, but another user with key for id' cannot leak any information from any ciphertexts associated with identities different from id'. This guarantees great flexibility since it avoids to use digital certificates. IBE can be viewed as a very special case of PE where the class of predicates that of equality. PE systems can be attribute-hiding or not, meaning that the ciphertexts hides the attributes or not. A very interesting case of PE system is the so called Hidden Vector Encryption (HVE for shorthand). It is a PE system for the Match predicate. In HVE, the attribute $\vec{x}$ associated with the ciphertext is a string of length ℓ over an alphabet Σ and keys are associated with strings $\vec{y}$ of length ℓ over the alphabet $\Sigma_\star = \Sigma \cup \{\star\}$. The predicate $\mathsf{Match}(\vec{x}, \vec{y}) = 1$ iff for each $i \in [\ell]$ it holds that $y_i = \star$ or $y_i = x_i$. HVE systems allow to search over encrypted data. For instance, consider a system where the HVE ciphertexts are associated with attributes specifyng 4 fields: 'Name', 'Surname', 'Position', 'Department'. In this scenario the owner of a HVE key for pattern $\vec{y} = (\star, 'Iovino', \star, 'CS')$ would allow its to decrypt the ciphertexts associated with attribute whose 'Surname' field is 'Iovino' and whose 'Department' field is 'CS'. From this example it is clear that HVE can be usefully applied to implement an encrypted database that allows conjunctive queries. We will show other applications of HVE in this thesis. While other powerful PE systems have been studied in literature, this thesis will focus mostly on HVE implementations that have been one of the subjects of study of the author. However, other systems, mainly inner-product encryption and Anonymous (H)IBE systems will be discussed.

The Contribution of This Thesis. The main result of this thesis will be the implementation and the security proof (based on simple assumptions) of a HVE cryptosystem secure against unrestricted adversaries. At time of writing, previous HVE systems as well as any other non-trivial predicate encryption systems were proven secure nly against a restricted class of adversaries. Namely, these systems prohibited the attacker to specify two challenges that *both* satisfy the predicates for which the adversary has obtained a corresponding key. In the present thesis, the author presents the first non-trivial PE system that overcomes this limitation.

The following works form a basis for this thesis:

- Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Pairing 2008, [27].

- Angelo De Caro, Vincenzo Iovino and Giuseppe Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In Pairing 2010, [18].

- Carlo Blundo, Vincenzo Iovino and Giuseppe Persiano. Predicate encryption with partial public keys. In Cryptology and Network Security (CANS) 2010, [4].

- Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Private-key hidden vector encryption with key confidentiality. In Cryptology and Network Security (CANS) 2009, [3].

- Angelo De Caro, Vincenzo Iovino and Giuseppe Persiano. Efficient Fully Secure (Hierarchical) Predicate Encryption for Conjunctions, Disjunctions and k-CNF/DNF formulae. Technical report avaiable from `http://eprint.iacr.org/2010/492` , [2].

- Angelo De Caro, Vincenzo Iovino and Giuseppe Persiano. Hidden Vector Encryption Fully Secure Against Unrestricted Queries - No Query Left Unanswered. Unpublished Manuscript.

- Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. To appear in ISCC 2011, [17].

Some of them are used only partially or like a reference. Some other material exhibitted in this thesis is taken, with or without modifications, from other works explicitly quoted.

1.1 Organization of The Thesis.

In Chapter 2 we will introduce the concept of Predicate Encryption system and its security notions. We will also present the very important special cases of Hidden Vector Encryption (HVE) and of Inner Product Encryption (IPE). In Chapter 3 we will present applications of Predicate Encryption to the real world and reductions among Predicate Encryption primitives. In Chapter 4 we will study the relation of the game-based notions of security given previously and simulation-based notions of security. We will show that they do not coincide and even the most elementary form of Predicate Encryption system, namely Identity-based Encryption (IBE), is impossible to achieve in the simulation-based paradigm. Next, in Chapter 5 we will present the bilinear groups. They will be our main mathematical building block which our constructions will be based on. In Chapter 6 we will present our first HVE construction. It is due to the author and Giuseppe Persiano [27]. Its security will be in the selective-id model but it is implemented on prime order bilinear groups, so it is more efficient than following constructions. We will give pointers to its implementation in Appendix C. To move to fully secure contructions we will use the Dual System Encryption methodology that it will be discussed in Chapter 7. In this Chapter we will also discuss the history and the different techniques encountered in the development of previous predicate encryption systems like IBE. As an intermediate result between our selectively secure HVE and our HVE secure against unrestricted adversary, in Chapter 8 we will present contructions for 0- and 1-secure HVE. The material here is taken from a work of the author with Angelo De Caro and Giuseppe Persiano [2]. In Chapter 9 we will make an interlude and will discuss (Anonymous) (Hierarchical) IBE (AHIBE) primitives and will present their implementation both in the public-key and in the symmetric-key model. This appeared in a work of the author with Angelo De Caro and Giuseppe Persiano [18]. In Chapter 11, we will extend the concept of Hierarchical Predicate Encryption presenting our construction for a Hierarchical Hidden Vector Encryption scheme. This Chapter comes from [2]. In Chapter 12 we will present our main result: an HVE scheme secure against unrestricted adversaries. This chapter is a novel work not previously appeared in literature. Next, in Chapter 13 we will show a construction for Inner Product Encryption that is a more general form of Predicate Encryption. Finally, we will discuss open problems and challenges in this area of research. The selection and the order of the presentation is meant to take in account the research of the author as well as the history of the developement of this area of research. Indeed, though the main result has as special case both the selectively secure scheme and the scheme secure against restricted adversaries, to present the general result directly would overshadow the obstacles and the ideas developed to reach it.

Chapter 2

Predicate Encryption Systems

In this chapter we give formal definition for Predicate Encryption schemes and its security properties.

Following standard terminology, we call a function $\nu(\lambda)$ *negligible* if for all constants $c > 0$ and sufficiently large λ, $\nu(\lambda) < 1/\lambda^c$ and denote by $[n]$ the set of integers $\{1, \ldots, n\}$. Moreover the writing "$a \leftarrow A$" for a finite set A denotes that a is randomly and uniformly selected from A.

2.1 Predicate Encryption Systems

Let Π be a class of binary predicates over a set of strings Σ and let $\mathcal{M}$ a space of messages that do not contain the special symbol $\perp$. We will often omit the reference to the message space $\mathcal{M}$. We assume that associated with each predicate P in this class there is an efficient description $< P >$ of such predicate. With slight abuse of notation we will say that an algorithm takes as input a predicate $P \in \Pi$ when it is clear that it takes as input its description $< P >$. A predicate $P \in \Pi$ maps strings $\vec{x} \in \Sigma$ of (polynomial size in the description of the predicate) to $\{0, 1\}$. Sometimes we also call such strings $\vec{x}$, vectors or attribute. We will often omit the reference to the set of definition Σ of Π. If $P(\vec{x}) = 1$ we also write that $P(\vec{x}) = TRUE$ or say that $P(\vec{x})$ is satisfied.

A Predicate Encryption scheme for the class of predicates Π and for message space $\mathcal{M}$ is a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, $\mathsf{Decrypt}$) with the following semantics.

$\mathsf{Setup}(1^\lambda)$: takes as input a security parameter λ (given in unary), and outputs the public parameters Pk and the master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, P)$: takes as input the master secret key Msk and a predicate $P \in \Pi$, and outputs a secret key $\mathsf{Sk}_{\vec{y}}$.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M)$: takes as input the public parameters Pk and a binary string $\vec{x} \in \{0, 1\}^{\mathsf{poly}}(\lambda)$ and a message $M \in \mathcal{M}$ and outputs a ciphertext Ct.

$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: takes as input the public parameters Pk, a ciphertext Ct encrypting $\vec{x}$ and M and a secret key $\mathsf{Sk}_{\vec{y}}$ and outputs a message $M' \in \mathcal{M}$.

For correctness we require that for pairs $(\mathsf{Pk}, \mathsf{Msk})$, output by $\mathsf{Setup}(1^\lambda)$, it holds that for all strings $\vec{x}\{0,1\}^{\mathsf{poly}}(\lambda)$ and predicates $P \in \Pi$, we have that

$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M), \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})) = M'$ if $P(\vec{x})$ is satisfied and a special symbol $\perp$ otherwise, except with negligible in λ probability.

2.2 Security definitions for Predicate Encryption

In this section we formalize our security requirement by means of a security game GReal_ξ parametrized by a subset ξ of $\{0,1\}$, that is ξ is one between $\{0\}$, $\{1\}$,$\{0,1\}$. Sometimes, if ξ is, respectively, either $\{0\}$ or $\{1\}$ we also omit the brackets and take ξ to be, respectively, 0 or 1. When it is clear from context we also omit the subscript ξ. GReal_ξ is played abetween a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. GReal_ξ consists of a Setup phase and of a Query Answering phase. In the Query Answering phase, the adversary can issue a polynomial number of Key Queries and one Challenge Construction query for two pairs $(\vec{x}_0, M_0)$ and $(\vec{x}_1, M_1)$ of the same length and at the end of this phase $\mathcal{A}$ outputs a guess. We stress that key queries can be issued by $\mathcal{A}$ even after he has received the challenge from $\mathcal{C}$. In GReal the adversary is restricted to queries for predicates $P \in \Pi$ such that $P(\vec{x}_0) = P(\vec{x}_1) \in \xi$ when $M_0 = M_1$ and to queries for predicates $P \in \Pi$ such that $P(\vec{x}_0) = P(\vec{x}_1) = 0$ when $M_0 \neq M_1$.

More precisely, we define game GReal_ξ in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering 1. Upon receiving a query for predicate $P \in \Pi$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, P)$.

Challenge Construction. Upon receiving the two pairs attribute/message $(\vec{x}_0, M_0)$ and $(\vec{x}_1, M_1)$ of the same length, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta, M_\eta)$.

Key Query Answering 2. Identical to the first Key Query Answering phase.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all predicates P for which $\mathcal{A}$ has issued a Key Query, it holds that either $M_0 = M_1$ and $P(\vec{x}_0) = P(\vec{x}_1) \in \xi$ or $M_0 \neq M_1$ and $P(\vec{x}_0) = P(\vec{x}_1) = 0$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\Pi,\xi}(\lambda)$ of $\mathcal{A}$ in GReal_ξ to be the probability of winning minus 1/2. When $\xi = \{0,1\}$ we omit the superscript ξ.

Definition 1. *A predicate encryption scheme for the class of predicates* Π *is* ξ-secure *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A}}_{\Pi,\xi}(\lambda)$ *is a negligible function of* λ.

It is trivial to observe that no scheme can be secure against an adversary that possesses a secret key for a predicate P such that $P(\vec{x}_0) \neq P(\vec{x}_1)$.

The above ξ-parametrized definition of security has as special case that we take as the strongest notion of security.

Definition 2. *A predicate encryption scheme for the class of predicates* Π *is* secure *if it is* $\{0, 1\}$*-secure*

Notice that for 0-security the adversaries are restricted to query for predicates P such that $P(\vec{x}_0) = P(\vec{x}_1) = 0$ when $M_0 = M_1$. We call adversaries with such a restriction *restricted adversaries.* If we let $\xi = 0$ we have the following weaker form of security.

Definition 3. *A predicate encryption scheme for the class of predicates* Π *is* secure against restricted adversaries *if it is* 0*-secure*

We also call $\{0, 1\}$-security *security against unrestricted adversaries* due to the fact that for $\{0, 1\}$-secure schemes the adversaries have no such a restriction. In literature 0-security is also called *match revealing* and $\{0, 1\}$-security *match concealing.*

2.3 Chosen-Ciphertext Attack.

Our definitions of security assume adversaries capable to perform a Chosen-Plaintext Attack (CPA). A more strong notion of security is that of security against Chosen-Ciphertext Attacks (CCA). In the latter model the adversary has also access to a decryption oracle for all ciphertexts but the challenge ciphertext. It can be viewed that CCA-secure schemes cannot be malleable meaning that given an encryption of a message M you cannot produce an encryption for $f(M)$ for some function M. This is not guaranteed by CPA-security. A general transformation by Canetti, Halevi and Katz [16] allows to convert a CPA-secure scheme into a CCA-secure scheme by a slight loss of efficiency. The transformation is generic, thus we will focus on CPA-security. A more efficient transformation was given by Boneh and Katz [11].

2.4 The Trivial Construction

Boneh and Waters [14] show a trivial construction of PE for each class of predicates assuming a public-key encryption system. Anyway, this construction is linear in the number of predicates in the class (and thus, highly inefficient) and just serves as an existential result. The construction is essentially a brute force system. Assume the existence of a secure public-key system. The setup generates a pair public-key and secret-key (of the public-key system) for each predicate in the class. For the predicates that satisfy a given attribute, the PE encryption algorithm encrypts the message by using the public-key system with the corresponding public-key and encrypts $\perp$ for the predicates that do not satisfy the attribute. Thus, the length of the ciphertext is proportional to the number of predicates in the class. The key associated with a predicate is just the secret key of the public-key system associated with this predicate. It allows to only decrypt the ciphertexts associated with attributes that satisfy the predicate. It is obvious that this PE system is correct and a standard hybrid argument reduces its security to the security of the underlying public-key system.

2.5 Attribute-hiding vs Payload-hiding

The above definitions are meant to protect both the message M and the attribute $\vec{x}$ of a ciphertext encrypted for the pair $(\vec{x}, M)$. Schemes satisfying such a property of security are also called *attribute-hiding* or *anonymous*. We could relax the definition to hide only the message M. Such schemes are told to satisfy the *payload-hiding* property. We derive this notion of security from the previous restricting the adversary to ask a challenge with $\vec{x}_0 = \vec{x}_1$. Notice that in this case, without loss of generality, we can suppose that $M_0 \neq M_1$ (otherwise the adversary would have no advantage) and thus only the $\{0,1\}$-security has sense. More precisely, we define the following game GReal (notice that is different from the previous games but for sake of simplicity we use the same name) .

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering 1. Upon receiving a query for predicate $P \in \Pi$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, P)$.

Challenge Construction. Upon receiving the two pairs attribute/message $(\vec{x}, M_0)$ and $(\vec{x}, M_1)$ of the same length, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M_\eta)$.

Key Query Answering 2. Identical to the first Key Query Answering phase.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all predicates P for which $\mathcal{A}$ has issued a Key Query, it holds that $P(\vec{x}) = 0$.

We define the advantage $\mathsf{Adv}_{\Pi}^{\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus 1/2.

Definition 4. *A predicate encryption scheme for the class of predicates* Π *is* payload-hiding *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}_{\Pi}^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ*.*

Attribute-based Encryption. In literature, Predicate Encryption schemes that only guarantee the security of the plaintext, i.e., payload-hiding schemes, are also called

Attributed-Based Encryption (ABE) schemes. Usually, they allow to achieve more experessive predicates at the cost of losing the security of the attribute.

2.6 Selective-id Security

Another weaker form of security is the selective-id model, also called selective-attribute model in the context of PE systems. In it, the adversary commits to the challenge attribute $\vec{x}_0$ and $\vec{x}_1$ *before* seeing the public-key. It is well-known that there are schemes that are selective-id secure but not secure. Anyway, proof of security in the selective-id model are either simpler to obtain or rely on simpler assumptions, so that a lot of schemes in literature were proved to be selectively-id secure. In Section 6 we present a such scheme. Henceforth, we will often remove the term 'id' from 'selective-id' and we will simply speak about 'selective' security. Even for selective security it makes sense to distinguish security parametrized by $0, 1$ or $\{0, 1\}$. We now present a formal definition of $\{0, 1\}$-selectively security. by a mean of a game GReal_ξ in the following way.

Commit. $\mathcal{A}$, given the security parameter λ, gives $\mathcal{A}$ two attributes $\vec{x}_0$ and $\vec{x}_1$.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering 1. Upon receiving a query for predicate $P \in \Pi$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, P)$.

Challenge Construction. Upon receiving the two messages M_0 and M_1 of the same length, $\mathcal{C}$ picks random $\eta \in \{0, 1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta, M_\eta)$.

Key Query Answering 2. Identical to the first Key Query Answering phase.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all predicates P for which $\mathcal{A}$ has issued a Key Query, it holds that either $M_0 = M_1$ and $P(\vec{x}_0) = P(\vec{x}_1) \in \xi$ or $M_0 \neq M_1$ and $P(\vec{x}_0) = P(\vec{x}_1) = 0$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\Pi,\xi}(\lambda)$ of $\mathcal{A}$ in GReal_ξ to be the probability of winning minus $1/2$. When $\xi = \{0, 1\}$ we omit the superscript ξ.

Definition 5. *A predicate encryption scheme for the class of predicates* Π *is* xi-selectively secure *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A}}_{\Pi,\xi}(\lambda)$ *is a negligible function of* λ*.*

It is trivial to observe that no scheme can be secure against an adversary that possesses a secret key for a predicate P such that $P(\vec{x}_0) \neq P(\vec{x}_1)$.

The above ξ-parametrized definition of selective security has as special case that we take as the strongest notion of security.

Definition 6. *A predicate encryption scheme for the class of predicates* Π *is* selectively secure *if it is* $\{0, 1\}$*-selectively secure*

Definition 7. *A predicate encryption scheme for the class of predicates* Π *is* selectively secure against restricted adversaries *if it is* 0*-selectively secure*

In Section 6.1 we will separate selective security from full security.

2.7 Full-fledged vs Predicate-only schemes

For some applications, it is useful to consider a variant of the Predicate Encryption schemes, that we call a predicate-only scheme whereas the previously defined schemes are called *full-fledged*. Here, the encryption procedure takes only the attribute $\vec{x}$ and not even the message. The procedure $\mathsf{Decrypt}$ only returns $P(\vec{x})$ when execute with a key for P and a ciphertext for $\vec{x}$. In this context, thus, it makes more sense to substitute the $\mathsf{Decrypt}$ procedure with a Test procedure that has the same semantic except that it takes only the attribute (and no message) and returns a boolean value. The ξ-security notion for predicate-only scheme is identical to that for full-fledged schemes except that, since such schemes do not involve messages, the challenge is constituted by two attributes $\vec{x}_0$ and $\vec{x}_1$ and the requirement is that the adversary cannot issue query for predicates P such that $P(\vec{x}_0) = P(\vec{x}_1) \in \xi$.

2.8 Hidden Vector Encryption

In this thesis we will focus on the very special case of Predicate Encryption that is Hidden Vector Encryption (HVE). HVE was first introduced by Boneh and Waters [14]. We now present HVE. Let $\vec{x}$ be vector of length ℓ over the alphabet $\{0, 1\}$ and $\vec{y}$ vector of the same length over the alphabet $\{0, 1, \star\}$. Define the predicate $\mathsf{Match}(\vec{x}, \vec{y}) =$ TRUE if and only if for any $i \in [\ell]$, it holds that $x_i = y_i$ or $y_i = \star$. That is, the two vectors must match in the positions j where $y_j \neq \star$.

A Hidden Vector Encryption scheme is a predicate encryption scheme for the predicate Match (formally, for the class of predicates $\{\mathsf{Match}(\cdot, \vec{y})\}_{\vec{y} \in \{0,1,\star\}^{\mathsf{poly}(\lambda)}}$). For completeness, we present of definition of a (predicate-only) Hidden Vector Encryption scheme as a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test) with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: takes as input a security parameter λ and a length parameter ℓ (given in unary), and outputs the public parameters Pk and the master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: takes as input the master secret key Msk and a vector $\vec{y} \in \{0, 1, \star\}^\ell$, and outputs a secret key $\mathsf{Sk}_{\vec{y}}$.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: takes as input the public parameters Pk and a vector $\vec{x} \in \{0, 1\}^\ell$ and outputs a ciphertext Ct.

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: takes as input the public parameters Pk, a ciphertext Ct encrypting $\vec{x}$ and a secret key $\mathsf{Sk}_{\vec{y}}$ and outputs $\mathsf{Match}(\vec{x}, \vec{y})$.

For correctness we require that for pairs $(\mathsf{Pk}, \mathsf{Msk})$, output by $\mathsf{Setup}(1^\lambda, 1^\ell)$, it holds that for all vectors $\vec{x} \in \{0, 1\}^\ell$ and $\vec{y} \in \{0, 1, \star\}^\ell$, we have that

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{x}), \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})) = \mathsf{Match}(\vec{x}, \vec{y})$ except with negligible in λ probability.

Remark 8. *Notice that the* Setup *of HVE scheme takes as input also a length parameter that is not given in the definition of Predicate Encryption scheme. Anyway it is easy to adapt the definition to take in account such* separated *parameters.*

2.8.1 Security definitions for HVE

For completeness, we present a security definition for HVE just for the case of $\xi = \{0,1\}$, and thus we will also omit the parameter. We stress that we can elaborate such definition for each $\xi \subset \{0,1\}$ analogously to what done in Section 2.2. As usual, we formalize our security requirement by means of a security game GReal between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. GReal consists of a Setup phase and of two Query Answering phases. In the Query Answering phases, the adversary can issue a polynomial number of Key Queries and one Challenge Construction query and at the end of this phase $\mathcal{A}$ outputs a guess. We stress that key queries can be issued by $\mathcal{A}$ even after he has received the challenge from $\mathcal{C}$. In GReal the adversary is restricted to queries for vectors $\vec{y}$ such that $\mathsf{Match}(\vec{y}, x_0) = \mathsf{Match}(\vec{y}, x_1)$.

More precisely, we define game GReal in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ and the length parameter ℓ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering 1.. Upon receiving a query for vector $\vec{y}$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Construction. Upon receiving the pair $(\vec{x}_0, \vec{x}_1)$, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta)$.

Key Query Answering 2. Identical to the first Key Query Answering phase.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all $\vec{y}$ for which $\mathcal{A}$ has issued a Key Query, it holds $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y})$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus 1/2.

Definition 9. *An Hidden Vector Encryption scheme is* secure *if for all probabilistic polynomial time adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HVE}}(\lambda)$ *is a negligible function of* λ.

It is trivial to observe that no scheme can be secure in the sense of Definition 2 against an adversary that possesses a secret key for a vector $\vec{y}$ such that $\mathsf{Match}(\vec{y}, \vec{x}_0) \neq \mathsf{Match}(\vec{y}, \vec{x}_1)$.

2.9 (Anonymous) Identity-Based Encryption.

Identity-based Encryption (IBE) is a very important class of predicate encryption schemes that it was initially proposed to solve the problem of distribution of certificates. IBE was first proposed by Shamir [36], and the first concrete scheme was built by Boneh and Franklin [8]. In a IBE scheme the encryptor can encrypt a message without knowing the public-key of the receiver but only specifying his/her identity. The receiver, having a special decryption key associated with his/her identity, can decrypt each ciphertext encrypted with his/her identity. IBE schemes do not guarantee the security of the identity. Anonymous IBE (AIBE) schemes also guarantee this level of security. AIBE schemes are also used to search over encrypted data. In fact, you can see identities as keywords, and having a decryption key for a keyword k allow you to check whether a given ciphertext was encrypted for the same keyword without leaking any other information neither on the message nor on the keyword of the ciphertext. Predicate-only AIBE schemes are also called *Public-key Encryption with Keyword Search* (PEKS) schemes or also *Searchable Encryption* (SE) schemes. Notice that the class of predicates of AIBE/PEKS schemes is that of the equality predicate. PEKS schemes allow to perform secure tests of equality on the encrypted data. PEKS primitive was introduced and built by Boneh, Di Crescenzo, Ostrovsky and Persiano [7].

2.10 Inner-product encryption.

Another important predicate is the *inner-product* predicate introduced by Katz, Sahai and Waters [28]. In it, the set of attributes is $\Sigma = Z_N^n$ and the class of predicates is $\Pi = \{P_{\vec{x}} \mid x \in Z_N^n\}$ with $P_{\vec{x}}(\vec{y}) = 1$ iff $<\vec{x}, \vec{y}>= 0 \mod N$. Here N is the modulus and can be a composite number and n is a length parameter. This predicate has as special case HVE. We show applications of inner-product encryption in Chapter 3.

2.11 Hierarchical PE

Another extension of the concept of PE is that of allowing a mechanism to delegate more specialized predicates from key for more general predicates. We will offer two examples of this, in Chapters 9 and 11. The first one regards Hierarchical IBE, a primitive first proposed by Horwitz and Lynn [26]. The second implements a hierarchy in the context of HVE. This was first proposed by Shi and Waters [38]. In the case of HVE it is natural to offer the possibility of delegatin a key for vector $\vec{z}$ from a key for vector $\vec{v}$ if $\vec{z}$ is lower in the hierarchy of $\vec{v}$ meaning that $\vec{z}$ agrees with to $\vec{v}$ in all positions i where $v_i \neq \star$ but, possibly, $\vec{z}$ can be different from $\vec{v}$ in some positions j where $\vec{v} = \star$. A scheme that implements this mechanism is called Hierarchical HVE (HHVE for short) and will be discussed in Chapter 11. Other hierarchical systems (not discussed in this thesis) include hierarchical inner-product encryption introduced by Okamoto and Takashima [33]. In this system, given a key for some vector $(\vec{v}, 0^n)$ over the alphabet $\mathbb{Z}_N$ you may delegate, for example, a vector $\vec{w} = (v, a, 0^{n-1})$ for some $a \in \mathbb{Z}_N$.

2.12 Symmetric-key PE

Previous definitions are formulated in the public-key setting. Notice that in it, it has no meaning to consider the security of the predicate. That is, we would like that keys would not leak any information on the associated predicate. In the public-key model this is impossible to achieve. In fact, an adversary could test whether a given key K is for the predicate P by simply creating a ciphertext for an attribute x such that $P(x)$ is satisfied. This is possible since the adversary knows the public-key. Therefore, it is natural to consider a symmetric-key scenario where the adversary does not know the public-key and to formalize the notion of key security. Shi, Shen and Waters [37] presented the first symmetric-key PE for inner product. Following this work, Blundo, Iovino and Persiano [3] considered a weaker notion for HVE. In their model, the leakage of the positions where the vector key contains a $\star$ is not considered a breach of security. The key has to hide just the positions different from $\star$. This has sense for applications as the search in an encrypted database, where is not considered dangerous if an attacker can discover which fields of the database the user is searching on until it is hidden what the user is searching. For example the user could query the database for tuples satisfying 'Name=Vincenzo' and 'Surname=Iovino'. An attacker could identifies that the user is searching in the fields 'Name' and 'Surname' rather than 'City' or 'Job', but the content of those fields is hidden, that is the attacker cannot to learn 'Vincenzo' and 'Iovino'. In Chapter 10 we will also present a symmetric-key implementation of Anonymous IBE.

2.13 PE with Partial Public-keys

We would like to stress though that 'Key Security' is not achievable in a pure public-key scenario: as told previously, given key for P for an unknown predicate P an adversary could check if $P(x)$ holds by creating a ciphertext C for attribute vector x using the public-key, and then testing P against C. In the rest of this discussion we focus on HVE. We thus can consider a *partial public-key* model in which the key owner can decide on a policy that describes which subset of the ciphertexts can be generated. More specifically, a policy $\mathsf{Pol} = \langle \mathsf{Pol}_1, \ldots, \mathsf{Pol}_\ell \rangle$ is simply a vector of length ℓ of subsets of Σ with the following intended meaning: the public key associated with policy Pol allows to create ciphertexts with attribute vector $\vec{x} = \langle x_1, \ldots, x_\ell \rangle$ iff and only for $i \in [\ell]$ we have that $x_i \in \mathsf{Pol}_i$. The private key scenario corresponds to a policy Pol with $\mathsf{Pol}_i = \emptyset$ for all i's; whereas a public key scenario corresponds to a policy with $\mathsf{Pol}_i = \Sigma$ for all i's. For example, for $\ell = 2, \Sigma = \{0,1\}$, and policy $\mathsf{Pol} = \langle \{1\}, \{0,1\} \rangle$, then public key $\mathsf{PPK}_{\mathsf{Pol}}$ associated with Pol allows to create ciphertexts with attribute vector $\vec{x} = \langle 1, 0 \rangle$ but not $\vec{x} = \langle 0, 1 \rangle$. In the formal definition of Token Security we thus require that an adversary is not able to distinguish between tokens with pattern $\vec{y}_0$ or $\vec{y}_1$ with respect to a policy Pol provided that the two patterns have the same value of the predicate Match for all attributes $\vec{x}$ that can be encrypted under policy Pol. Notice that this shares the same limitation of the previously quoted system of [3] in that the keys leak information on the $\star$'s positions. This model has been proposed by the author and Carlo Blundo and Giuseppe Persiano. Details can be found in [4].

Chapter 3

Applications

PE schemes have a lot of applications. We present a bunch of them. Then, we focus on the HVE predicate and show that other important applications can be recast as instance of it.

3.1 Outsourcing of encrypted data.

Searching over encrypted data. PE allow to search over encrypted data in an oblivious and secure way. Alice encrypts an attribute string x. By a key for a predicate y, Bob can test if the encrypted ciphertext is associated with an attribute x such that $P(x, y) = TRUE$. Such a functionality can be exploited to construct secure anti-spam check over encrypted data. In the case of HVE schemes we can search if an encrypted document satisfies a given pattern, for example if the document begins with the word "SECRET".

Outsourcing of encrypted data. Another important application raises in the context of secure databases. The owner of the database outsources his encrypted data to a remote storage. Then, he wants to retrieve from the database only the tuples satisfying a given search pattern. This is accomplished by using a symmetric-key PE scheme along with a traditional symmetric-key encryption scheme. For each pair $(value, attribute)$ the DB owner computes c, the encryption of $attribute$ given by the PE scheme, and C, the encryption of $value$ for the symmetric-key scheme. It sends to the remote storage the pair $(c.C)$. When it wants to search the presence in the DB of a tuple satisfying the pattern y, it computes the PE key for y and sends it to the remote storage. The latter uses the Test functions of the PE scheme to find tuples that satisfy the pattern, and for each of them it returns the corresponding encrypted value c, that the user decrypts by using the simmetric-key. In this scenario it makes sense to extend the security also to hide information on the *query*. That is, it would be useful to guarantee the privacy of what the user searches.

3.2 Certificate-less Public-key Infrastructures

Identity-Based Encryption (IBE) was proposed by Shamir [36] as a mechanism to avoid certificates in public-key infrastructures. To secretely send a message to a person, you may use an IBE system by specifying his/her email as recipient without the need to know his/her public-key. If the IBE is Anonymous, the transmitted encrypted message would not even leak the recipient of the message.

3.3 Applications of HVE: Comparison, range and subset queries

From the predicate HVE it is possible to derive other useful predicates. We present some of them.

Conjunctive comparison queries.

Suppose $\Sigma = \{1, \ldots, n\}^w$ for some n, w. Let $\Phi_{n,w}$ be the set of n^w predicates

$$P_{a_1,\ldots,a_w}(x_1, \ldots, x_w) = \begin{cases} 1 & \text{if } x_j \geq a_j \text{ for all } j = 1, \ldots, w, \\ 0 & \text{otherwise} \end{cases}$$

for all $\vec{a} = (a_1, \ldots, a_w) \in \{1, \ldots, n\}^w$. Then $|\Phi_{n,w}| = n^w$.

Let $\mathsf{Setup}_H, \mathsf{Encrypt}_H, \mathsf{KeyGen}_H, \mathsf{Decrypt}_H)$ be a secure HVE over $\{0,1\}^{nw}$, that is for $\ell = nw$. We construct a $\Phi_{n,w}$-searchable system as follows:

- $\mathsf{Setup}(1^\lambda)$ is the same as $\mathsf{Setup}_H(1^\lambda)$.
- $\mathsf{Encrypt}(PK, x, M)$ where $x = (x_1, \ldots, x_w) \in \{1, \ldots, n\}^w$, builds a vector $\sigma(x) = (\sigma_{i,j}) \in \{0,1\}^{nw}$ as follows:
$$\sigma_{ij} = \begin{cases} 1 & \text{if } j \geq x_i, \\ 0 & \text{otherwise} \end{cases}$$
Then, it outputs $\mathsf{Encrypt}_H(PK, \sigma(x), M)$.
- $\mathsf{KeyGen}(\mathsf{Msk}, P_{\vec{a}})$ where $\vec{a} = (a_1, \ldots, a_w) \in \{1, \ldots, n\}^w$, constructs $\sigma_\star(\vec{a}) = (\sigma_{ij}) \in \{0, 1, \star\}^{nw}$ as follow:
$$\sigma_{ij} = \begin{cases} 1 & \text{if } x_i = j, \\ \star & \text{otherwise} \end{cases}$$
Finally it outputs $\mathsf{KeyGen}_H(\mathsf{Msk}, \sigma_\star(\vec{a}))$.
- $\mathsf{Decrypt}(K_{\vec{a}}, \mathsf{Ct})$ simply outputs $\mathsf{Decrypt}_H(K_{\vec{a}}, \mathsf{Ct})$.

Notice that the system has ciphertext of size $O(nw)$ and keys of size $O(w)$. Observe that for a predicate $P_{\vec{a}} \in \Phi_{n,w}$ and an attribute $x \in \{1, \ldots, n\}^w$ we have that: $P_{\vec{a}}(x) = 1$ iff $\mathsf{Match}(\sigma(x), \sigma_\star(\vec{a})) = 1$. Therefore, correctness and security follow from the properties of the HVE.

Range queries. Comparison queries can also be extended to support range queries. To search whether $x \in [a, b]$, the encryptor encrypts a pair (x, x) and the system can generate a key for predicate that tests whether $x \geq a \wedge x \leq b$ or not.

Subset queries.

Let T be a set of size n. We associate with a subset $A \subset T$ a predicate

$$P_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

Let Φ be the set of all predicates P_σ for $\sigma \in 2^T$. Given an HVE scheme $\mathsf{Setup}_H, \mathsf{Encrypt}_H, \mathsf{KeyGen}_H, \mathsf{Decrypt}_H$ we build a Φ-searchable system as follows:

- $\mathsf{Setup}(1^\lambda)$ is the same as $\mathsf{Setup}_H(1^\lambda)$.
- $\mathsf{Encrypt}(PK, x, M)$ where $x \in T$, builds a vector $\sigma(x) = (\sigma_j) \in \{0,1\}^n$ as follows:

$$\sigma_j = \begin{cases} 1 & \text{if } x = j, \\ 0 & \text{otherwise} \end{cases}$$

  Then, it outputs $\mathsf{Encrypt}_H(PK, \sigma(x), M)$.

- $\mathsf{KeyGen}(\mathsf{Msk}, P_{\vec{a}})$ where $\vec{a} \in 2^T$, constructs $\sigma_\star(\vec{a}) = (\sigma_j) \in \{0, 1, \star\}^n$ as follow:

$$\sigma_j = \begin{cases} 0 & \text{if } j \notin A, \\ \star & \text{otherwise} \end{cases}$$

  Finally it outputs $\mathsf{KeyGen}_H(\mathsf{Msk}, \sigma_\star(\vec{a}))$.

- $\mathsf{Decrypt}(K_{\vec{a}}, \mathsf{Ct})$ simply outputs $\mathsf{Decrypt}_H(K_{\vec{a}}, \mathsf{Ct})$.

Notice that the system has ciphertext of size $O(n)$ and keys of size $O(n)$. Observe that for a predicate $P_{\vec{a}} \in \Phi$ and an attribute $x \in T$ we have that: $P_{\vec{a}}(x) = 1$ iff $\mathsf{Match}(\sigma(x), \sigma_\star(\vec{a})) = 1$. Therefore, correctness and security follow from the properties of the HVE. It is easy to extend the above construction to conjunctive subset queries.

In next section we show how to construct an encryption scheme for the class of Boolean predicates that can be expressed as a k-CNF or k-DNF formula and disjunctions from an HVE scheme.

We first start by giving formal definitions for the Boolean Satisfaction Problem and its security properties.

3.4 Boolean Satisfaction Encryption.

Let $\mathbb{B} = \{\mathbb{B}_n\}_{n>0}$ be a class of Boolean predicates indexed by the number n of variables. We define the $\mathsf{Satisfy}$ predicate as $\mathsf{Satisfy}(\Phi, \vec{z}) = \Phi(\vec{z})$ for $\vec{z} \in \{0,1\}^n$.

An *Encryption scheme for class* $\mathbb{B}$ is a tuple of four efficient probabilistic algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test) with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^n)$: takes as input a security parameter λ and the number n of variables, and outputs the public parameters Pk and the master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \Phi)$: takes as input the master secret key Msk and a formula $\Phi \in \mathbb{B}_n$ and outputs a secret key Sk_Φ.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{z})$: takes as input the public parameters Pk and a truth assignment $\vec{z}$ for n variables and outputs a ciphertext Ct.

$\mathsf{Test}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk}_\Phi)$: takes as input the public parameters Pk, a ciphertext Ct and a secret key Sk_Φ and outputs TRUE iff and only if the ciphertext is an encryption of a truth assignment $\vec{z}$ that satisfies Φ.

Correctness of Boolean Satisfaction Encryption. We require that for all pairs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, it holds that for any truth assignment $\vec{z}$ for n variables, for any formula $\Phi \in \mathcal{B}_n$ over n variables we have that the probability that
$\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{z}), \mathsf{KeyGen}(\mathsf{Msk}, \Phi)) \neq \mathsf{Satisfy}(\Phi, \vec{z})$ is negligible in λ.

3.4.1 Security Definitions for Boolean Satisfaction Encryption.

For Boolean Satisfaction encryption, we have a game similar to that of HVE. GReal can be described in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm, $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$. Then $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. For $\Phi \in \mathbb{B}_n$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, \Phi)$.

Challenge Construction. Upon receiving the pair $(\vec{z}_0, \vec{z}_1)$ of truth assignments over n variables, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, \vec{z}_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and, for all Φ for which $\mathcal{A}$ has issued a Key Query, it holds that $\mathsf{Satisfy}(\Phi, z_0) = \mathsf{Satisfy}(\Phi, z_1)$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathbb{B}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus 1/2.

Definition 10. *An Encryption scheme for class* $\mathbb{B}$ *is* secure *if for all PPT adversaries* $\mathcal{A}$*, we have that* $\mathsf{Adv}^{\mathcal{A}}_{\mathbb{B}}(\lambda)$ *is a negligible function of* λ*.*

3.4.2 Reducing k-CNF to HVE

We consider formulae Φ in k-CNF, for constant k, over n variables in which each clause $C \in \Phi$ contains exactly k distinct variables. We call such a clause *admissible* and denote by $\mathbb{C}_n$ the set of all admissible clauses over the n variables $x_1, \ldots, x_n$ and set $M_n = |\mathbb{C}|$. Notice that $M_n = \Theta(n^k)$. We also fix a canonical ordering $C_1, \ldots, C_{M_n}$ of the clauses in $\mathbb{C}_n$.

Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct a k-CNF scheme $\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ as follows:

$\mathsf{Setup}_{\mathsf{kCNF}}(1^\lambda, 1^n)$: The algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^\lambda, 1^{M_n})$.

$\mathsf{KeyGen}_{\mathsf{kCNF}}(\mathsf{Msk}, \Phi)$: For a k-CNF formula Φ, the key generation algorithm constructs vector $\vec{y} \in \{0, 1, \star\}^{M_n}$ by setting, for each $i \in \{1, \ldots, M_n\}$, $y_i = 1$ if $C_i \in \Phi$; $y_i = \star$ otherwise. We denote this transformation by $y = \mathsf{FEncode}(\Phi)$. Then the key generation algorithm returns $\mathsf{Sk}_\Phi = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \vec{y})$.

$\mathsf{Encrypt}_{\mathsf{kCNF}}(\mathsf{Pk}, \vec{z})$: The algorithm constructs vector $\vec{x} \in \{0, 1\}^{M_n}$ in the following way: For each $i \in \{1, \ldots, M_n\}$ the algorithms sets $x_i = 1$ if C_i is satisfied by $\vec{z}$; $x_i = 0$ if C_i is not satisfied by $\vec{z}$. We denote this transformation by $\vec{x} = \mathsf{AEncode}(\vec{z})$. Then the encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \vec{x})$.

$\mathsf{Test}_{\mathsf{kCNF}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$: The algorithm returns the output of $\mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_\Phi, \mathsf{Ct})$.

Correctness. Correctness follows from the observation that for formula Φ and assignment $\vec{z}$, we have that $\mathsf{Match}(\mathsf{AEncode}(\vec{z}), \mathsf{FEncode}(\Phi)) = 1$ if and only if $\mathsf{Satisfy}(\Phi, \vec{z}) = 1$.

Security. Let $\mathcal{A}$ be an adversary for kCNF that tries to break the scheme for n variables and consider the following adversary $\mathcal{B}$ for $\mathcal{H}$ that uses $\mathcal{A}$ as a subroutine and tries to break a $\mathcal{H}$ with $\ell = M_n$ by interacting with challenger $\mathcal{C}$. $\mathcal{B}$ receives a Pk for $\mathcal{H}$ and passes it to $\mathcal{A}$. Whenever $\mathcal{A}$ asks for the key for formula Φ, $\mathcal{B}$ constructs $\vec{y} = \mathsf{FEncode}(\Phi)$ and asks $\mathcal{C}$ for a key $\mathsf{Sk}_{\vec{y}}$ for $\vec{y}$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge by providing truth assignments $\vec{z}_0$ and $\vec{z}_1$, $\mathcal{B}$ simply computes $\vec{x}_0 = \mathsf{AEncode}(\vec{z}_0)$ and $\vec{x}_1 = \mathsf{AEncode}(\vec{z}_1)$ and gives the pair $(\vec{x}_0, \vec{x}_1)$ to $\mathcal{C}$. $\mathcal{B}$ then returns the challenge ciphertext obtained from $\mathcal{C}$ 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 Φ we have that $\mathsf{Satisfy}(\Phi, \vec{z}_0) = \mathsf{Satisfy}(\Phi, \vec{z}_1)$, then all $\mathcal{B}$'s queries $\vec{y}$ to $\mathcal{C}$ also have the property $\mathsf{Match}(\vec{y}, \vec{x}_0) = \mathsf{Match}(\vec{y}, \vec{x}_1)$. Thus $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s. By combining the above reduction with our constructions for HVE, we have the following theorems.

Theorem 11. *For any constant $k > 0$, if Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates that can be represented by k-CNF formulae.*

3.4.3 Reducing Disjunctions to HVE

In this section we consider the class of Boolean predicates that can be expressed as a single disjunction. We assume without loss of generality that a disjunction does not contain a variable and its negated.

Let $\mathcal{H} = (\mathsf{Setup}_{\mathcal{H}}, \mathsf{KeyGen}_{\mathcal{H}}, \mathsf{Encrypt}_{\mathcal{H}}, \mathsf{Test}_{\mathcal{H}})$ be an HVE scheme and construct the predicate-only scheme $\vee = (\mathsf{Setup}_{\vee}, \mathsf{KeyGen}_{\vee}, \mathsf{Encrypt}_{\vee}, \mathsf{Test}_{\vee})$ for disjunctions in the following way:

$\mathsf{Setup}_{\vee}(1^{\lambda}, 1^n)$: the algorithm returns the output of $\mathsf{Setup}_{\mathcal{H}}(1^{\lambda}, 1^n)$.

$\mathsf{KeyGen}_{\vee}(\mathsf{Msk}, C)$: For a clause C, the key generation algorithm constructs vector $\vec{y} \in \{0, 1, \star\}^n$ in the following way. Let $\vec{w}$ be a truth assignment to the n variables that does not satisfy clause C. For each $i \in \{1, \ldots, n\}$, the algorithms sets $y_i = w_i$ if the i-th variable appears in C; $y_i = \star$ otherwise. We denote this transformation by $\vec{y} = \mathsf{CEncode}(C)$. The output is $\mathsf{Sk}_C = \mathsf{KeyGen}_{\mathcal{H}}(\mathsf{Msk}, \vec{y})$.

$\mathsf{Encrypt}_{\vee}(\mathsf{Pk}, \vec{z})$: The encryption algorithm returns $\mathsf{Ct} = \mathsf{Encrypt}_{\mathcal{H}}(\mathsf{Pk}, \vec{z})$.

$\mathsf{Test}_{\vee}(\mathsf{Sk}_C, \mathsf{Ct})$: The algorithm returns $1 - \mathsf{Test}_{\mathcal{H}}(\mathsf{Sk}_C, \mathsf{Ct})$.

Correctness. It follows from the observation that for a clause C and assignment $\vec{z}$, $\mathsf{Satisfy}(C, \vec{z}) = 1$ if and only if $\mathsf{Match}(\mathsf{CEncode}(C), \vec{z}) = 0$.

Security. It is easy to see that if $\mathcal{H}$ is secure then $\vee$ is secure. In particular, notice that if for $\mathcal{A}$'s query C we have that $\mathsf{Satisfy}(C, \vec{z}_0) = \mathsf{Satisfy}(C, \vec{z}_1) = \xi \in \{0, 1\}$, then for $\mathcal{B}$'s query $\vec{y} = \mathsf{CEncode}(C)$ to $\mathcal{C}$ we have that $\mathsf{Match}(\vec{y}, \vec{z}_0) = \mathsf{Match}(\vec{y}, \vec{z}_1) = 1 - \xi$.

Theorem 12. *If Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates that can be represented by a disjunction.*

3.4.4 Reducing k-DNF to k-CNF

We observe that if Φ is a predicate represented by a k-DNF formula then its negation $\bar{\Phi}$ can be represented by a k-CNF formula. Therefore let
$\mathsf{kCNF} = (\mathsf{Setup}_{\mathsf{kCNF}}, \mathsf{KeyGen}_{\mathsf{kCNF}}, \mathsf{Encrypt}_{\mathsf{kCNF}}, \mathsf{Test}_{\mathsf{kCNF}})$ and consider the following
scheme $\mathsf{kDNF} = (\mathsf{Setup}_{\mathsf{kDNF}}, \mathsf{KeyGen}_{\mathsf{kDNF}}, \mathsf{Encrypt}_{\mathsf{kDNF}}, \mathsf{Test}_{\mathsf{kDNF}})$. The setup algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ is the same as $\mathsf{Setup}_{\mathsf{kCNF}}$. The key generation algorithm $\mathsf{Setup}_{\mathsf{kDNF}}$ for predicate Φ represented by a k-DNF simply invokes the key generation algorithm $\mathsf{Setup}_{\mathsf{kCNF}}$ for $\bar{\Phi}$ that can be represented by a k-CNF formula. The encryption algorithm $\mathsf{Encrypt}_{\mathsf{kDNF}}$ is the same as $\mathsf{Encrypt}_{\mathsf{kCNF}}$. The test algorithm $\mathsf{Test}_{\mathsf{kDNF}}$ on input ciphertext Ct and key for k-DNF formula Φ (that is actually a for k-CNF formula $\bar{\Phi}$) thus $\mathsf{Test}_{\mathsf{kCNF}}$ on Ct and the key and complements the result. Correctness and security can be easily argued as for Disjunctions.

By combining the above reduction with the construction given by Theorem 11.

Theorem 13. *If Assumption 1 and 2 hold for generator $\mathcal{G}$ then there exists a secure encryption scheme for the class of predicates represented by k-DNF formulae.*

3.5 Applications of Inner-product Encryption

From the predicate inner-product it is possible to derive other useful predicates. The first one is HVE and its dual.

3.5.1 HVE and Dual HVE.

We first show how to obtain an HVE scheme ($\mathsf{Setup}_H, \mathsf{KeyGen}_H, \mathsf{Encrypt}_H, \mathsf{Decrypt}_H$) for alphabet $\Sigma = \mathbb{Z}_N$ from any inner-product encryption scheme ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$) of dimension 2ℓ.

- Setup_H is the same of Setup.
- To generate a secret key corresponding to the vector $\vec{a} = (a_1, \dots, a_\ell)$, first construct $\vec{A} = (A_1, \dots, A_{2\ell})$ as follows:

$$\begin{cases} A_{2i-1} = 1, A_{2i} = a_i & \text{if } a_i \neq \star, \\ A_{2i-1} = 0, A_{2i} = 0 & \text{if } a_i = \star. \end{cases}$$

  Then output the key for A obtained by running KeyGen_H.
- To encrypt a message M for the attribute $\vec{x} = (x_1, \dots, x_\ell)$, choose random $\vec{r} = (r_1, \dots, r_\ell) \in \mathbb{Z}_N^\ell$ and construct a vector $\vec{X}_{\vec{r}} = (X_1, X_{2\ell})$ as follows:

$$X_{2i-1} = -r_i \cdot x_i, X_{2i} = r_i$$

  (where the operations are done modulo N). Then output the ciphertext for M and $\vec{X}_{\vec{r}}$ computed by $\mathsf{Encrypt}_H$.

To see that correctness holds, let $\vec{a}, \vec{A}, \vec{x}, \vec{r}, \vec{X}_{\vec{r}}$ be as above. Then:

$$\mathsf{Match}(\vec{x}, \vec{a}) = 1 \rightarrow \forall \vec{r} :< \vec{A}, \vec{X}_{\vec{r}} >= 0.$$

Furthermore, assuming $gcd(a_i - x_i, N) = 1$ for all i:

$$\mathsf{Match}(\vec{x}, \vec{a}) = 0 \rightarrow \text{Prob}[< \vec{A}, \vec{X}_{\vec{r}} >= 0] = 1/N,$$

which is negligible in the security parameter. Analogously, one can prove security of the construction as well.

A straightforward modification of the above gives a scheme that is the dual of HVE, in the sense that the $\star$'s are in the ciphertexts. Yet, another generalization could be of allowing $\star$'s both in ciphertexts and keys. This can be implemented by using inner-product encryption in a similar way as above.

3.5.2 Polynomial evaluation encryption

We can also construct predicate encryption schemes for predicates corresponding to polynomial evaluation. Let $\Phi^{\mathsf{poly}}_{\leq d} = \{f_p | p \in \mathbb{Z}_N[x], deg(p) \leq d\}$, where

$$\Phi_p(x) = \begin{cases} 1, \text{ if } p(x) = 0 \\ 0 \text{otherwise} \end{cases}$$

for $x \in \mathbb{Z}_N$. Given an inner-product encryption scheme $(\mathsf{Setup}_I, \mathsf{KeyGen}_I, \mathsf{Encrypt}_I, \mathsf{Decrypt}_I)$ of dimension $d+1$, we can construct a predicate encryption scheme for $\Phi^{\mathsf{poly}}_{\leq d}$ as follows:

- The setup algorithm is unchanged.
- To generate a secret key corresponding to the polynomial $p(x) = a_d x^d + \ldots + a_1 x + a_0$, set $\vec{p} = (a_d, \ldots, a_0)$ and output a key for $\vec{p}$ obtained by KeyGen_I.
- To encrypt a message M for the attribute $w \in \mathbb{Z}_N$, set $\vec{w} = (w^d \mod \mathbb{Z}_N, \ldots, w \mod N, 1)$ and output the ciphertext for M and $\vec{w}$ computed by $\mathsf{Encrypt}_I$.

Since $p(w) = 0$ iff $< \vec{p}, \vec{w} >= 0$, correctness and security follow. The above shows that we can construct construct PE schemes for predicates corresponding to univariate polynomials whose degree d is polynomial in the security parameter. This can be generalized to the case of polynomials in t varibles, and degree at most d in each variable, as lons as d^t is polynomial in the security parameter. We can also construct schemes that are the dual of the above, in which attributes correspond to polynomials and predicates involve the evaluation of the input polynomial at some fixed point.

3.5.3 CNF and DNF encryption from inner-product encryption

We now present a way of obtaining CNF/DNF encryption from inner-product encryption. The general construction is exponential in the size of the formulae but some it can be useful for some applications where the kind of allowed formulae is limited. We start by constructing disjunctions of equality tests. For example, the predicate OR_{I_1,I_2} where $OR_{I_1,I_2}(x) = 1$ iff either $x = I_1$ or $x = I_2$ can be encoded as the univariate polynomial $p(x) = (x - I_1)(x - I_2)$, which evaluates to 0 iff the relevant predicate evaluates is satisfied. Similarly, the predicate OR_{I_1,I_2} where $OR_{I_1,I_2}(x_1, x_2) = 1$ iff either $x_1 = I_1$ or $x_2 = I_2$ can be encoded as the bivariate polynomial $p'(x_1, x_2) = (x_1 - I_1)(x_2 - I_2)$. We now show how to handle conjunctions. Consider, the predicate AND_{I_1,I_2} where $AND_{I_1,I_2}(x_1, x_2) = 1$ iff both $x_1 = I_1$ and $x_2 = I_2$. Here, we build the secret key by choosing random $r \in \mathbb{Z}_N$ and letting such a secret key correspond to the polynomial $p''(x_1, x_2) = r \cdot (x_1 - I_1) + (x_2 - I_2)$. If $AND_{I_1,I_2}(x_1, x_2)$ is satisfied then $p''(x_1, x_2) = 0$, whereas if it is not satisfied then with very high probability over choice of r, the polynomial will not evalute to 0. We notice that the resulting scheme can only be proven selective-id secure even if the underlying polynomial equation encryption scheme is fully secure. The reason is that the secret key may leak the value r in which case the adversary will be able to find x_1 and x_2 such that $AND_{I_1,I_2}(x_1, x_2) \neq 1$ yet $p''(x_1, x_2) = 0$. This is not a problem in the selective-id setting where the adversary must commit to x_1 and x_2 to the outset of the experiment. The above ideas can be extended to handle arbitrary CNF and DNF formulae but as pointed out in the previous section the complexity of the resulting scheme is exponential in the number of variables.

3.5.4 Exact Threshold Encryption

We consider the setting of fuzzy IBE [35], which can be mapped to the predicate encryption framework as follows: fix a set $A = \{1, ..., l\}$ and let the set of attributes be all subsets of A. Predicates take the form $\Phi = \{\Phi_S | S \subset A\}$ where $\Phi_S(S) = 1$ iff the intersection of S and S' has size at least t. Sahai and Waters [35] show a construction of a payload-hiding predicate encryption scheme for this class of predicates. By using inner-product encryption, we can construct a scheme where the attribute space is the same as before, but the class of predicates corresponding to overlap in exactly t positions. (the scheme will also be attribute hiding.) Namely, set $\Phi = \{\Phi_S | S \subset A\}$ with $\Phi_S(S) = 1$ iff the intersection of S and S has size exactly t. Then, given any inner product encryption scheme of dimension $l + 1$:

- The setup algorithm is unchanged.
- To generate a secret key for the predicate Φ_S , first define a vector $\vec{v} \in \mathbb{Z}_N^{l+1}$ as follows: for $1 \leq i \leq l : v_i = 1$ iff $i \in S$ and $v_{l+1} = 1$. Then output the key for the vector $\vec{v}$ computed by using the inner-product scheme.
- To encrypt a message M for the attribute $S \subset A$, define a vector $\vec{v}$ as follows: for $1 \leq i \leq l : v_i = 1$ iff $i \in S$ and $v_{l+1} = t \mod N$. Then output the ciphertext for M and $\vec{v}$ computed by using the inner-product scheme. Since the intersection of S and S has size t exactly when $< v, v >= 0$, correctness and security follow.

Chapter 4

The limitations of game-based security

In this chapter we initiate a study of alternative security notions for predicate encryption systems. In particular we show that for these primitives the notions of simulation-based security and game-based security do not coincide. Following recent line of research, we also extend the concept of PE to Functional Encryption (FE) schemes that, roughly speaking, are like PE except that the predicates are not necessarily binary but can be an arbitrary efficently computable function. The stuff of this chapter is based (with few modifications) on the work of [13]. We first present a formal definition of FE schemes.

4.1 Functional Encryption Schemes

A functionality F is defined as follows.

Definition 14. *A functionality F defined over (K, X) is an efficiently computable function $F : K \times X \to \{0,1\}^\star$. The set K is called the key space and the set X is called the plaintext space. We require that the key space contain a special key ϵ called the empty key.*

A functional encryption scheme for the functionality F enalbes one to evalutate $F(k, x)$ given the encryption of x and a key sk_k for k. The algorithm that, given encryption of x and sk_y returns $F(y, x)$ is called *decryption*.

Definition 15. *A functional encryption scheme for the functionality F is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt})$ such that for all $x \in X$ and $k \in K$ it holds that:*

For all pairs $(\mathsf{Pk}, \mathsf{Msk})$, output by $\mathsf{Setup}(1^\lambda)$, we have that
$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, x), \mathsf{KeyGen}(\mathsf{Msk}, k)) = F(x, k)$
except with negligible in λ probability.

Relation to PE and other primitives. It is easily seen that PE systems are special case of FE. The definition given above only captures the *sintax* of FE, and, thus, it also includes primitives payload-hiding secure like IBE and Attribute-based Encryption in which the security of the plaintetex $x \in X$ is not a concern.

The role of the empty key. The empty key ϵ is useful to capture the information that eventually a ciphertext may leak. An example of such leakage could be that of the *length* of the plaintext that, as we know, it is impossible to hide. Another example could be in the context of IBE or ABE schemes that do not hide the *attributes*; in this case the empty key can be used to obtain such information.

4.2 Game-based security for Functional Encryption

In this section we present a game-based definition of security for FE. It is essentially similar to that for PE. We formalize our security requirement by means of a security game GReal between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. GReal consists of a Setup phase and of a Query Answering phase. In the Query Answering phase, the adversary can issue a polynomial number of Key Queries and one Challenge Construction query and at the end of this phase $\mathcal{A}$ outputs a guess. We stress that key queries can be issued by $\mathcal{A}$ even after he has received the challenge $(x_0, x_1) \in X^2$ from $\mathcal{C}$. In GReal the adversary is restricted to queries for $\in K$ such that $F(k, x_0) = F(k, x_1)$. Indeed if such condition it is not satisfied it is obvious that no FE scheme can be secure according to such definition.

More precisely, we define game GReal in the following way.

Setup. $\mathcal{C}$ runs the Setup algorithm on input the security parameter λ (given in unary) to generate public parameters Pk and master secret key Msk. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. Upon receiving a query for $k \in K$, $\mathcal{C}$ returns $\mathsf{KeyGen}(\mathsf{Msk}, k)$.

Challenge Construction. Upon receiving the pair $(x_0, x_1) \in X^2$, $\mathcal{C}$ picks random $\eta \in \{0, 1\}$ and returns $\mathsf{Encrypt}(\mathsf{Pk}, x_\eta)$.

Winning Condition. Let η' be $\mathcal{A}$'s output. We say that $\mathcal{A}$ *wins* the game if $\eta = \eta'$ and for all k for which $\mathcal{A}$ has issued a Key Query, it holds $F(k, x_0) = F(k, x_1)$.

We define the advantage $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{FE}}(\lambda)$ of $\mathcal{A}$ in GReal to be the probability of winning minus 1/2.

4.3 Insufficiency of the game-based notion of security.

The previous game-based notion of security is a natural generalization of analogous definition for standard public-key encryption. Unfortunately, this definition is insufficient for some functionalities. We present an example due to [13]. Let π is a fixed one-way permutation and consider the functionality F that admits only the empty key defined as $F(\epsilon, x) = \pi(x)$. It is easy to observe that a FE scheme that encrypts x by outputting $\pi(x)$ is secure both according to the game-based and simulation-based notion of security. Anyway we can consider an incorrect implementation of this functionalities. This can be given by a FE scheme that encrypts x by simply outputting x. Clearly, this system should be consider not secure because it leaks all the information about the plaintext. Anyway, it satisfies the game-based definition of security because the constraint impose that $F(\epsilon, x_0) = F(\epsilon, x_1)$, that is true iff $x_0 = x_1$. Furthermore it does not satisfy the simulation-based security given in next section. Intuitively this is true because if x is randomly chosen the real adversary would be able to recover x always whereas for the simulator to recover x it would have to break the one-wayness of π.

4.4 Simulation-based security for Functional Encryption.

We present a simulation-based notion of security that captures the natural intuition that a secret key for k should only reveal $F(k, x)$ when given an encryption of x. In what follows, we indicate by $A^{B(\cdot)[[x]]}$ that algorithm A can issue a query q to its oracle B, at which point $B(q, x)$ will be executed and output a pair (y, x'). The value y is returned to A as result of its query and the value x is updated to x' and is fed to B next time the oracle is queried.

Definition 16. *A FE scheme is simulation-secure if there exist an (oralce) PPT algorithm $Sim = (Sim_1, Sim_O, Sim_2)$ such that for any (oracle) PPT algorithms $Message$ and Adv the following two distributions are computationally indinstinguishable (over λ):*

Real Distribution

$(PK, MSK) \leftarrow \mathsf{Setup}(1^\lambda)$
$(x, \tau) \leftarrow Message^{\mathsf{KeyGen}(MSK,\cdot)}(PK)$
$c \leftarrow \mathsf{Encrypt}(PK, x)$
$\alpha \leftarrow Adv^{\mathsf{KeyGen}(MSK,\cdot)}(PK, c, \tau)$
Let $y_1, \dots, y_n$ be the queries to KeyGen *issued by $Message$ and Adv in the previous steps.*
Output $(PK, x, \tau, \alpha, y_1, \dots, y_n)$.

Ideal Distribution

$(PK, \sigma) \leftarrow \mathcal{S}_1(1^\lambda)$
$(x, \tau) \leftarrow Message^{Sim_O(\cdot)[[\sigma]]}(PK)$
$\alpha \leftarrow Sim_2^{F(\cdot,x), Adv^O(PK,\cdot,\tau)}(\sigma, F(\epsilon, x))$
Let $y_1, \dots, y_n$ be the queries to F issued by Sim in the previous steps.
Output $(PK, x, \tau, \alpha, y_1, \dots, y_n)$.

We also consider a weakening of this definition.

Definition 17. *A FE scheme is weakly simulation-secure if for any (oracle) PPT algorithms $Message$ and Adv there exists an (oracle) PPT algorithm Sim the following two distributions are computationally indinstinguishable (over λ):*

Real Distribution

$(PK, MSK) \leftarrow \mathsf{Setup}(1^\lambda)$
$(x, \tau) \leftarrow Message(1^\lambda)$
$c \leftarrow \mathsf{Encrypt}(PK, x)$
$\alpha \leftarrow Adv^{\mathsf{KeyGen}(MSK,\cdot)}(PK, c, \tau)$
Let $y_1, \ldots, y_n$ be the queries to KeyGen *issued by $Message$ and Adv in the previous steps. ehe random plaintext is independent from the view of Sim so that every simulated transcript is distinguishable from the transcript output by the adversary. Output $(x, \tau, \alpha, y_1, \ldots, y_n)$.*

Ideal Distribution
$(x, \tau) \leftarrow Message(1^\lambda)$
$\alpha \leftarrow Sim^{F(\cdot,x)}(1^\lambda, \tau, F(\epsilon, x))$
Let $y_1, \ldots, y_n$ be the queries to F issued by Sim in the previous steps
Output $(x, \tau, \alpha, y_1, \ldots, y_n)$.

We are now ready to state a theorem due to [13].

Theorem 18 ([13])**.** *Then F be the IBE functionality. Then, there does not exist any weakly simulation-secure FE scheme for F in the non-programmable random oracle model.*

Proof. We present a sketch of this proof. Let H represent the random oracle. Consider the following concrete adversary algorithms: $Message(1^\lambda)$ works as follows: Let len_{sk} be the maximum bit length produced by the keygen algorithm for the key 0 for security parameter λ. Then the vector x (that is, the concatenation of the message and the identity) consists of the following elements: for $i = 1, ..., len_{sk} + \lambda$, the element $(r_i, 0)$ where r_i is a randomly and independently chosen bit for each i. The value τ is empty. That is the messagge is the random vector $\vec{r}$ and the identity is 0. $Adv^{\mathsf{KeyGen}(MSK,)}(PK, c, \tau)$ works as follows: call the random oracle H on the input (pp, c) to obtain a string w of length λ. Now request the secret key for the identity (w) first, and then for the identity 0. Use the key for identity 0 to decrypt the entire ciphertext. Output a full transcript of the entire computation done by Adv, including all calls to the random oracle and the interaction with the KeyGen oracle. Now consider what Sim must do in order to output a distribution indistinguishable from the real interaction. Because Adv only makes a single key query of the form (w), it is the case that Sim must make exactly one query its first query to F of this form as well. Furthermore, the distinguisher can check if this w is the output of H applied to some string of the form (PK, c). Thus, the simulator must perform this query to H before making any queries to F . The simulator at this point has no information whatsoever about the plaintexts r_i (which is only revealed when the simulator queries F for identity 0 afterwards). Thus, this fixed string $z = (pp, c)$ has the (impossible) property that after receiving only len_{sk} bits of information, it can deterministically decode z to be a an arbitrary string of length $len_{sk} + \lambda$. □

The same work also shows that in the *programmable* random oracle model, simulation-secure IBE schemes do exist.

Chapter 5

Bilinear groups

In this chapter we present the fundamental building block for our constructions. In the selectively secure construction of Chapter 6 we will use bilinear groups of prime order. In other constructions, we will need bilinear groups of composite order. An excellent survey of the mathematics of bilinear maps can be found in the thesis of Ben Lynn [32]. Another useful reference is a survey by Galbraith, Paterson and Smart [21].

5.1 Symmetric bilinear groups of prime order

We use multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p and a non-degenerate pairing function $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. That is, for all $g \in \mathbb{G}, g \neq 1$, $\mathbf{e}(g, g) \neq 1$ and $\mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{ab}$. We denote by g and $\mathbf{e}(g, g)$ generators of $\mathbb{G}$ and $\mathbb{G}_T$. We call a *symmetric bilinear* instance a tuple $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$ and assume that there exists an efficient generation procedure that, on input security parameter 1^k, outputs an instance with $|p| = \Theta(k)$.

5.2 Bilinear groups of composite order

Composite order bilinear groups were first used in Cryptography by [10] (see also [5]). Some of its properties are very similar to that for the prime order groups, but we remark them for completeness. We suppose the existence of an efficient group generator algorithm $\mathcal{G}$ which takes as input the security parameter λ and outputs a description $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ of a bilinear setting, where $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order N, and $\mathbf{e} : \mathbb{G}^2 \to \mathbb{G}_T$ is a map with the following properties:

1. (Bilinearity) $\forall\ g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_N$ it holds that $\mathbf{e}(g^a, h^b) = \mathbf{e}(g, h)^{ab}$.
2. (Non-degeneracy) $\exists\ g \in \mathbb{G}$ such that $\mathbf{e}(g, g)$ has order N in $\mathbb{G}_T$.

We assume that the group descriptions of $\mathbb{G}$ and $\mathbb{G}_T$ include generators of the respective cyclic groups. We require that the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map $\mathbf{e}$ are computable in deterministic polynomial time in λ. In some of our constructions we will make hardness assumptions for bilinear settings whose order N is product of more than two distinct primes each of length $\Theta(\lambda)$. For an integer m dividing N, we let $\mathbb{G}_m$ denote the subgroup of $\mathbb{G}$ of order m. 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 $\mathbf{e}(g, h) = 1$. This is called the *orthogonality* property and is a crucial tool in our constructions.

Chapter 6

Selective secure HVE

In this chapter we present an implementation of an HVE scheme proved secure in the selective model against *restricted* adversaries. The proof offers an example of the partitioning paradigm (we will disccus it in Chapter 7). First, we will show the weakness of selective security.

6.1 Weakness of Selective Security

Selective security is weaker than full security. Assuming the existence of a selectively secure scheme, it is easy to construct a related selectively secure scheme that is not fully secure. We sketch a proof. Consider the case of IBE. Assume the existence of a selectively secure IBE $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. We construct a related system $(\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ as follows. The Setup' of the news ystem selects a random identity id that records in the Pk. The rest of the procedure is unchanged. The $\mathsf{Encrypt}'$ algorithm works as usual for each identity different from id but for id it returns the plaintext in clear. The KeyGen' and the decryption procedures are unchanged. It is trivial to analyze the selective security of the new system. Indeed, with very high probability the view of any adversary attacking the new system is identical to the view of the same adversary attacking the old system. Anyway, the system is not full secure because if the adversary sees the public-key, it can choose as its challenge identity id so that it receives the challenge message in clear.

6.2 Complexity Assumptions for the selectively secure HVE

In our construction we make the following hardness assumptions.

Decision BDH. Given a tuple $[g, g^{z_1}, g^{z_2}, g^{z_3}, Z]$ for random exponents $z_1, z_2, z_3 \in \mathbb{Z}_p$ it is hard to distinguish between $Z = \mathbf{e}(g,g)^{z_1 z_2 z_3}$ and a random Z from $\mathbb{G}_T$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{DBDHExp}_{\mathcal{A}}$ as follows.

```
$\mathsf{DBDHExp}^{\mathcal{A}}(1^k)$
01. Choose instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$ with security parameter $1^k$;
02. Choose $a, b, c \in \mathbb{Z}_p$ at random;
03. Choose $\eta \in \{0, 1\}$ at random;
04. **if** $\eta = 1$ **then** choose $z \in \mathbb{Z}_p$ at random
05.     **else** set $z = abc$;
06. set $A = g^a, B = g^b, C = g^c$ and $Z = \mathbf{e}(g, g)^z$;
07. let $\eta' = \mathcal{A}(\mathcal{I}, A, B, C, Z)$;
08. if $\eta = \eta'$ **then** return 0 **else** return 1;
```

Assunption Bilinear Decisional Diffie-Hellman For all probabilistic polynomial-time algorithms $\mathcal{A}$,

$$\left|\mathrm{Prob}[\mathsf{DBDHExp}^{\mathcal{A}}(1^k) = 1] - 1/2|\right| = \nu(k)$$

for some negligible function ν.

Decision Linear. Given a tuple $[g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^s, Z]$ for random exponents $z_1, z_2, z_3, s \in \mathbb{Z}_p$ it is hard to distinguish between $Z = g^{z_2(s-z_3)}$ and a random Z from $\mathbb{G}$. More specifically, for an algorithm $\mathcal{A}$ we define experiment $\mathsf{DLExp}_{\mathcal{A}}$ as follows.

```
$\mathsf{DLExp}^{\mathcal{A}}(1^k)$
01. Choose instance $\mathcal{I} = [p, \mathbb{G}, g]$ with security parameter $1^k$;
02. Choose $z_1, z_2, z_3, s \in \mathbb{Z}_p$ at random;
03. Choose $\eta \in \{0, 1\}$ at random;
04. **if** $\eta = 1$ **then** choose $z \in \mathbb{Z}_p$ at random
05.     **else** set $z = z_2(s - z_3)$;
06. set $Z_1 = g^{z_1}, Z_2 = g^{z_2}, Z_{13} = g^{z_1 z_3}, S = g^s$, and $Z = g^z$;
07. let $\eta' = \mathcal{A}(\mathcal{I}, Z_1, Z_2, Z_{13}, S, Z)$;
08. if $\eta = \eta'$ **then** return 0 **else** return 1;
```

Assumption Decision Linear For all probabilistic polynomial-time algorithms $\mathcal{A}$,

$$\left|\mathrm{Prob}[\mathsf{DLExp}^{\mathcal{A}}(1^k) = 1] - 1/2|\right| = \nu(k)$$

for some negligible function ν.

Note that Symmetric Decision Linear implies Symmetric Decision BDH and the Symmentric Decision Linear assumption has been used in [15].

6.3 Our construction

In this section we describe our construction for a selectively secure HVE scheme against *restricted* adversaries. It was presented in a work of the author with Giuseppe Persiano [27].

Setup. Procedure Setup, on input security parameter 1^k and attribute length $n = \mathsf{poly}(k)$, computes the public key Pk and the master secret key Msk in the following way.

Choose a random instance $\mathcal{I} = [p, \mathbb{G}, \mathbb{G}_T, g, \mathbf{e}]$.

Choose y at random in $\mathbb{Z}_p$ and set $Y = \mathbf{e}(g, g)^y$.

For $1 \le i \le n$, choose t_i, v_i, r_i, m_i at random in $\mathbb{Z}_p$ and set $T_i = g^{t_i}, V_i = g^{v_i}$ and $R_i = g^{r_i}, M_i = g^{m_i}$.

Then, $\mathsf{Setup}(1^k, n)$ returns $[\mathsf{Pk}, \mathsf{Msk}]$ where

$$\mathsf{Pk} = [\mathcal{I}, Y, (T_i, V_i, R_i, M_i)_{i=1}^n] \text{ and } \mathsf{Msk} = [y, (t_i, v_i, r_i, m_i)_{i=1}^n].$$

Encryption. Procedure $\mathsf{Encrypt}$ takes as input cleartext $M \in \mathbb{G}_T$, attribute string $\vec{x}$ and public key Pk and computes ciphertext as follows.

Choose s at random in $\mathbb{Z}_p$, and for $1 \le i \le n$, choose s_i at random in $\mathbb{Z}_p$ and compute ciphertext

$$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}, M) = [\Omega, C_0, (X_i, W_i)_{i=1}^n],$$

where $\Omega = M \cdot Y^s$, $C_0 = g^s$ and

$$X_i = \begin{cases} T_i^{s-s_i}, & \text{if } x_i = 1; \\ R_i^{s-s_i}, & \text{if } x_i = 0. \end{cases} \quad \text{and} \quad W_i = \begin{cases} V_i^{s_i}, & \text{if } x_i = 1; \\ M_i^{s_i}, & \text{if } x_i = 0. \end{cases}$$

Private-key generation. Procedure KeyGen on input Msk and $\vec{y} \in \{0, 1, \star\}^n$ derives private key $K_{\vec{y}}$ relative to attribute string $\vec{y}$ in the following way.

If $\vec{y} = (\star, \star, \ldots, \star)$ then $K_{\vec{y}} = g^y$. Else, for $1 \le i \le n$, choose a_i at random in $\mathbb{Z}_p$ under the constraint that $\sum_{i=1}^{i=n} a_i = y$ and let $K_{\vec{y}} = (Y_i, L_i)_{i=1}^n$, where

$$Y_i = \begin{cases} g^{\frac{a_i}{t_i}}, & \text{if } y_i = 1; \\ g^{\frac{a_i}{r_i}}, & \text{if } y_i = 0; \\ \emptyset, & \text{if } y_i = \star. \end{cases} \quad \text{and} \quad L_i = \begin{cases} g^{\frac{a_i}{v_i}}, & \text{if } y_i = 1; \\ g^{\frac{a_i}{m_i}}, & \text{if } y_i = 0; \\ \emptyset, & \text{if } y_i = \star. \end{cases}$$

Decryption. Procedure $\mathsf{Decrypt}$ decrypts cyphertext $\mathsf{Ct}_{\vec{x}}$ using secret key $K_{\vec{y}}$ such that $\mathsf{Match}(\vec{x}, \vec{y}) = 1$.

$$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}_{\vec{x}}, K_{\vec{y}}) = \Omega^{-1} \cdot \prod_{i \in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)\mathbf{e}(W_i, L_i)$$

where $S_{\vec{y}}$ is the set of indices i such that $y_i \neq \star$. If $S_{\vec{y}} = \emptyset$ then $K_{\vec{y}} = g^y$ and

$$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}_{\vec{x}}, K_{\vec{y}}) = \Omega^{-1} \cdot \mathbf{e}(C_0, K_{\vec{y}}).$$

This ends the description of our construction. We next prove that the quadruple is indeed an HVE.

Theorem 19. *The quadruple of algorithms* ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt}$) *specified above is an HVE.*

Proof. We verify that this procedure computes M correctly when $\mathsf{Match}(\vec{x}, \vec{y}) = 1$. The case in which $\vec{y} = (\star, \star, \cdots, \star)$ is obvious.

Denote with $S^1_{\vec{y}}$ (resp. $S^0_{\vec{y}}$) the (possibly empty) set of indices i such that $y_i = 1$ (resp $y_i = 0$). Then we have

$$\begin{aligned}
\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}_{\vec{x}}, K_{\vec{y}}) &= \Omega^{-1} \prod_{i \in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)\mathbf{e}(W_i, L_i) \\
&= M\mathbf{e}(g,g)^{-ys} \cdot \prod_{i \in S^1_{\vec{y}}} \mathbf{e}(g^{t_i(s-s_i)}, g^{\frac{a_i}{t_i}})\mathbf{e}(g^{w_i s_i}, g^{\frac{a_i}{w_i}}) \\
&\quad \cdot \prod_{i \in S^0_{\vec{y}}} \mathbf{e}(g^{r_i(s-s_i)}, g^{\frac{a_i}{r_i}})\mathbf{e}(g^{m_i s_i}, g^{\frac{a_i}{m_i}}) \\
&= M\mathbf{e}(g,g)^{-ys} \prod_{i \in S^1_{\vec{y}}} \mathbf{e}(g,g)^{(s-s_i)a_i}\mathbf{e}(g,g)^{s_i a_i} \\
&\quad \cdot \prod_{i \in S^0_{\vec{y}}} \mathbf{e}(g,g)^{(s-s_i)a_i}\mathbf{e}(g,g)^{s_i a_i} \\
&= M\mathbf{e}(g,g)^{-ys} \prod_{i \in S_{\vec{y}}} \mathbf{e}(g,g)^{(s-s_i)a_i}\mathbf{e}(g,g)^{s_i a_i} \\
&= M\mathbf{e}(grg)^{-ys} \prod_{i \in S_{\vec{y}}} \mathbf{e}(g,g)^{s a_i} \\
&= M\mathbf{e}(g,g)^{-ys}\mathbf{e}(g,g)^{ys} = M.
\end{aligned}$$

□

We omit the proof of correctness in the case that the predicate it is not satisfied but in Section 8.3 we will present a proof for another very similar scheme that will also imply a proof for this scheme.

6.4 Proofs

The security proof of our selectively secure scheme offers a concrete example of the partitioning paradigm. The simulator will know the challenge attribute vector so that it can partition the space of the keys in a such way that it can create each key but keys that correspond to predicates that satisfy the challenge attribute vector. Precisely, the proof is a sequence of games one for each position in the challenge ciphertext. For each one of them, the simulator embeds the challenge coming from the assumption in that specific position of the challenge ciphertext. Then, it has to guarantee that it can create keys for all vectors that do not match the challenge vector. When it receives a query for a vector, it finds a position where the challenge vector and this vector differ, and it uses this position to embed other values of the assumption to correctly simulate the key. The reader could have noticed that, following these lines, the simulator cannot create keys for vectors that match two different challenges. Indeed, we state and proof the security against restricted adversaries. Details will follow in the proof of attribute-hiding security. We start by giving a proof of payload-hiding security. We could to give a direct proof of attribute-hiding that would also imply a proof of payload-hiding. Anyway, the payload-hiding security would follow from a stronger assumption (Decision Linear instead than BDDH). Furthermore it is instructive to show this proof separately.

Theorem 20 (Proof of Payload-Hiding)**.** *Assume BDDH holds. Then HVE scheme* ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{KeyGen}, \mathsf{Decrypt}$) *described above is payload-hiding.*

Proof. Suppose that there exists PPT adversary $\mathcal{A}$ which has success in experiment $\mathsf{SemanticExp}$ with probability non-negligibly larger than 1/2. We then construct an adversary $\mathcal{B}$ for the experiment $\mathsf{DBDHExp}$. $\mathcal{B}$ takes in input $[\mathcal{I}, A = g^a, B = g^b, C = g^c, Z]$, where Z is $\mathbf{e}(g,g)^{abc}$ or a random element of $\mathcal{G}_T$.

Init. $\mathcal{B}$ receives from $\mathcal{A}$ the attribute string $\vec{x}$ it wishes to be challenged upon.

Setup. Set $Y = \mathbf{e}(A, B)$. For every $1 \le i \le n$, $\mathcal{B}$ chooses $t'_i, v'_i, r'_i, m'_i \in \mathbb{Z}_p$ at random and set

$$T_i = \begin{cases} g^{t'_i}, & \text{if } x_i = 1; \\ B^{t'_i}, & \text{if } x_i = 0; \end{cases} \quad \text{and} \quad V_i = \begin{cases} g^{v'_i}, & \text{if } x_i = 1; \\ B^{v'_i}, & \text{if } x_i = 0; \end{cases}$$

$$R_i = \begin{cases} B^{r'_i}, & \text{if } x_i = 1; \\ g^{r'_i}, & \text{if } x_i = 0; \end{cases} \quad \text{and} \quad M_i = \begin{cases} B^{m'_i}, & \text{if } x_i = 1; \\ g^{m'_i}, & \text{if } x_i = 0; \end{cases}$$

$\mathcal{B}$ runs $\mathcal{A}$ on input $\mathsf{Pk} = [\mathcal{I}, Y, (T_i, V_i, R_i, M_i)_{i=1}^n]$.

Notice that Pk has the same distribution of a public key received by $\mathcal{A}$ in the Setup phase of $\mathsf{SemanticExp}$ with $y = a \cdot b$, and with $t_i = t'_i$, $v_i = v'_i$, $r_i = br'_i$, and $m_i = bm'_i$ for i with $x_i = 1$, and $t_i = bt'_i$, $v_i = bv'_i$, $r_i = r'_i$, and $m_i = m'_i$ for i with $x_i = 0$.

Query Phase I. $\mathcal{B}$ answers $\mathcal{A}$'s queries for $\vec{y}$ such that $\mathsf{Match}(\vec{x},\vec{y}) = 0$ as follows.

Let j be an index where $x_j \neq y_j$ and $y_j \neq \star$ (there must exist at least one such index). For every $i \neq j$ such $y_i \neq \star$, choose a'_i at random in $\mathbb{Z}_p$ and let $a' = \sum a'_i$. Set Y_j and L_j as

$$Y_j = \begin{cases} A^{1/t'_j} g^{-a'/t'_j}, & \text{if } y_j = 1; \\ A^{1/r'_j} g^{-a'/r'_j}, & \text{if } y_j = 0. \end{cases} \quad \text{and} \quad L_j = \begin{cases} A^{1/v'_j} g^{-a'/v'_j}, & \text{if } y_j = 1; \\ A^{1/m'_j} g^{-a'/m'_j}, & \text{if } y_j = 0. \end{cases}$$

and, for $i \neq j$, set Y_i, L_i as follows

$$Y_i = \begin{cases} B^{a'_i/t'_i}; & \text{if } x_i = y_i = 1; \\ B^{a'_i/r'_i}; & \text{if } x_i = y_i = 0; \\ g^{a'_i/r'_i}; & \text{if } x_i = 1 \wedge y_i = 0; \\ g^{a'_i/t'_i}; & \text{if } x_i = 0 \wedge y_i = 1; \\ \emptyset; & \text{if } y_i = \star. \end{cases} \quad \text{and} \quad L_i = \begin{cases} B^{a'_i/v'_i}; & \text{if } x_i = y_i = 1; \\ B^{a'_i/m'_i}; & \text{if } x_i = y_i = 0; \\ g^{a'_i/m'_i}; & \text{if } x_i = 1 \wedge y_i = 0; \\ g^{a'_i/v'_i}; & \text{if } x_i = 0 \wedge y_i = 1; \\ \emptyset; & \text{if } y_i = \star. \end{cases}$$

Notice that $K_{\vec{y}}$ has the same distribution of the key returned by the KeyGen procedure. In fact, for all i such that $y_i = 1$, we have that $Y_i = g^{\frac{a_i}{t_i}}$ and $L_i = g^{\frac{a_i}{v_i}}$ and, for all i such that $y_i = 0$, we have that $Y_i = g^{\frac{a_i}{r_i}}$ and $L_i = g^{\frac{a_i}{m_i}}$ with $a_i = ba'_i$, for all $i \neq j$, and $a_j = b(a - a')$. Moreover, notice that $\sum_i a_i = y$.

Challenge. $\mathcal{A}$ returns two messages $M_0, M_1 \in \mathbb{G}_T$.

$\mathcal{B}$ chooses $\eta \in \{0,1\}$ at random and, for $i = 1, \cdots, n$, $s_i \in \mathbb{Z}_p$ and constructs $\mathsf{Ct}_{\vec{x}} = (\Omega, C, (X_i, W_i)_{i=1}^n)$, where $\Omega = M_\eta Z$, $C_0 = C$ and

$$X_i = \begin{cases} C^{t'_i} g^{-t'_i s_i}; & \text{if } x_i = 1; \\ C^{r'_i} g^{-r'_i s_i}; & \text{if } x_i = 0. \end{cases} \quad \text{and} \quad W_i = \begin{cases} g^{-v'_i s_i}; & \text{if } x_i = 1; \\ g^{-m'_i s_i}; & \text{if } x_i = 0. \end{cases}$$

Observe that if $Z = \mathbf{e}(g,g)^{abc}$ then $\mathsf{Ct}_{\vec{x}}$ is an encryption of M_η with $s = c$. If instead Z is random in $\mathbb{G}_T$ then $\mathsf{Ct}_{\vec{x}}$ is independent from η.

Query Phase II. Identical to Query Phase I.

Output. $\mathcal{A}$ outputs η'. $\mathcal{B}$ returns 0 iff $\eta' = \eta$.

To conclude the proof observe that, if $Z = \mathbf{e}(g,g)^{abc}$ then, since $\mathcal{A}$ is a successful adversary for payload-hiding security, the probability that $\mathcal{B}$ returns 0 is at least $1/2 + 1/\mathsf{poly}(k)$. On the other hand if Z is random in $\mathbb{G}_T$ the probability that $\mathcal{B}$ returns 0 is at most $1/2$. This contradicts the BDDH assumption. $\square$

To prove anonymity of the scheme we consider a sequence of distributions, where the first distribution is the distribution of a randomly computed ciphertext for a fixed message M and attribute string $\vec{x}$ and the last distribution is the random distribution on $\mathbb{G}_T \times \mathbb{G}^{2n+1}$.

More specifically, for $j = 0, 1, \ldots, n$, let $p_j^{\mathcal{A}}(\vec{x}) = \text{Prob}[\mathsf{Exp}_j^{\mathcal{A}}(\vec{x}) = 0]$, where $\mathsf{Exp}_j^{\mathcal{A}}$ is the following experiment.

```
$\mathsf{Exp}_j^{\mathcal{A}}(\vec{x})$

1. Choose $\mathcal{I} = [p, \mathbb{G}, g]$ with security parameter $1^k$;
2. $[\mathsf{Msk}, \mathsf{Pk}] \leftarrow \mathsf{Setup}(1^k, n)$;
3. Run $\mathcal{A}$ on input $\mathsf{Pk}$ and answer secret key queries for $\vec{y}$ such that $\mathsf{Match}(\vec{x}, \vec{y}) = 0$ by
   running $\mathsf{KeyGen}$ on input $\mathsf{Msk}$ and $\vec{y}$.
4. Pick $R_0 \leftarrow \mathbb{G}_T$, $s \leftarrow \mathbb{Z}_p$ and set $C_0 = g^s$.
5. For $i = 1$ to $j$
   (a) choose $X_i, W_i \leftarrow \mathbb{Z}_p$;
6. For $i = j+1$ to $n$
   (a) pick $s_i \leftarrow \mathbb{Z}_p$ and set
       $X_i = \begin{cases} T_i^{s-s_i}, & \text{if } x_i = 1; \\ R_i^{s-s_i}, & \text{if } x_i = 0. \end{cases}$ and $W_i = \begin{cases} V_i^{s_i}, & \text{if } x_i = 1; \\ M_i^{s_i}, & \text{if } x_i = 0. \end{cases}$
7. **return:** $\mathcal{A}(R_0, C_0, (X_i, W_i)_{i=1}^n)$;
```

We stress that in the previous experiment $\mathcal{A}$ is a restricted adversary. From the proof of payload-hiding security it follows, that under the BDDH, for all probabilistic poly-time adversaries $\mathcal{A}$, and for all message M the quantity

$$|p_0^{\mathcal{A}}(\vec{x}) - p^{\mathcal{A}}(M, \vec{x})|$$

is negligible where $p^{\mathcal{A}}(M, \vec{x})$ is the output of the experiment that is similar to $\mathsf{Exp}_0^{\mathcal{A}}(\vec{x})$ with the only exception that Ω is set equal to $Y \cdot M$. Next we will show that, under the DL assumption, for any probabilistic poly-time adversary $\mathcal{A}$ and for $\ell = 1, \cdots, n$, the quantity

$$|p_\ell^{\mathcal{A}}(\vec{x}) - p_{\ell-1}^{\mathcal{A}}(\vec{x})|$$

is negligible. This implies that

$$|p_0^{\mathcal{A}}(\vec{x}) - p_n^{\mathcal{A}}(\vec{x})|$$

is negligible.

Theorem 21 (Proof of Attribute-Hiding against Restricted Adversaries)**.** *Under the DL assumption, for $\ell = 1, 2, \ldots, n$ and for any $\vec{x} \in \{0,1\}^n$, we have that*

$$|p_\ell^{\mathcal{A}}(\vec{x}) - p_{\ell-1}^{\mathcal{A}}(\vec{x})| < \nu(n)$$

Proof. Suppose that there exists PPT adversary $\mathcal{A}$ which distinguishes the tuple of the experiment of $\mathsf{Exp}_{\ell-1}$ from the tuple of the experiment Exp_ℓ with probability $1/2 + 1/\mathsf{poly}(k)$. We then construct an adversary $\mathcal{B}$ for the experiment DLExp. $\mathcal{B}$ takes in input $[\mathcal{I}, Z_1 = g^{z_1}, Z_2 = g^{z_2}, Z_{13} = g^{z_1 z_3}, U = g^u, Z]$, where Z is $g^{z_2(u-z_3)}$ or a random element of $\mathcal{G}$.

Init. $\mathcal{B}$ receives from $\mathcal{A}$ the attribute string $\vec{x}$ it wishes to be challenged upon.

Setup. $\mathcal{B}$ sets $Y = \mathbf{e}(Z_1, Z_2)$. For every $1 \le i \le n$, $\mathcal{B}$ chooses $t'_i, v'_i, r'_i, m'_i \in \mathbb{Z}_p$ at random and sets

$$T_\ell = \begin{cases} Z_2^{t'_\ell}, & \text{if } x_\ell = 1; \\ Z_1^{t'_\ell}, & \text{if } x_\ell = 0; \end{cases} \quad \text{and} \quad V_k = \begin{cases} Z_1^{v'_\ell}, & \text{if } x_\ell = 1; \\ Z_1^{v'_\ell}, & \text{if } x_\ell = 0; \end{cases}$$

$$R_\ell = \begin{cases} Z_1^{r'_\ell}, & \text{if } x_\ell = 1; \\ Z_2^{r'_\ell}, & \text{if } x_\ell = 0; \end{cases} \quad \text{and} \quad M_\ell = \begin{cases} Z_1^{m'_\ell}, & \text{if } x_\ell = 1; \\ Z_1^{m'_\ell}, & \text{if } x_\ell = 0; \end{cases}$$

And for $i \neq \ell$, $\mathcal{B}$ sets

$$T_i = \begin{cases} g^{t'_i}, & \text{if } x_i = 1; \\ Z_1^{t'_i}, & \text{if } x_i = 0; \end{cases} \quad \text{and} \quad V_i = \begin{cases} g^{v'_i}, & \text{if } x_i = 1; \\ Z_1^{v'_i}, & \text{if } x_i = 0; \end{cases}$$

$$R_i = \begin{cases} Z_1^{r'_i}, & \text{if } x_i = 1; \\ g^{r'_i}, & \text{if } x_i = 0; \end{cases} \quad \text{and} \quad M_i = \begin{cases} Z_1^{m'_i}, & \text{if } x_i = 1; \\ g^{m'_i}, & \text{if } x_i = 0; \end{cases}$$

Give $\mathsf{Pk} = [\mathcal{I}, Y, (T_i, V_i, R_i, M_i)_{i=1}^n]$ to $\mathcal{A}$. Notice that Pk has the same distribution of a public key received by $\mathcal{A}$ at step 3 of Exp_ℓ^A and $\mathsf{Exp}_{\ell-1}^A$ with $y = z_1 \cdot z_2$, and with $t_i = t'_i, v_i = v'_i, r_i = z_1 r'_i, m_i = z_1 m'_i$ for $i \neq \ell$ with $x_i = 1$, and $t_i = z_1 t'_i, v_i = z_1 v'_i, r_i = r'_i, m_i = m'_i$ for $i \neq$ with $x_i = 0$, and $t_\ell = z_2 t'_\ell, v_\ell = z_1 v'_\ell, r_\ell = z_1 r'_\ell, m_\ell = z_1 m'_\ell$ for ℓ with $x_\ell = 1$, and $t_k = z_1 t'_\ell, v_\ell = z_1 v'_\ell, r_\ell = z_2 r'_\ell, m_\ell = z_1 m'_\ell$ for ℓ with $x_\ell = 0$.

Query Phase I. $\mathcal{B}$ answers $\mathcal{A}$'s queries for $\vec{y}$ such that $\mathsf{Match}(\vec{x},\vec{y}) = 0$. We distinguish two cases.

Case 1: $x_\ell = y_\ell$. In this case there exists index j such that $y_j \neq x_j$ and $y_j \neq \star$.

Then, for $i \neq j$ choose at random $a'_i \in \mathbb{Z}_p$ and denote by a' the sum $a' = \sum_{i\neq j,\ell} a'_i$.

For $i \neq j, \ell$ $\mathcal{B}$ sets

$$Y_i = \begin{cases} Z_1^{a'_i/t'_i}, & \text{if } x_i = y_i = 1; \\ Z_1^{a'_i/r'_i}, & \text{if } x_i = y_i = 0; \\ g^{a'_i/r'_i}, & \text{if } x_i = 1, y_i = 0; \\ g^{a'_i/t'_i}, & \text{if } x_i = 0, y_i = 1; \\ \emptyset, & \text{if } y_i = \star. \end{cases} \quad \text{and} \quad L_i = \begin{cases} Z_1^{a'_i/v'_i}, & \text{if } x_i = y_i = 1; \\ Z_1^{a'_i/m'_i}, & \text{if } x_i = y_i = 0; \\ g^{a'_i/m'_i}, & \text{if } x_i = 1, y_i = 0; \\ g^{a'_i/v'_i}, & \text{if } x_i = 0, y_i = 1; \\ \emptyset, & \text{if } y_i = \star. \end{cases}$$

$\mathcal{B}$ sets

$$Y_\ell = \begin{cases} Z_1^{a'_\ell/t'_\ell}, & \text{if } y_\ell = 1; \\ Z_1^{a'_\ell/r'_\ell}, & \text{if } y_\ell = 0; \\ \emptyset, & \text{if } y_\ell = \star. \end{cases} \quad \text{and} \quad L_\ell = \begin{cases} Z_2^{a'_\ell/v'_\ell}, & \text{if } y_\ell = 1; \\ Z_2^{a'_\ell/m'_\ell}, & \text{if } y_\ell = 0; \\ \emptyset, & \text{if } y_\ell = \star. \end{cases}$$

Finally $\mathcal{B}$ sets

$$Y_j = \begin{cases} Z_2^{(1-a'_\ell)/t'_j} g^{-a'/t'_j}, & \text{if } y_j = 1; \\ Z_2^{(1-a'_\ell)/r'_j} g^{-a'/r'_j}, & \text{if } y_j = 0. \end{cases} \quad \text{and} \quad L_j = \begin{cases} Z_2^{(1-a'_\ell)/v'_j} g^{-a'/v'_j}, & \text{if } y_j = 1; \\ Z_2^{(1-a'_\ell)/m'_j} g^{-a'/m'_j}, & \text{if } y_j = 0. \end{cases}$$

By the settings above we have that, for $i \neq j, \ell$ $a_i = z_1 a'_i$, $a_\ell = z_1 z_2 a'_\ell$ and $a_j = z_1 z_2 - z_1 z_2 a'_\ell - z_1 a'$. Therefore, the a_i's are independently and randomly chosen in $\mathbb{Z}_p$ under the costraint that $\sum_i a_i = z_1 z_2 = y$.

Case 2: $x_\ell \neq y_\ell$.

Let $a' = \sum_{i\neq\ell} a'_i$ and set

$$Y_\ell = \begin{cases} Z_2^{1/r'_\ell} g^{-a'/r'_\ell}, & \text{if } x_\ell = 1; \\ Z_2^{1/t'_\ell} g^{-a'/t'_\ell}, & \text{if } x_\ell = 0; \end{cases} \quad \text{and} \quad L_\ell = \begin{cases} Z_2^{1/m'_\ell} g^{-a'/m'_\ell}, & \text{if } x_\ell = 1; \\ Z_2^{1/v'_\ell} g^{-a'/v'_\ell}, & \text{if } x_\ell = 0; \end{cases}$$

For $i \neq \ell$ set Y_i, L_i exactly like in the previous case.

By the settings above we have that $a_i = z_1 a'_i$, $a_\ell = z_1 z_2 - z_1 a'$. Therefore, the a_i's are independently and randomly chosen in $\mathbb{Z}_p$ under the costraint that $\sum_i a_i = z_1 z_2 = y$.

Challenge. $\mathcal{B}$ chooses $R_0 \in \mathbb{G}_T$ at random and for $1 \leq i \leq n$ choose $s'_i \in \mathbb{Z}_p, R_i \in \mathbb{G}_T$ at random and construct the tuple

$$D^* = (R_0, C_0 = U, (X_i, W_i)_{i=1}^n)$$

where X_i, W_i are

$$X_i = \begin{cases} R_i, & \text{if } i \le \ell; \\ U^{t'_i} g^{-t'_i s'_i}, & \text{if } i > \ell, x_i = 1; \\ U^{r'_i} g^{-r'_i s'_i}, & \text{if } i > \ell, x_i = 0; \\ Z, & \text{if } i = \ell; \end{cases} \quad \text{and} \quad W_i = \begin{cases} g^{v'_i s'_i}, & \text{if } i \le \ell, x_i = 1; \\ g^{m'_i s'_i}, & \text{if } i \le \ell, x_i = 0; \\ g^{-v'_i s'_i}, & \text{if } i > \ell, x_i = 1; \\ g^{-m'_i s'_i}, & \text{if } i > \ell, x_i = 0; \\ Z_{13}, & \text{if } i = \ell; \end{cases}$$

If in the experiment of Decision-Linear $Z = g^{z_2(u-z_3)}$ the tuple D^* has the same distribution as the tuple of the experiment $\mathsf{Exp}^A_{\ell-1}$, whereas if Z is random it has the same distribution as the tuple of the experiment Exp^A_ℓ. In fact this is seen for $s = u, s_\ell = z_3, s_i = s'_i, i \neq \ell$.

Query Phase II. Identical to Query Phase I.

Output. $\mathcal{A}$ outputs v which represents a guess for the tuple in input ($v = 0$ for $D_{\ell-1}$ and $v = 1$ for D_ℓ). $\mathcal{B}$ forwards the same bit as its guess for the tuple of the experiment DLExp.

By the settings above this concludes the proof.

□

Theorem 22. *The scheme of Section 6.3 is selectively secure against restricted adversaries.*

Proof. It follows by theorems 20 and 21. □

Chapter 7

Dual System Encryption Methodology

In this chapter we present a fundamental tool to prove the security of our schemes, the *Dual System Encryption* technique introduced by [40]. First Predicate Encryption Systems for HVE and inner-product [14, 28] were proved secure in the selective-id model where the attacker must declare the challenge attributes before seeing the public-key. This model of security was proposed by Canetti, Halevi and Katz [16] that motivated it by showing that selective-id secure IBE schemes imply CCA secure public-key encryption. Usually, in the selective-id model the proofs of security are simpler because the simulator can build the PK basing on the id selected by the adversary. Anyway, it is a well known fact that selective-id security is a weaker notion than full security. Building fully secure schemes in the standard model can also be a complicated task for IBE.

First implementations of fully secure IBE in the standard model were given by Boneh and Boyen [6] and Waters [39]. The latter was efficient and proved secure by the well established assumption Decisional Bilinear Diffie-Hellman. These systems as well previous selective-id implementations of IBE use a partitioning techniquey to prove the security. The partitioning technique, roughly, proceeds by dividing the space of the identities in two classes: identities for which the simulator can answer key queries; and identities that it can use to answer the challenge query. In the standard model,y the simulator programs the public-key so that the assumption (that we reduce the security of the scheme to) can be embedded favorably to perform the simulation. In the random oracle model the partitinong is programmed in the random oracle.

The drawback of this strategy is that it does not seem useful in proving the security of more advanced primitives like ABE, HVE, or also HIBE. Indeed the ABE and HVE systems that use the partitioning technique were only proved secure in the selective-id model. The implementation of HIBE systems which use the partitioning strategy suffer from an exponential degradation in the hierarchy. Furthermore also the efficient IBE systems that fall in this category had large public-key parameters. Gentry [22] proposed a new approach achieving an Anonymous IBE system with short public-key that had a security proof that departed from the partitioning technique. Gentry's proof uses a q-based assumption, that is an assumption whose size depends from the number of queries requested from the adversary. It makes this by embedding a degree q polynomial F in the reduction and by attaching a tag $F(I)$ to the query for identity I.

In this reduction, the simulator can create both queries and challenge for each identity. This seems a paradox because the simulator could break the assumption by decrypting a challenge ciphertext with the corresponding key. The simulation overcomes this problem by constructing the challenge ciphertext in a way such that it always decrypts when the decryption key is for the same identity of the challenge ciphertext. Gentry and Halevi [23] showed how to extend these ideas to obtain the first HIBE systems whose security does not degrade exponentially with the number of levels. The system of Gentry and Halevi still relies on q-based complexity assumptions and the proofs are quite complex. Katz and Wang and subsequently Goh, Jarecki, Katz and Wang [24] presented a different approach to prove the full security of IBE systems that do not fall in the partitioning strategy. Their technique uses a two-key approach but it is based on the random oracle. The advantage of this technique is that it achieves *tight* security reductions whereas prior works as well as any other IBE implmentation in the standard model has a loose reduction from some computational assumption (for example, the losing factor of the reduction is the number of queries issued by the adversary).

To overcome these difficulties Waters [40] introduced the Dual System Encryption (DSE) methodology. The DSE consists in proving the indistinguishability of a series of games. In each game both keys and challenge ciphertext can take one of the forms: normal form and *semi-functional* form. A semi-functional key can decrypt a normal cipherthext and a semi-functional ciphertex can decrypt a normal key but a semi-functional key cannot decrypt a semi-functional ciphertext. In the first game, that is be the real security game, the challenge ciphertext and all keys are normal. Next, the challenge ciphertext is changed to a semi-functional one. No adversary can distinguish this game (under some complexity assumption) since the keys are normal. Then, in the next series of games, the keys are changed one game at a time from normal to semi-functional, again arguing indistinguishability. Here, it is used the fact that the challenge key is for an identity different from the identity used to encrypt the challenge ciphertext (or more generally, for PE systems the key is a for a predicate that does not satisfy the challenge attribute). Hence, the adversary cannot distinguish the normal key for identity different from the challenge identity from a semi-functional key for the same identity. Finally, when the all keys and the challenge ciphertext are semi-functional, proving security is straightforward since no key is useful to decrypt.

Notice that the indistinguishability of normal keys from semi-functional keys gives rise to an issue. The reduction algorithm could break the computational assumption used to prove the security in this way. It creates a semi-functional ciphertext for identity I and test if challenge query for the same identity I is normal or semi-functional by decrypting with it. Indeed, if it is semi-functional the decryption should not work whereas if it is normal the decryption works. This paradoxical situation can be avoided in different ways. Waters overcomes these difficulties by embedding tags in both ciphertexts and keys. The decryption works if and only if the tags of the key is different from the tag of the ciphertext. By setting the tag to be a function of the identity the potential paradox is avoided. Lewko and Waters [31] use a different solution. In their implementation of DSE keys and ciphertexts can also be *nominally* semi-functional. A nominally semi-functional ciphertext for identity I can decrypt a nominally semi-functional key for the same identity I. Therefore, if the reduction algorithm tries to create a semi-functional ciphertext for the challenge identity it can only generate one that is nominally semi-functional when used along with the challenge key. Thus, it is unable to detect the form of the key because decryption works in both the cases. In our implementation of fully secure HVE, we use a *all-but-one* approach guaranteeing that the simulator misses some group elements that it would need to create a challenge ciphertext for the same attribute of the challenge key. In fact, by adding this missing element to the assumption (that we reduce the indistinguishability to) the assumption would turn to be false. Our approach can be viewed as a mix of DSE and a partitioning paradigm. This is possible due to the way how DSE achieves security. Indeed, the powerful of DSE relies on the fact that, by changing the keys one at time, we only need the relationship between the challenge ciphertext and a *single* key at a time, so that the partitioning space is small.

To prove the security against unrestricted adversary (see Chapter 12) we also need to worry about the fact that keys whose predicate satisfies *both* the challenge attributes cannot be simulated with a semi-functional form since the adversary could discover the nature of the challenge key by trying to decrypt it. We solve the problem by identifying a property that this kind of keys enjoys and so simulating this *matching* keys with a normal form. This is achieved by using $\ell \cdot q$ games instead of simply q games. Indeed, for each of the ℓ games on the positions we make q games on the keys. In each of this q game for a fixed position we consider to be normal the key that have $\star$ in that position. In the last of this q games all keys are semi-functional but the *matching* keys but this is sufficient to change the distribution (information-theoretically in our implementation) of the challenge ciphertext. Then we turn all keys to normal and continue the process for the next position. Details are given in Chapter 12.

Chapter 8

Constructing 0 and 1-secure HVE

In this chapter we will describe our constructions for a 0-secure and 1-secure HVE schemes. Though the construction of Chapter 12 is secure against unrestricted adversaries, the constructions of this Chapter can be instructive to understand the difficulties and the challenges found to reach the main result. Furthermore the proof of security for this scheme is more tight than that for the scheme secure against restricted adversaries. We first present the assumptions used in the proofs of security.

8.1 Complexity Assumptions for the 0-secure and 1-secure constructions

We now present the complexity assumption used to prove the 0-security of our construction.

The first assumption is a subgroup-decision type assumption for bilinear settings. Specifically, Assumption 1 posits the difficulty of deciding whether an element belongs to one of two specified subgroups, even when generators of some of the subgroups of the bilinear group are given. More formally, we have the following definition.

First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_3 \leftarrow \mathbb{G}_{p_3},\ A_{13} \leftarrow \mathbb{G}_{p_1p_3},\ A_{12} \leftarrow \mathbb{G}_{p_1p_2},\ A_4 \leftarrow\in \mathbb{G}_{p_4},\ T_1 \leftarrow \mathbb{G}_{p_1p_3},\ T_2 \leftarrow \mathbb{G}_{p_2p_3}$, and set $D = (\mathcal{I}, A_3, A_4, A_{13}, A_{12})$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be

$$\mathsf{Adv}_1^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

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

Our second assumption can be seen as the Decision Diffie-Hellman Assumption for composite order groups. Specifically, Assumption 2 posits the difficult of deciding if a triple of elements constitute a Diffie-Hellman triplet with respect to one of the factors of the order of the group, even when given, for each prime divisor p of the group order, a generator of the subgroup of order p. Notice that for bilinear groups of prime order the Diffie-Hellman assumption does not hold. More formally, we have the following definition.

First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4 \leftarrow \mathbb{G}_{p_4}, \alpha, \beta \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1p_4}$, and set $T_1 = A_1^{\alpha\beta} \cdot D_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^\alpha \cdot B_4, A_1^\beta \cdot C_4)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be

$$\mathsf{Adv}_2^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

Assumption 2. *We say that Assumption* 2 *holds for generator* $\mathcal{G}$ *if for all probabilistic polynomial-time algorithms* $\mathcal{A}$, $\mathsf{Adv}_2^{\mathcal{A}}(\lambda)$ *is a negligible function of* λ.

Our final assumption is again a subgroup-decision type of assumption. It will be used in the proofs of our 1-secure HVE scheme. It is defined as follows. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_2 \leftarrow \mathbb{G}_{p_2},\ A_3 \leftarrow \mathbb{G}_{p_3},\ A_4, B_4, \leftarrow \mathbb{G}_{p_4},\ A_{14}, B_{14} \leftarrow \mathbb{G}_{p_1p_4}$ and set $T_1 = B_{14}$, $T_2 = B_4$ and $D = (\mathcal{I}, A_2, A_3, A_4, A_{14})$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be

$$\mathsf{Adv}_4^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

Assumption 4 We say that Assumption 4 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_4^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

In Appendix B, we prove that Assumption 1 and 2 hold in the generic group model.

8.2 0-Secure HVE

We start by presenting our 0-secure scheme. We stress that in Chapter 12 this scheme will also be proved secure against unrestricted adversaries.

To make our descriptions 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 chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4,\ \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses 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 parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: 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 the tuple $(Y_i)_{i \in S_{\vec{y}}}$. Here we use the fact that $S_{\vec{y}}$ has size at least 2.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: 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 tuple $(X_i)_{i \in [\ell]}$.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: The test algorithm computes $T = \prod_{i \in S_{\vec{y}}} \mathbf{e}(X_i, Y_i)$. It returns TRUE if $T = 1$, FALSE otherwise.

Correctness

8.3 Correctness of our HVE scheme

Lemma 23. *The HVE scheme presented in Sections 12.3 and 8.2 is correct.*

PROOF. Suppose that $\mathsf{Ct} = (X_i)_{i \in [\ell]}$ is a ciphertext for attribute vector $\vec{x}$ and that $\mathsf{Sk}_{\vec{y}} = (Y_i)_{i \in S_{\vec{y}}}$ is a key for vector $\vec{y}$ such that $\mathsf{Match}(\vec{x}, \vec{y}) = 1$. We show that $\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$ returns TRUE. By definition we have that $\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$ computes the following product

$$\begin{aligned}
\prod_{i \in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) &= \prod_{i \in S_{\vec{y}}} \mathbf{e}(T_{i,x_i}^s \cdot Z_i, g_{12}^{a_i/t_{i,y_i}} \cdot W_i) && (8.1)\\
&= \prod_{i \in S_{\vec{y}}} \mathbf{e}(g_1^{t_{i,x_i} s}, g_1^{a_i/t_{i,y_i}}) && (8.2)\\
&= \prod_{i \in S_{\vec{y}}} \mathbf{e}(g_1^{t_{i,x_i} s}, g_1^{a_i/t_{i,x_i}}) && (8.3)\\
&= \prod_{i \in S_{\vec{y}}} \mathbf{e}(g_1, g_1)^{a_i s} && (8.4)\\
&= \mathbf{e}(g_1, g_1)^{s \sum_{i \in S_{\vec{y}}} a_i} && (8.5)\\
&= 1. && (8.6)
\end{aligned}$$

Equation 8.1 follows by definition of the X_i's and Y_i's. Equation 8.2 follows from Equation 8.1 by definition of the T_{i,x_i} and by the ortogonality property of bilinear groups of composite order discussed in Section 5.2. Equation 8.3 follows from Equation 8.2 since, if $\mathsf{Match}(\vec{x}, \vec{y}) = 1$, then, for each $i \in S_{\vec{y}}$, $x_i = y_i$. Equation 8.4 follows from Equation 8.3

by the bilinear property of $\mathbf{e}$. Equation 8.5 follows from Equation 8.4 by simple algebraic manipulations. Equation 8.6 follows from Equation 8.5 since by construction we have $\sum_{i\in S_{\vec{y}}} a_i = 0$. Finally, notice that if the value obtained by this computation is 1 (the identity of the target group) then, by definition, the Test procedure returns TRUE as expected. Suppose now that $\mathsf{Match}(\vec{x},\vec{y}) = 0$. Let A be the set of positions i such that $x_i = y_i$ and $B = S_{\vec{y}} \setminus A$. By definition we have that $\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$ computes the following product

$$\prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) \quad = \quad \mathbf{e}(g_1, g_1)^{s\sum_{i\in A} a_i} \cdot \mathbf{e}(g_1, g_1)^{s\sum_{i\in B} a_i w_i} \tag{8.7}$$

Equation 8.7 follows by some of the facts already observed in the discussion of Equations 8.1-8.5, by definition of A and B, and by setting $w_i \stackrel{\text{def}}{=} t_{i,x_i}/t_{i,y_i}$. Since that with very high probability $s \neq 0 \mod \mathbb{Z}_N$ then, by the fact that $\mathbf{e}$ is non-degenerate, such a product equals 1 (the unity of the target group) if and only if

$$\sum_{i\in A} a_i + \sum_{i\in B} a_i w_i = 0. \tag{8.8}$$

By recalling that $\sum_{i\in S_{\vec{y}}} a_i = 0$ and by the fact that A and B are a partition of $S_{\vec{y}}$, from Equation 8.8 follows that

$$\sum_{i\in S_{\vec{y}}} a_i + \sum_{i\in B} a_i(w_i - 1) = \sum_{i\in B} a_i(w_i - 1). \tag{8.9}$$

If $i \in B$, then $x_i \neq y_i, y_i \neq \star$, and thus, without loss of generality, $w_i = t_{i,0}/t_{i,1}$ is distributed randomly in $\mathbb{Z}_N$ with very high probability since that $t_{i,0}$ and $t_{i,1}$ are chosen randomly and independently from $\mathbb{Z}_N$. Hence, since that the set B is of cardinality exponentially smaller than N, with very high probability for each $i \in B$, $w_i - 1$ is distributed independently and randomly in $\mathbb{Z}_N$. From the fact that $\mathsf{Match}(\vec{x},\vec{y}) = 0$ it follows that the set B is non-empty and so, from the fact that the $w_i - 1$'s are distributed randomly and independently in $\mathbb{Z}_N$, it follows that Equation 8.9 is different from $0 \mod N$ with very high probability. Thus, with very high probability, the product computed by the Test procedure is different from the identity of the target group and so with the same probability it returns FALSE as expected. □

We stress that this proof also implies an analogous proof of correctness for the scheme of Section 6.3.

8.4 0-Security of our HVE scheme

In this section we prove that our HVE scheme is 0-secure. To prove security we rely on Assumptions 1 and 2. For a probabilistic polynomial-time 0-adversary $\mathcal{A}$ which makes q queries for KeyGen, our proof of security will be structured as a sequence of $q + 2$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$. The first game, GReal, is the real HVE security game described in the previous section. The remaining games, called $\mathsf{G}_0, \ldots, \mathsf{G}_q$, are described (and shown indistinguishable) in the following sections. In the rest of this section, when we refer to adversaries we mean 0-adversaries and when we refer to GReal we mean $\mathsf{GReal}(0)$.

Description of G_0. G_0 is like GReal, except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters Pk given to $\mathcal{A}$. Specifically,

Setup. $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ 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 sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ interacts with $\mathcal{A}$ on input Pk.

Key Query Answering. On vector $\vec{y}$, $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Construction. $\mathcal{C}$ picks one of the two challenge vectors provided by $\mathcal{A}$ and encrypts it with respect to public parameters Pk'.

Proof of indistinguishability of GReal and G_0

Lemma 24. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{GReal}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_0} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

Proof. We show a PPT algorithm $\mathcal{B}$ which receives $(\mathcal{I}, A_3, A_4, A_{13}, A_{12})$ and T and, depending on the nature of T, simulates GReal or G_0 with $\mathcal{A}$. This suffices to prove the Lemma.

Setup. $\mathcal{B}$ starts by constructing Pk and Pk' as follow. $\mathcal{B}$ sets $g_3 = A_3, g_{12} = A_{12}, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets $T_{i,b} = T^{t_{i,b}}$ and $T'_{i,b} = A_{13}^{t_{i,b}}$. Then $\mathcal{B}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. Whenever $\mathcal{A}$ asks to see the secret key $\mathsf{Sk}_{\vec{y}}$ associated with vector $\vec{y}$, $\mathcal{B}$ runs algorithm KeyGen on input Msk and $\vec{y}$.

Challenge Construction. The challenge is created by $\mathcal{B}$ by picking one of the two vectors provided by $\mathcal{A}$, let us call it $\vec{x}$, and by encrypting it by running the $\mathsf{Encrypt}$ algorithm on input $\vec{x}$ and Pk'.

This concludes the description of algorithm $\mathcal{B}$.

Now suppose $T \in \mathbb{G}_{p_1p_3}$, and thus it can be written as $T = h_1 \cdot h_3$ for $h_1 \in \mathbb{G}_{p_1}$ and $h_3 \in \mathbb{G}_{p_3}$. This implies that Pk received in input by $\mathcal{A}$ in the interaction with $\mathcal{B}$ has the same distribution as in GReal. Furthermore, it's easy to see that the answers to key queries and the challenge ciphertext given by $\mathcal{B}$ to $\mathcal{A}$ have the same distribution as the answers and the challenge ciphertext received by $\mathcal{A}$ in GReal. We can thus conclude that $\mathcal{C}$ has simulated GReal with $\mathcal{A}$.

Instead, when $T \in \mathbb{G}_{p_2 p_3}$, Pk provided by $\mathcal{B}$ has the same distribution as that produced by $\mathcal{C}$ in G_0. Therefore, $\mathcal{C}$ is simulating G_0 for $\mathcal{A}$. □

Description of G_k, for $1 \leq k \leq q$. Each of these games is like G_0, except that the first k key queries issued by $\mathcal{A}$ are answered with keys whose $\mathbb{G}_{p_1}$ parts are random. The remaining key queries (that is, from the $(k+1)$-st to the q-th) are answered like in the previous game. The $\mathbb{G}_{p_2}$ parts of all the answers to key queries are like in G_0. More precisely, in G_k, the Setup phase and the Challenge Construction are like in G_0 and the Key Query phase is the following.

Key Query Answering. $\mathcal{C}$ answers the first k key queries in the following way. On input vector $\vec{y}$, for $i \in S_{\vec{y}}$, $\mathcal{C}$ chooses random $a_i, c_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_{\vec{y}}} a_i = 0$ and random $W_i \in \mathbb{G}_{p_4}$. $\mathcal{C}$ sets, for $i \in S_{\vec{y}}$, $Y_i = g_1^{c_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i$. The remaining $q-k$ queries are answered like in G_0.

Proof of indistinguishability of G_{k-1} and G_k

Lemma 25. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_k} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/(2\ell)$ in breaking Assumption* 2.

Proof. $\mathcal{B}$ receives $(\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4)$ and T and, depending on the nature of T, simulates G_{k-1} or G_k with $\mathcal{A}$.

$\mathcal{B}$ starts by guessing the index j such that the j-th bit $y_j^{(k)}$ of the k-th query $\vec{y}^{(k)}$ is different from $\star$ and different from the j-th bit x_j of the challenge vectors provided by $\mathcal{A}$ that $\mathcal{C}$ uses to construct the challenge ciphertext. Notice that the probability that $\mathcal{B}$ correctly guesses j and $y_j^{(k)}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. Notice that, if during the simulation this is not the case, then $\mathcal{B}$ aborts the simulation and fails. We next describe and prove the correctness of the simulation under the assumption that $\mathcal{B}$'s initial guess is correct. Notice that if the initial guess is correct x_j and $y_j^{(k)}$ are uniquely determined and it holds that $x_j = 1 - y_j^{(k)}$.

Setup. $\mathcal{B}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$ and $g_{12} = A_1 \cdot A_2$. For each $i \in [\ell] \setminus \{j\}$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Moreover, $\mathcal{B}$ chooses random $t_{j,x_j} \in Z_N, R_{j,x_j} \in \mathbb{G}_{p_3}$, $r_{j,y_j^{(k)}} \in Z_N$ and $R_{j,y_j^{(k)}} \in \mathbb{G}_{p_3}$ and sets

$$T_{j,x_j} = g_2^{t_{j,x_j}} \cdot R_{j,x_j} \qquad T_{j,y_j^{(k)}} = g_2^{r_{j,y_j^{(k)}}} \cdot R_{j,y_j^{(k)}}.$$

Notice that by assumption $x_j \neq y_j^{(k)}$. $\mathcal{B}$ then sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i \in [\ell], b \in \{0,1\}}]$. In addition, for each $i \in [\ell] \setminus \{j\}$ and $b \in \{0,1\}$ and $\mathcal{B}$ chooses random $R'_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R'_{i,b}$. Moreover $\mathcal{B}$ chooses random R_{j,x_j} and sets $T'_{j,x_j} = g_1^{t_{x,x_j}} \cdot R'_{j,x_j}$. The value of $T'_{j,y_j^{(k)}}$ remains unspecified. As we shall see below, in answering key queries, $\mathcal{B}$ will implicitly set

$T'_{j,y_j^{(k)}} = g_1^{1/\beta} \cdot R'_{j,y_j^{(k)}}$ for a random $R'_{j,y_j^{(k)}} \in \mathbb{G}_{p_3}$. $\mathcal{B}$ starts the interaction with $\mathcal{A}$ on input Pk. Notice that Pk has the same distribution as that seen by $\mathcal{A}$ in G_{k-1} and G_k.

Key Query Answering. For the first $k-1$ queries $\mathcal{B}$ proceeds as follows. Let $\vec{y}$ be the input vector. For $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random a_i such that $\sum_{i \in S_{\vec{y}}} a_i = 0$, random $z_i \in \mathbb{Z}_N$, and random $W_i \in \mathbb{G}_{p_4}$. Then, for $i \in S_{\vec{y}}$, $\mathcal{B}$ computes

$$Y_i = \begin{cases} g_1^{z_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i, & \text{if } i \neq j; \\ g_1^{z_j} \cdot g_2^{a_j/t_{j,y_j}} \cdot W_j, & \text{if } i = j \text{ and } y_j = x_j; \\ g_1^{z_j} \cdot g_2^{a_j/r_{j,y_j}} \cdot W_j, & \text{if } i = j \text{ and } y_j \neq \star. \end{cases}$$

Also notice that the first $k-1$ answers produced by $\mathcal{B}$ have the same distribution as the first $k-1$ answers seen by $\mathcal{A}$ in G_{k-1} and G_k.

Let us now describe how $\mathcal{B}$ answers the k-th query the vector $\vec{y}^{(k)}$. Let h be an index such that $h \neq j$ and $y_h^{(k)} \neq \star$; such an index always exists by our assumption that all keys are for vectors with at least two entries different from $\star$. Also we remind the reader that $y_j^{(k)} = 1 - x_j$.

Let $S = S_{\vec{y}} \setminus \{j,h\}$. For each $i \in S$, $\mathcal{B}$ chooses random $a_i \in \mathbb{Z}_N$ and $W_i \in \mathbb{G}_{p_4}$. Moreover $\mathcal{B}$ chooses random $a'_j \in \mathbb{Z}_N$ and $W_j, W_h \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_{12}^{a_i/t_{i,y_i^{(k)}}} \cdot W_i, \quad Y_j = T \cdot g_2^{a'_j/r_{j,y_j^{(k)}}} \cdot W_j,$$

$$Y_h = (A_1^{\alpha} B_4)^{-1/t_{h,y_h^{(k)}}} \cdot g_1^{-s/t_{h,y_h^{(k)}}} \cdot g_2^{-(s+a_j)/t_{h,y_h^{(k)}}} \cdot W_h,$$

where $s = \sum_{i \in S} a_i$. This terminates the description of how $\mathcal{B}$ handles the k-th key query. Let us now verify that when $T = A_1^{\alpha\beta} \cdot D_4$ then $\mathcal{B}$'s answer to the k-th key query is like in G_{k-1}. By our settings, we have that $Y_j = g_1^{\alpha/t'_{j,y_j^{(k)}}} \cdot g_2^{a'_j/r_{j,y_j^{(k)}}} \cdot D_4 \cdot W_j$ with $t'_{j,y_j^{(k)}} = 1/\beta$. By the Chinese Remainder Theorem, we can conclude that the answer to the k-th query of $\mathcal{A}$ is distributed as in G_{k-1}. Instead, if T is random in $\mathbb{G}_{p_1 p_4}$ then the $\mathbb{G}_{p_1}$ parts of the Y_i's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in G_k.

For the l-th key queries for $l = k+1, \ldots, q$, notice that if the j-th bit of the l-th query vector is equal to x_j then $\mathcal{B}$ has all the t_{i,y_i}'s needed for running algorithm KeyGen. If this is not the case then, by the previous settings, $t_{j,y_j} \equiv 1/\beta \bmod p_1$ and $\mathcal{B}$ can use $A_1^{\beta} \cdot C_4 = g_1^{1/t_{j,y_j}} \cdot C_4$ (see Assumption 2). So, the answers to the last $q-k$ queries have the same distribution as in G_k and G_{k-1}.

Challenge Construction. The challenge is created by running algorithm $\mathsf{Encrypt}$ on input the randomly chosen challenge vector $\vec{x}$ and Pk'. Under the assumption that $\mathcal{B}$ has correctly guessed x_j and thus $x_j = 1 - y_j^{(k)}$, Pk' contains all the values to compute an encryption of $\vec{x}$. Then the challenge ciphertext is distributed exactly like in G_{k-1} and G_k. $\square$

G_q **gives no advantage.** We observe that in G_q the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η and the Pk and the answer to the key queries give no help to $\mathcal{A}$. Therefore we can conclude that for all adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_q} = 0$. We have thus proved.

Theorem 26. *If Assumptions* 1 *and* 2 *hold for generator* $\mathcal{G}$*, then the HVE scheme presented is* 0*-secure.*

8.5 Constructing 1-secure HVE

In this section we describe our construction for a 1-secure HVE scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$*:* The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4,\ \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ with known factorization and chooses random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$. For $i \in [\ell]$ and $b \in \{0,1\}$, the algorithm chooses random $t_{i,b} \in Z_N$, random $v_i \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. The public parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12} = g_1 \cdot g_2, g_4, (t_{i,b})_{i\in[\ell], b\in\{0,1\}}, (v_i)_{i\in[\ell]}]$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$*:* Let $S_{\vec{y}}$ be the set of indices i such that $y_i \neq \star$. For $i \in S_{\vec{y}}$, the key generation algorithm chooses random $a_i \in \mathbb{Z}_N$ such that $\sum_{i\in S_{\vec{y}}} a_i = 0$. For $i \in S_{\vec{y}}$, the algorithm sets $Y_i = g_{12}^{a_i/t_{i,y_i}} g_4^{a_i/v_i}$. The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$. Notice that here we used the fact that $S_{\vec{y}}$ has size at least 2.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$*:* The encryption algorithm chooses random $s \in \mathbb{Z}_N$. For $i \in [\ell]$, the algorithm chooses random $Z_i \in \mathbb{G}_{p_3}$, sets $X_i = T_{i,x_i}^s Z_i$, and returns $(X_i)_{i\in[\ell]}$.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$*:* The test algorithm returns TRUE if $\prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) = 1$.

Correctness. It is easy to verify the correctness of the scheme.

8.6 1-Security of our HVE scheme

To prove that our HVE scheme is 1-secure, we rely on static Assumptions 1 and 4. For a probabilistic polynomial-time 1-adversary $\mathcal{A}$ our proof of security will be structured as a sequence of 2 games between $\mathcal{A}$ and a challenger $\mathcal{C}$. The first game, $\mathsf{GReal}(1)$, is the real HVE security game described in the previous section. The remaining games, called $\mathsf{G}_0, \mathsf{G}_1$, are described (and shown indistinguishable) in the following sections.

In the rest of this section, when we refer to adversaries we mean 1-adversaries and when we refer to GReal we mean $\mathsf{GReal}(1)$.

Description of G_0. G_0 is like GReal, except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters Pk given to $\mathcal{A}$. Specifically,

Setup. $\mathcal{C}$ chooses random $g_1 \in \mathbb{G}_{p_1}, g_2 \in \mathbb{G}_{p_2}, g_3 \in \mathbb{G}_{p_3}, g_4 \in \mathbb{G}_{p_4}$. For $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$, random $v_i \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_2^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$ and $T'_{i,b} = g_1^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. On a query for vector $\vec{y}$, $\mathcal{C}$ returns the output of KeyGen on input $\vec{y}$ and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}, (v_i)_{i\in[\ell]}]$, where $g_{12} = g_1 \cdot g_2$.

Challenge Construction. $\mathcal{C}$ picks one of the two challenge vectors provided by $\mathcal{A}$ and encrypts it with respect to public parameters Pk'.

Proof of indistinguishability of GReal and G_0 for 1-security.

Lemma 27. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{GReal}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_0} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

The proof follows the line of that of Lemma 24 and therefore we omit the details.

Description of G_1. This game is like G_0, except that in the answers provided by $\mathcal{C}$ the key queries. Specifically the queries are answered without the $\mathbb{G}_{p_1}$ part. The $\mathbb{G}_{p_2}$ part of all answers is like in G_0. Specifically, we have.

Query answering. $\mathcal{C}$ answers the queries in the following way. On input vector $\vec{y}$, for $i \in S_{\vec{y}}$, $\mathcal{C}$ chooses random $a_i, b_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\vec{y}}} a_i = \sum_{i\in S_{\vec{y}}} b_i = 0$. $\mathcal{C}$ sets, for $i \in S_{\vec{y}}$, $Y_i = g_2^{a_i/t_{i,y_i}} \cdot g_4^{b_i/v_i} \cdot W_i$.

Proof of indistinguishability of G_0 and G_1.

Lemma 28. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_0} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_1} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage at least ϵ in breaking Assumption* 4.

PROOF. $\mathcal{B}$ receives $(\mathcal{I}, A_2, A_3, A_4, A_{14})$ and T and, depending on the nature of T, simulates G_0 or G_1 with $\mathcal{A}$.

Setup. $\mathcal{B}$ sets $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$. For $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b}, v_i \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{t_{i,b}} \cdot g_4^{v_i} \cdot R_{i,b}$. These settings determine $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. used by $\mathcal{B}$ to interact with $\mathcal{A}$. Notice that Pk has the same distribution as that seen by $\mathcal{A}$ in G_0 and G_1.

Key Query Answering. $\mathcal{B}$ computes the answer to query for vector $\vec{y}$ as follows. For $i \in S_{\vec{y}}$, $\mathcal{B}$ chooses random $a_i \in \mathbb{Z}_N$ subject to $\sum_{i\in S_{\vec{y}}} a_i = 0$ and sets $Y_i = g_2^{a_i/t_{i,y_i}} \cdot T^{a_i/v_i}$. Now suppose $T = B_{14}$ and write $T = g_1 g_4^c$ for some $g_1 \in \mathbb{G}_{p_1}$ and $c \in \mathbb{Z}_{p_4}$. By our setting we have $Y_i = g_1^{a_i/v_i} \cdot g_2^{a_i/t_{i,y_i}} \cdot g_4^{ca_i/v_i}$ which implicitly sets $t_{i,y_i} \equiv v_i \mod p_1$. It is easy to see that the answer to the query is distributed as in G_0. Instead, if $T = B_4$ then the key does not contain the $\mathbb{G}_{p_1}$ part and thus the answer to the query is distributed as in G_1.

Notice that, since $\mathcal{A}$ is a 1-adversary, then for every query vector $\vec{y}$ that $\mathcal{A}$ can submit, it holds that for each $i \in [\ell]$, $y_i = \star$ or $y_i = x_{0,i} = x_{1,i}$. Therefore, or each $i \in [\ell]$, $\mathcal{B}$ needs only to determine $t_{i,x_{0,i}} = t_{i,x_{1,i}}$ and it does so by setting it congruent to $v_i \mod p_1$.

Challenge Construction. For $i \in [\ell]$, $\mathcal{B}$ chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets, for $i \in [\ell]$, $X_i = A_{14}^{v_i} \cdot Z_i$. Finally notice that by writing $A_{14} = (g_1 \cdot g_4^c)^s$, the challenge ciphertext is distributed exactly like in G_0 and G_1. □

G_1 **gives no advantage.** We observe that the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η and the Pk and the answer to the key queries give no help to $\mathcal{A}$. Therefore we can conclude that for all adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathsf{G}_1}^{\mathcal{A}} = 0$. We have thus proved.

Theorem 29. *If Assumptions* 1 *and* 4 *hold for generator* $\mathcal{G}$*, then the HVE scheme presented is* 1*-secure.*

Chapter 9

(Anonymous) Hierarchical IBE

[1] In this Chapter we present a class of predicate encryption schemes with a powerful capability: delegation. Delegation is the possibility of deriving from a key for a predicate P a key for a less 'general' predicate P', that is, a predicate such that if $P'(x) = 1$ then $P(x) = 1$ (but the viceversa can not hold). Specifically, we consider the primitive Hierarchical IBE (HIBE). In HIBE the user encrypts a message with a vector of identities $(id_1, \ldots, id_m)$ of length $\leq m$. The keys are also associated with vectors of identities $\vec{id}$ of length $\leq m$. A key for vector $\vec{id}$ can decrypt a ciphertext for vector $\vec{id}'$ iff $\vec{id}$ is a *prefix* of $\vec{id}'$. We formalize this in next section.

9.1 Public-Key Anonymous HIBE

9.1.1 Hierarchical Identity Based Encryption

A Hierarchical Identity Based Encryption scheme (henceforth abbreviated in HIBE) over an alphabet Σ is a tuple of five efficient and probabilistic algorithms: (Setup, $\mathsf{Encrypt}$, KeyGen, $\mathsf{Decrypt}$, $\mathsf{Delegate}$).

$\mathsf{Setup}(1^\lambda, 1^\ell)$: takes as input security parameter λ and maximum depth of an identity vector ℓ and outputs public parameters Pk and master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_j))$: takes as input master secret key Msk, identity vector $ID \in \Sigma^j$ with $j \leq \ell$ and outputs a private key $\mathsf{Sk}_{\mathsf{Id}}$.

$\mathsf{Delegate}(\mathsf{Pk}, \mathsf{Id}, \mathsf{Sk}_{\mathsf{Id}}, \mathsf{Id}_{j+1})$: takes as input public parameters Pk, secret key for identity vector $\mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_j)$ of depth $j < \ell$, $\mathsf{Id}_{j+1} \in \Sigma$ and outputs a secret key for the depth $j+1$ identity vector $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j, \mathsf{Id}_{j+1})$.

[1]The material in this Chapter is a polite version of a work of the author with Angelo De Caro and Giuseppe Persiano [18]

$\mathsf{Encrypt}(\mathsf{Pk}, M, \mathsf{Id})$: takes as input public parameters Pk, message M and identity vector Id and outputs a ciphertext Ct.

$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk})$: takes as input public parameters Pk, ciphertext Ct and secret key Sk and outputs the message M. We make the following obvious consistency requirement. Suppose ciphertext Ct is obtained by running the $\mathsf{Encrypt}$ algorithm on public parameters Pk, message M and identity Id and that Sk is a secret key for identity Id obtained through a sequence of KeyGen and $\mathsf{Delegate}$ calls using the same public parameters Pk. Then $\mathsf{Decrypt}$, on input Pk, Ct and Sk, returns M except with negligible probability.

9.1.2 Security definition

We give complete form of the security definition following [38]. Our security definition captures semantic security and ciphertext anonymity by means of the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup. The challenger $\mathcal{C}$ runs the Setup algorithm to generate public parameters Pk which it gives to the adversary $\mathcal{A}$. We let S denote the set of private keys that the challenger has created but not yet given to the adversary. At this point, $S = \emptyset$.

Phase 1. $\mathcal{A}$ makes Create, Delegate, and Reveal key queries. To make a Create query, $\mathcal{A}$ specifies an identity vector Id of depth j. 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 $\mathsf{Sk}_{\mathsf{Id}}$ in the set S and $\mathsf{Id}_{j+1} \in \Sigma$. In response, $\mathcal{C}$ appends Id_{j+1} to Id and makes a key for this new identity by running the delegation algorithm on $\mathsf{Id}, \mathsf{Sk}_{\mathsf{Id}}$ and Id_{j+1}. $\mathcal{C}$ 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. We note 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 $(M_0, \mathsf{Id}_0^\star)$ and $(M_1, \mathsf{Id}_1^\star)$ to $\mathcal{C}$. We require that no revealed identity in Phase 1 is a prefix of either $\mathsf{Id}_0^\star$ or $\mathsf{Id}_1^\star$. $\mathcal{C}$ chooses random $\beta \in \{0, 1\}$, encrypts M_β under $\mathsf{Id}_\beta^\star$ and sends the resulting ciphertext to $\mathcal{A}$.

Phase 2. This is the same as Phase 1 with the added restriction that any revealed identity vector must not be a prefix of either $\mathsf{Id}_0^\star$ or $\mathsf{Id}_1^\star$.

Guess. $\mathcal{A}$ must output a guess β' for β. The advantage of $\mathcal{A}$ is defined to be $\mathrm{Prob}[\beta' = \beta] - \frac{1}{2}$.

Definition 30. *An Anonymous Hierarchical Identity Based Encryption scheme is secure if all polynomial time adversaries achieve at most a negligible (in λ) advantage in the previous security game.*

Remark 31. *Our constructions will guarantee the anonimity of the identity vector but decryption will only work if the identity vector associated with the ciphertex is given. That is, the ciphertexts do not leak information on the identity vector but in order to decrypt, you have to know this identity vector. This can be a severe limitation for some applications but it can be acceptable for others. Notice that when the system is instanced as an Anonymous IBE (not hierarchical) where the identity is single, this is not a limitation because the decryption identity is contained in the key.*

9.2 Complexity Assumptions for our AHIBE scheme

In this section we give our complexity assumptions that will be used to prove the security of our AHIBE scheme and of its secret-key AIBE variant. They are formulated in composite order bilinear groups.

9.2.1 Assumption I_1

For a generator $\mathcal{G}$ returning bilinear settings of order N product of four primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ by running $\mathcal{G}(1^\lambda)$ and then pick

$$g_1, A_1 \leftarrow \mathbb{G}_{p_1}, A_2, B_2 \leftarrow \mathbb{G}_{p_2}, g_3, B_3 \leftarrow \mathbb{G}_{p_3}, g_4 \leftarrow \mathbb{G}_{p_4}, T_1 \leftarrow \mathbb{G}_{p_1p_2p_3}, T_2 \leftarrow \mathbb{G}_{p_1p_3}$$

and set $D = (\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption I_1 to be:

$$\mathsf{Adv}_{\mathsf{I}_1}^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|.$$

Assumption 3. *We say that Assumption I_1 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$ $\mathsf{Adv}_{\mathsf{I}_1}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

9.2.2 Assumption I_2

For a generator $\mathcal{G}$ returning bilinear settings of order N product of four primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ by running $\mathcal{G}(1^\lambda)$ and then pick

$$\alpha, s, r \leftarrow \mathbb{Z}_N,\ g_1 \leftarrow \mathbb{G}_{p_1},\ g_2, A_2, B_2 \leftarrow \mathbb{G}_{p_2},\ g_3 \leftarrow \mathbb{G}_{p_3},\ g_4 \leftarrow \mathbb{G}_{p_4},\ T_2 \leftarrow \mathbb{G}_T$$

and set $T_1 = \mathbf{e}(g_1, g_1)^{\alpha s}$ and $D = (\mathcal{I}, g_1, g_2, g_3, g_4, g_1^\alpha A_2, g_1^s B_2, g_2^r, A_2^r)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption I_2 to be:

$$\mathsf{Adv}_{\mathsf{I}_2}^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|.$$

Assumption 4. *We say that Assumption I_2 holds for generator $\mathcal{G}$ if for all probabilistic polynomial time algorithm $\mathcal{A}$ $\mathsf{Adv}_{\mathsf{I}_2}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

9.2.3 Assumption I_3

For a generator $\mathcal{G}$ returning bilinear settings of order N product of four primes, we define the following distribution. First pick a random bilinear setting $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ by running $\mathcal{G}(1^\lambda)$ and then pick

$$\hat{r}, s \leftarrow \mathbb{Z}_N,\ g_1, U, A_1 \leftarrow \mathbb{G}_{p_1},\ g_2, A_2, B_2, D_2, F_2 \leftarrow \mathbb{G}_{p_2},\ g_3 \leftarrow \mathbb{G}_{p_3},$$

$$g_4, A_4, B_4, D_4 \leftarrow \mathbb{G}_{p_4},\ A_{24}, B_{24}, D_{24} \leftarrow \mathbb{G}_{p_2p_4},\ T_2 \leftarrow \mathbb{G}_{p_1p_2p_4}$$

and set $T_1 = A_1^s D_{24}$ and $D = (\mathcal{I}, g_1, g_2, g_3, g_4, U, U^s A_{24}, U^{\hat{r}}, A_1A_4, A_1^{\hat{r}}A_2, g_1^{\hat{r}}B_2, g_1^s B_{24})$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption I_3 to be:

$$\mathsf{Adv}_{\mathsf{I}_3}^{\mathcal{A}}(\lambda) = |\mathrm{Prob}[\mathcal{A}(D, T_1) = 1] - \mathrm{Prob}[\mathcal{A}(D, T_2) = 1]|.$$

Assumption 5. *We say that Assumption I_3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial time algorithm $\mathcal{A}$ $\mathsf{Adv}_{\mathsf{I}_3}^{\mathcal{A}}(\lambda)$ is a negligible function of λ.*

9.2.4 Our construction

In this section we describe our construction for an Anonymous HIBE scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses random description $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and random $Y_1, X_1, u_1, \ldots, u_\ell \in \mathbb{G}_{p_1}, Y_3 \in \mathbb{G}_{p_3}, X_4, Y_4 \in \mathbb{G}_{p_4}$ and $\alpha \in \mathbb{Z}_N$. The public parameters are published as:

$$\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u_1, \ldots, u_\ell, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha).$$

The master secret key is $\mathsf{Msk} = (X_1, \alpha)$.

$\mathsf{KeyGen}(\mathsf{Msk}, \mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_j))$: The key generation algorithm chooses random $r_1, r_2 \in \mathbb{Z}_N$ and, for $i \in \{1, 2\}$, random $R_{i,1}, R_{i,2}, R_{i,j+1}, \ldots, R_{i,\ell} \in \mathbb{G}_{p_3}$. The secret key $\mathsf{Sk}_{\mathsf{Id}} = (K_{i,1}, K_{i,2}, E_{i,j+1}, \ldots, E_{i,\ell})$ is computed as

$$K_{1,1} = Y_1^{r_1} R_{1,1}, \quad K_{1,2} = Y_1^\alpha \left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1\right)^{r_1} R_{1,2}$$

$$E_{1,j+1} = u_{j+1}^{r_1} R_{1,j+1},\ \ldots,\ E_{1,\ell} = u_\ell^{r_1} R_{1,\ell},$$

$$K_{2,1} = Y_1^{r_2} R_{2,1}, \quad K_{2,2} = \left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1\right)^{r_2} R_{2,2}$$

$$E_{2,j+1} = u_{j+1}^{r_2} R_{2,j+1},\ \ldots,\ E_{1,\ell} = u_\ell^{r_2} R_{2,\ell}.$$

Notice that, $\mathsf{Sk}_{\mathsf{Id}}$ is composed by two sub-keys. The first sub-key, $(K_{1,1}, K_{1,2}, E_{1,j+1}, \ldots, E_{1,\ell})$, is used by the decryption algorithm to compute the blinding factor, the second, $(K_{2,1}, K_{2,2}, E_{2,j+1}, \ldots, E_{2,\ell})$, is used by the delegation algorithm and can be used also to verify that the identity vector of a given ciphertext matches the identity vector of the key.

$\mathsf{Delegate}(\mathsf{Pk}, \mathsf{Id}, \mathsf{Sk}_{\mathsf{Id}}, \mathsf{Id}_{j+1})$: Given a key $\mathsf{Sk}_{\mathsf{Id}} = (K'_{i,1}, K'_{i,2}, E'_{i,j+1}, \ldots, E'_{i,\ell})$ for $\mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_j)$, the delegation algorithm creates a key for $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j, \mathsf{Id}_{j+1})$ as follows. It chooses random $\tilde{r_1}, \tilde{r_2} \in \mathbb{Z}_N$ and, for $i \in \{1, 2\}$, random $R_{i,1}, R_{i,2}, R_{i,j+2}, \ldots, R_{i,\ell} \in \mathbb{G}_{p_3}$. The secret key $(K_{i,1}, K_{i,2}, E_{i,j+2}, \ldots, E_{i,\ell})$ is computed as

$$K_{1,1} = K'_{1,1}(K'_{2,1})^{\tilde{r_1}} R_{1,1}, K_{1,2} = K'_{1,2}(K'_{2,2})^{\tilde{r_1}} (E'_{1,j+1})^{\mathsf{Id}_{j+1}} (E'_{2,j+1})^{\tilde{r_1}\mathsf{Id}_{j+1}} R_{1,2},$$

$$E_{1,j+2} = E'_{1,j+2} \cdot (E'_{2,j+2})^{\tilde{r_1}} R_{1,j+2}, \ldots, E_{1,\ell} = E'_{1,\ell} \cdot (E'_{2,\ell})^{\tilde{r_1}} R_{1,\ell}.$$

and

$$K_{2,1} = (K'_{2,1})^{\tilde{r_2}} R_{2,1}, \quad K_{2,2} = (K'_{2,2})^{\tilde{r_2}} \cdot (E'_{2,j+1})^{\tilde{r_2}\mathsf{Id}_{j+1}} R_{2,2},$$

$$E_{2,j+2} = (E'_{2,j+2})^{\tilde{r_2}} R_{2,j+2}, \ldots, E_{2,\ell} = (E'_{2,\ell})^{\tilde{r_2}} R_{2,\ell}.$$

We observe that the new key has the same distributions as the key computed by the KeyGen algorithm on $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j, \mathsf{Id}_{j+1})$ with randomness $r_1 = r'_1 + (r'_2 \cdot \tilde{r_1})$ and $r_2 = r'_2 \cdot \tilde{r_2}$.

$\mathsf{Encrypt}(\mathsf{Pk}, M, \mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_j))$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$ and random $Z, Z' \in \mathbb{G}_{p_4}$. The ciphertext (C_0, C_1, C_2) for the message $M \in \mathbb{G}_T$ is computed as

$$C_0 = M \cdot \mathbf{e}(Y_1, Y_1)^{\alpha s}, \quad C_1 = \left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} t\right)^s Z, \quad C_2 = Y_1^s Z'.$$

$\mathsf{Decrypt}(\mathsf{Pk}, \mathsf{Ct}, \mathsf{Sk})$: The decryption algorithm assumes that the key and ciphertext both correspond to the same identity $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j)$. If the key identity is a prefix of this instead, then the decryption algorithm starts by running the key delegation algorithm to create a key with identity matching the ciphertext identity exactly. The decryption algorithm then computes the blinding factor as:

$$\frac{\mathbf{e}(K_{1,2}, C_2)}{\mathbf{e}(K_{1,1}, C_1)} = \frac{\mathbf{e}(Y_1, Y_1)^{\alpha s} \mathbf{e}\left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1, Y_1\right)^{r_1 s}}{\mathbf{e}\left(Y_1, u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1\right)^{r_1 s}} = \mathbf{e}(Y_1, Y_1)^{\alpha s}.$$

By comparing our construction with the one of [31], we notice that component t of the public key and components C_1 and C_2 of the ciphertext have a $\mathbb{G}_{p_4}$ part. This addition makes the system anonymous. Indeed, if we remove from our construction the $\mathbb{G}_{p_4}$ parts of t and C_1 and C_2 (and thus obtain the scheme of [31]) then it is possible to test if ciphertext (C_0, C_1, C_2) is relative to identity $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j)$ for public key $(N, Y_1, Y_3, Y_4, t, u_1, \ldots, u_\ell, \Omega)$ by testing $\mathbf{e}(C_2, t \cdot (u_1^{\mathsf{Id}_1} \cdots u_\ell^{\mathsf{Id}_\ell}))$ and $\mathbf{e}(C_1, Y_1)$ for equality.

9.2.5 Security

Following Lewko and Waters [31], we define two additional structures: *semi-functional ciphertexts* and *semi-functional keys*. These will not be used in the real scheme, but we need them in our proofs.

Semi-functional Ciphertext. We let g_2 denote a generator of $\mathbb{G}_{p_2}$. A semi-functional ciphertext is created as follows: first, we use the encryption algorithm to form a normal ciphertext (C'_0, C'_1, C'_2). We choose random exponents $x, z_c \in \mathbb{Z}_N$. We set:

$$C_0 = C'_0, \quad C_1 = C'_1 g_2^{x z_c}, \quad C_2 = C'_2 g_2^{x}.$$

Semi-functional Keys. To create a semi-functional key, we first create a normal key $(K'_{i,1},\ K'_{i,2}, E'_{i,j+1}, \ldots, E'_{i,\ell})$ using the key generation algorithm. We choose random exponents $z, \gamma, z_k \in \mathbb{Z}_N$ and, for $i \in \{1,2\}$, random exponents $z_{i,j+1}, \ldots, z_{i,\ell} \in \mathbb{Z}_N$. We set:

$$K_{1,1} = K'_{1,1} \cdot g_2^{\gamma}, K_{1,2} = K'_{1,2} \cdot g_2^{\gamma z_k}, (E_{1,i} = E'_{1,i} \cdot g_2^{\gamma z_{1,i}})_{i=j+1}^{\ell},$$

and

$$K_{2,1} = K'_{2,1} \cdot g_2^{z\gamma}, K_{2,2} = K'_{2,2} \cdot g_2^{z\gamma z_k}, (E_{2,i} = E'_{2,i} \cdot g_2^{z\gamma z_{2,i}})_{i=j+1}^{\ell}.$$

We note that when the first sub-key of a semi-functional key is used to decrypt a semi-functional ciphertext, the decryption algorithm will compute the blinding factor multiplied by the additional term $\mathbf{e}(g_2, g_2)^{x\gamma(z_k - z_c)}$. If $z_c = z_k$, decryption will still work. In this case, we say that the key is nominally semi-functional. If the second sub-key is used to test the identity vector of the ciphertext, then the decryption algorithm computes $\mathbf{e}(g_2, g_2)^{xz\gamma(z_k - z_c)}$ and if $z_c = z_k$, the test will still work.

To prove security of our Anonymous HIBE scheme, we rely on the static Assumptions I_1, I_2 and 3. For a probabilistic polynomial-time adversary $\mathcal{A}$ which makes q key queries, our proof of security will consist of the following sequence of $q + 5$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$.

GReal: is the real Anonymous HIBE security game.

$\mathsf{G}_{\mathsf{Real}'}$: is the same as the real game except that all key queries will be answered by fresh calls to the key generation algorithm, ($\mathcal{C}$ will not be asked to delegate keys in a particular way).

$\mathsf{G}_{\mathsf{Restricted}}$: is the same as $\mathsf{G}_{\mathsf{Real}'}$ except that $\mathcal{A}$ cannot ask for keys for identities which are prefixes of one of the challenge identities modulo p_2. We will retain this restriction in all subsequent games.

G_k: for k from 0 to q, we define G_k like $\mathsf{G}_{\mathsf{Restricted}}$ 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.

$\mathsf{G}_{\mathsf{Final}_0}$: is the same as G_q, except that the challenge ciphertext is a semi-functional encryption with C_0 random in $\mathbb{G}_T$ (thus the ciphertext is independent from the messages provided by $\mathcal{A}$).

$\mathsf{G}_{\mathsf{Final}_1}$: is the same as $\mathsf{G}_{\mathsf{Final}_0}$, except that the challenge ciphertext is a semi-functional encryption with C_1 random in $\mathbb{G}_{p_1p_2p_4}$ (thus the ciphertext is independent from the identity vectors provided by $\mathcal{A}$). It is clear that in this last game, no adversary can have advantage greater than 0.

We will show these games are indistinguishable in the following lemmata.

Indistinguishability of GReal and $\mathsf{G}_{\mathsf{Real}'}$

Lemma 32. *For any algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{GReal}} = \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Real}'}}$.*

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. □

Indistinguishability of $\mathsf{G}_{\mathsf{Real}'}$ and $\mathsf{G}_{\mathsf{Restricted}}$

Lemma 33. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Real}'}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Restricted}}} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage $\geq \frac{\epsilon}{3}$ in breaking Assumption I_1.*

PROOF. Suppose that $\mathcal{A}$ has probability ϵ of producing an identity vector $\mathsf{Id} = (\mathsf{Id}_1, \ldots, \mathsf{Id}_k)$, that is a prefix of one of the challenge identities $\mathsf{Id}^\star = (\mathsf{Id}^\star_1, \ldots, \mathsf{Id}^\star_j)$ modulo p_2. That is, there exists i and $j \in \{0, 1\}$ such that that $\mathsf{Id}_i \neq \mathsf{Id}^\star_{j,i}$ modulo N and that p_2 divides $\mathsf{Id}_i - \mathsf{Id}^\star_{j,i}$ and thus $a = \gcd(\mathsf{Id}_i - \mathsf{Id}^\star_{j,i}, N)$ is a nontrivial factor of N. We notice that p_2 divides a and set $b = \frac{N}{a}$. The following three cases are exhaustive and at least one occurs with probability at least $\epsilon/3$.

1. $\operatorname{ord}(Y_1) \mid b$.

2. $\mathrm{ord}(Y_1) \nmid b$ and $\mathrm{ord}(Y_4) \mid b$.

3. $\mathrm{ord}(Y_1) \nmid b$, $\mathrm{ord}(Y_4) \nmid b$ and $\mathrm{ord}(Y_3) \mid b$.

Suppose case 1 has probability at least $\epsilon/3$. We describe algorithm $\mathcal{B}$ that breaks Assumption I_1. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$ and T and constructs Pk by running the Setup algorithm with the only exception that $\mathcal{B}$ sets $Y_1 = g_1, Y_3 = g_3$, and $Y_4 = g_4$. Notice that $\mathcal{B}$ has the master secret key Msk associated with Pk. Then $\mathcal{B}$ runs $\mathcal{A}$ on input Pk and uses knowledge of Msk to answer $\mathcal{A}$'s queries. At the end of the game, for all Ids for which $\mathcal{A}$ has asked for the key and for $\mathsf{Id}^\star \in \{\mathsf{Id}_0^\star, \mathsf{Id}_1^\star\}$, $\mathcal{B}$ computes $a = \gcd(\mathsf{Id}_i - \mathsf{Id}_i^\star, N)$. Then, if $\mathbf{e}\left((A_1A_2)^a, B_2B_3\right)$ is the identity element of $\mathbb{G}_T$ then $\mathcal{B}$ tests if $\mathbf{e}(T^b, A_1A_2)$ 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 $p_1 = \mathrm{ord}(Y_1)$ divides b, then $\mathcal{B}$'s output is correct.

The other two cases are similar. Specifically, in case 2, $\mathcal{B}$ breaks Assumption I_1 in the same way except that Pk is constructed by setting $Y_1 = g_4, Y_3 = g_3$, and $Y_4 = g_1$ (this has the effect of exchanging the roles of p_1 and p_4). Instead in case 3, $\mathcal{B}$ constructs Pk by setting $Y_1 = g_3, Y_3 = g_1$, and $Y_4 = g_4$ (this has the effect of exchanging the roles of p_1 and p_3). □

Indistinguishability of $\mathsf{G}_{\mathsf{Restricted}}$ and G_0

Lemma 34. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Restricted}}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_0} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_1.*

PROOF. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$ and T and simulates $\mathsf{G}_{\mathsf{Restricted}}$ or G_0 with $\mathcal{A}$ depending on whether $T \in \mathbb{G}_{p_1p_3}$ or $T \in \mathbb{G}_{p_1p_2p_3}$.

$\mathcal{B}$ sets the public parameters as follows. $\mathcal{B}$ chooses random exponents $\alpha, a_1, \ldots, a_\ell, b, c \in \mathbb{Z}_N$ and sets $Y_1 = g_1, Y_3 = g_4, Y_4 = g_3$, $X_4 = Y_4^c, X_1 = Y_1^b$ and $u_i = Y_1^{a_i}$ for $i \in [\ell]$. $\mathcal{B}$ sends $\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u_1, \ldots, u_\ell, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha)$ to $\mathcal{A}$. Notice that $\mathcal{B}$ knows the master secret key $\mathsf{Msk} = (X_1, \alpha)$ associated with Pk and thus can answer all $\mathcal{A}$'s queries.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two pairs, $(M_0, \mathsf{Id}_0^\star = (\mathsf{Id}_{0,1}^\star, \ldots, \mathsf{Id}_{0,j}^\star))$ and $(M_1, \mathsf{Id}_1^\star = (\mathsf{Id}_{1,1}^\star, \ldots, \mathsf{Id}_{1,j}^\star))$. $\mathcal{B}$ chooses random $\beta \in \{0,1\}$ and computes the challenge ciphertext as follows:

$$C_0 = M_\beta \cdot \mathbf{e}(T, Y_1)^\alpha, \quad C_1 = T^{a_1\mathsf{Id}_{\beta,1}^\star + \cdots + a_j\mathsf{Id}_{\beta,j}^\star + b}, \quad C_2 = T.$$

We complete the proof with the following two observations. If $T \in \mathbb{G}_{p_1p_3}$, then T can be written as $Y_1^{s_1}Y_4^{s_3}$. In this case (C_0, C_1, C_2) is a normal ciphertext with randomness $s = s_1, Z = Y_4^{s_3a_1\mathsf{Id}_{\beta,1}^\star + \cdots + a_j\mathsf{Id}_{\beta,j}^\star + b}$ and $Z' = Y_4^{s_3}$. If $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $Y_1^{s_1}g_2^{s_2}Y_4^{s_3}$ and this case (C_0, C_1, C_2) is a semi-functional ciphertext with randomness $s = s_1, Z = Y_4^{s_3a_1\mathsf{Id}_{\beta,1}^\star + \cdots + a_j\mathsf{Id}_{\beta,j}^\star + b}$, $Z' = Y_4^{s_3}$, $\gamma = s_2$ and $z_c = a_1\mathsf{Id}_{\beta,1}^\star + \cdots + a_j\mathsf{Id}_{\beta,j}^\star + b$. □

Indistinguishability of G_{k-1} and G_k

Lemma 35. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_k} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_1.*

Proof. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$ and T and simulates G_{k-1} or G_k with $\mathcal{A}$ depending on whether $T \in \mathbb{G}_{p_1p_3}$ or $T \in \mathbb{G}_{p_1p_2p_3}$.

$\mathcal{B}$ sets the public parameters by choosing random exponents $\alpha, a_1, \ldots, a_\ell, b, c \in \mathbb{Z}_N$ and setting $Y_1 = g_1, Y_3 = g_3, Y_4 = g_4, X_4 = Y_4^c, X_1 = Y_1^b$ and $u_i = Y_1^{a_i}$ for $i \in [\ell]$. $\mathcal{B}$ sends the public parameters $\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u_1, \ldots, u_\ell, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha)$ to $\mathcal{A}$. Notice that $\mathcal{B}$ knows the master secret key $\mathsf{Msk} = (X_1, \alpha)$ associated with Pk. Let us now explain how $\mathcal{B}$ answers the i-th key query for identity $(\mathsf{Id}_{i,1}, \ldots, \mathsf{Id}_{i,j})$.

For $i < k$, $\mathcal{B}$ creates a semi-functional key by choosing random exponents $r_1, r_2, f, z, w \in \mathbb{Z}_N$ and, for $i \in \{1, 2\}$, random $w_{i,2}, w_{i,j+1}, \ldots, w_{i,\ell} \in \mathbb{Z}_N$ and setting:

$$K_{1,1} = Y_1^{r_1} \cdot (B_2B_3)^f, \quad K_{1,2} = Y_1^\alpha \cdot (B_2B_3)^w \left(u_1^{\mathsf{Id}_{i,1}} \cdots u_j^{\mathsf{Id}_{i,j}} X_1\right)^{r_1} Y_3^{w_{1,2}},$$

$$E_{1,j+1} = u_{j+1}^{r_1} \cdot (B_2B_3)^{w_{1,j+1}}, \ldots, E_{1,\ell} = u_\ell^{r_1} \cdot (B_2B_3)^{w_{1,\ell}}.$$

and

$$K_{2,1} = Y_1^{r_2} \cdot (B_2B_3)^{zf}, \quad K_{2,2} = (B_2B_3)^{zw} \left(u_1^{\mathsf{Id}_{i,1}} \cdots u_j^{\mathsf{Id}_{i,j}} X_1\right)^{r_2} Y_3^{w_{2,2}},$$

$$E_{2,j+1} = u_{j+1}^{r_2} \cdot (B_2B_3)^{w_{2,j+1}}, \ldots, E_{2,\ell} = u_\ell^{r_2} \cdot (B_2B_3)^{w_{2,\ell}}.$$

By writing B_2 as g_2^ϕ, we have that this is a properly distributed semi-functional key with $\gamma = \phi \cdot f$ and $\gamma \cdot z_k = \phi \cdot w$.

For $i > k$, $\mathcal{B}$ runs the KeyGen algorithm using the master secret key $\mathsf{Msk} = (X_1, \alpha)$.

To answer the k-th key query for $\mathsf{Id}_k = (\mathsf{Id}_{k,1}, \ldots, \mathsf{Id}_{k,j})$, $\mathcal{B}$ sets $z_k = a_1\mathsf{Id}_{k,1} + \cdots + a_j\mathsf{Id}_{k,j} + b$, chooses random exponents $r_2' \in \mathbb{Z}_N$ and, for $i \in \{1, 2\}$, random $w_{i,2}, w_{i,j+1}, \ldots, w_{i,\ell} \in \mathbb{Z}_N$, and sets:

$$K_{1,1} = T, \quad K_{1,2} = Y_1^\alpha \cdot T^{z_k} Y_3^{w_{1,2}}, \quad (E_{1,m} = T^{a_m} Y_3^{w_{1,m}})_{m=j+1}^{\ell}.$$

and

$$K_{2,1} = T^{r_2'}, \quad K_{2,2} = T^{r_2' \cdot z_k} Y_3^{w_{2,2}}, \quad (E_{2,m} = T^{r_2' \cdot a_m} Y_3^{w_{2,m}})_{m=j+1}^{\ell}.$$

We have the following two observations. If $T \in \mathbb{G}_{p_1p_3}$, then T can be written as $Y_1^{r_1'} Y_3^{r_3}$ and $(K_{i,1}, K_{i,2}, E_{i,j+1}, \ldots, E_{i,\ell})$ is a normal key with randomness $r_1 = r_1'$, $r_2 = r_1' \cdot r_2'$. If $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $Y_1^{r_1'} g_2^{s_2} Y_3^{r_3}$. In this case the key is a semi-functional key with randomness $r_1 = r_1'$, $r_2 = r_1' \cdot r_2'$, $\gamma = s_2$ and $z = r_2'$.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two pairs, $(M_0, \mathsf{Id}_0^\star = (\mathsf{Id}_{0,1}^\star, \ldots, \mathsf{Id}_{0,j}^\star))$ and $(M_1, \mathsf{Id}_1^\star = (\mathsf{Id}_{1,1}^\star, \ldots, \mathsf{Id}_{1,j}^\star))$. $\mathcal{B}$ chooses random $\beta \in \{0, 1\}$ and random $z, z' \in \mathbb{Z}_N$ and computes the challenge ciphertext as follows:

$$C_0 = M_\beta \cdot \mathbf{e}(A_1A_2, Y_1)^\alpha, \quad C_1 = (A_1A_2)^{a_1\mathsf{Id}_{\beta,1}^\star + \cdots + a_j\mathsf{Id}_{\beta,j}^\star + b} Y_4^z, \quad C_2 = A_1A_2 Y_4^{z'}.$$

This implicitly sets $Y_1^s = A_1$ and $z_c = a_1 \mathsf{Id}^\star_{\beta,1} + \cdots + a_j \mathsf{Id}^\star_{\beta,j} + b \pmod{p_2}$. Since Id_k is not a prefix of $\mathsf{Id}^\star_\beta$ 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 a semi-functional ciphertext for Id_k, then we will have that $z_k = z_c$ and thus decryption always works (independently of T).

We can thus conclude that, if $T \in \mathbb{G}_{p_1 p_3}$ then $\mathcal{B}$ has properly simulated G_{k-1}. If $T \in \mathbb{G}_{p_1 p_2 p_3}$, then $\mathcal{B}$ has properly simulated G_k. □

Indistinguishability of G_q and $\mathsf{G}_{\mathsf{Final}_0}$

Lemma 36. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_q} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Final}_0}} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_2.*

Proof. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_2, g_3, g_4, g_1^\alpha A_2, g_1^s B_2, g_2^r, A_2^r)$ and T and simulates G_q or $\mathsf{G}_{\mathsf{Final}_0}$ with $\mathcal{A}$ depending on whether $T = \mathbf{e}(g_1, g_1)^{\alpha s}$ or T is a random element of $\mathbb{G}_T$.

$\mathcal{B}$ sets the public parameters as follows. $\mathcal{B}$ chooses random exponents $a_1, \ldots, a_\ell, b, c \in \mathbb{Z}_N$ and sets $Y_1 = g_1, Y_3 = g_3, Y_4 = g_4, X_4 = Y_4^c, X_1 = Y_1^b$, and $u_i = Y_1^{a_i}$ for $i \in [\ell]$. $\mathcal{B}$ computes $\Omega = \mathbf{e}(g_1^\alpha A_2, Y_1) = \mathbf{e}(Y_1^\alpha, Y_1)$ and send public parameters $\mathsf{Pk} = (N, Y_1, Y_2, Y_3, t = X_1 X_4, u_1, \ldots, u_\ell, \Omega)$ to $\mathcal{A}$.

Each time $\mathcal{B}$ is asked to provide a key for an identity $(\mathsf{Id}_1, \ldots, \mathsf{Id}_j)$, $\mathcal{B}$ creates a semi-functional key choosing random exponents $r_1, r_2, z, z' \in \mathbb{Z}_N$ and, for $i \in \{1, 2\}$, random $z_{i,j+1}, \ldots, z_{i,\ell}, w_{i,1}, w_{i,2}, w_{i,j+1}, \ldots, w_{i,\ell} \in \mathbb{Z}_N$ and setting:

$$K_{1,1} = Y_1^{r_1} \cdot g_2^z \cdot Y_3^{w_{1,1}}, \quad K_{1,2} = (g_1^\alpha A_2) \cdot g_2^{z'} \cdot \left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1\right)^{r_1} \cdot Y_3^{w_{1,2}},$$
$$E_{1,j+1} = u_{j+1}^{r_1} \cdot g_2^{z_{1,j+1}} \cdot Y_3^{w_{1,j+1}}, \ \ldots, \ E_{1,\ell} = u_\ell^{r_1} \cdot g_2^{z_{1,\ell}} \cdot Y_3^{w_{1,\ell}}.$$

and

$$K_{2,1} = Y_1^{r_2} \cdot (g_2^r)^z \cdot Y_3^{w_{2,1}}, \quad K_{2,2} = A_2^r \cdot (g_2^r)^{z'} \cdot \left(u_1^{\mathsf{Id}_1} \cdots u_j^{\mathsf{Id}_j} X_1\right)^{r_2} \cdot Y_3^{w_{2,2}},$$
$$E_{2,j+1} = u_{j+1}^{r_2} \cdot g_2^{z_{2,j+1}} \cdot Y_3^{w_{2,j+1}}, \ \ldots, \ E_{2,\ell} = u_\ell^{r_2} \cdot g_2^{z_{2,\ell}} \cdot Y_3^{w_{2,\ell}}.$$

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two pairs, $(M_0, \mathsf{Id}^\star_0 = (\mathsf{Id}^\star_{0,1}, \ldots, \mathsf{Id}^\star_{0,j}))$ and $(M_1, \mathsf{Id}^\star_1 = (\mathsf{Id}^\star_{1,1}, \ldots, \mathsf{Id}^\star_{1,j}))$. $\mathcal{B}$ chooses random $\beta \in \{0,1\}$ and random $z, z' \in \mathbb{Z}_N$ and computes the challenge ciphertext as follows:

$$C_0 = M_\beta \cdot T, \quad C_1 = (g_1^s B_2)^{a_1 \mathsf{Id}^\star_{\beta,1} + \cdots + a_j \mathsf{Id}^\star_{\beta,j} + b} \cdot Y_4^z, \quad C_2 = g_1^s B_2 \cdot Y_4^{z'}.$$

This implicitly sets $z_c = (a_1 \mathsf{Id}^\star_{\beta,1} + \cdots + a_j \mathsf{Id}^\star_{\beta,j} + b) \bmod p_2$. We note that $u_i = Y_1^{a_i \bmod p_1}$ and $X_1 = Y_1^{b \bmod p_1}$ are elements of $\mathbb{G}_{p_1}$, so when $a_1, \cdots, a_\ell$ and b 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 $T = \mathbf{e}(g, g)^{\alpha s}$, then the ciphertext constructed is a properly distributed semi-functional ciphertext with message M_β. If T instead is a random element of $\mathbb{G}_T$, then the ciphertext is a semi-functional ciphertext with a random message.
□

Indistinguishability of $\mathsf{G}_{\mathsf{Final}_0}$ and $\mathsf{G}_{\mathsf{Final}_1}$

Lemma 37. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Final}_0}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Final}_1}} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_3.*

Proof. First, notice that if exists an adversary $\mathcal{A}'$ which distinguishes an encryption for an identity vector $\mathsf{Id}^\star_0$ from an encryption for an identity vector $\mathsf{Id}^\star_1$, where $\mathsf{Id}^\star_0$ and $\mathsf{Id}^\star_1$ are chosen by $\mathcal{A}'$, then there exists an adversary $\mathcal{A}$ which distinguishes an encryption for an identity $\mathsf{Id}^\star$ chosen by $\mathcal{A}$ from an encryption for a random identity vector. Hence, we suppose that we are simulating the games for a such adversary.

$\mathcal{B}$ receives $(\mathcal{I}, g_1, g_2, g_3, g_4, U, U^s A_{24}, U^{\hat{r}}, A_1A_4, A_1^{\hat{r}}A_2, g_1^{\hat{r}}B_2, g_1^s B_{24})$ and T and simulates $\mathsf{G}_{\mathsf{Final}_0}$ or $\mathsf{G}_{\mathsf{Final}_1}$ with $\mathcal{A}$ depending on whether $T = A_1^s D_{24}$ or T is random in $\mathbb{G}_{p_1p_2p_4}$.

$\mathcal{B}$ sets the public parameters as follows. $\mathcal{B}$ chooses random exponents $\alpha, a_1, \dots, a_\ell \in \mathbb{Z}_N$ and sets $Y_1 = g_1, Y_3 = g_3, Y_4 = g_4, t = A_1A_4,\ u_i = U^{a_i}$ for $i \in [\ell]$, and $\Omega = \mathbf{e}(Y_1, Y_1)^\alpha$. $\mathcal{B}$ sends the public parameters $\mathsf{Pk} = (N, Y_1, Y_2, Y_3, t, u_1, \dots, u_\ell, \Omega)$ to $\mathcal{A}$.

Each time $\mathcal{B}$ is asked to provide a key for an identity $(\mathsf{Id}_1, \dots, \mathsf{Id}_j)$, $\mathcal{B}$ creates a semi-functional key choosing random exponents $r'_1, r'_2 \in \mathbb{Z}_N$ and, for $\in \{1,2\}$, random $z_{i,j+1}, \dots, z_{i,\ell}, w_{i,1}, w_{i,2},\ w_{i,j+1}, \dots, w_{i,\ell} \in \mathbb{Z}_N$ and setting:

$$K_{1,1} = (g_1^{\hat{r}}B_2)^{r'_1} Y_3^{w_{1,1}}, \quad K_{1,2} = Y_1^\alpha \left(\left(U^{\hat{r}}\right)^{a_1\mathsf{Id}_1 + \dots + a_j\mathsf{Id}_j} (A_1^{\hat{r}}A_2) \right)^{r'_1} Y_3^{w_{1,2}},$$

$$E_{1,j+1} = \left(U^{\hat{r}}\right)^{r'_1 a_{j+1}} Y_2^{z_{1,j+1}} Y_3^{w_{1,j+1}}, \dots, E_{1,\ell} = \left(U^{\hat{r}}\right)^{r'_1 a_\ell} Y_2^{z_{1,\ell}} Y_3^{w_{1,\ell}}.$$

and

$$K_{2,1} = (g_1^{\hat{r}}B_2)^{r'_2} Y_3^{w_{2,1}}, \quad K_{2,2} = \left(\left(U^{\hat{r}}\right)^{a_1\mathsf{Id}_1 + \dots + a_j\mathsf{Id}_j} (A_1^{\hat{r}}A_2) \right)^{r'_2} Y_3^{w_{2,2}},$$

$$E_{2,j+1} = \left(U^{\hat{r}}\right)^{r'_2 a_{j+1}} Y_2^{z_{2,j+1}} Y_3^{w_{2,j+1}}, \dots, E_{2,\ell} = \left(U^{\hat{r}}\right)^{r'_2 a_\ell} Y_2^{z_{2,\ell}} Y_3^{w_{2,\ell}}.$$

This implicitly sets the randomness $r_1 = \hat{r}r'_1$ and $r_2 = \hat{r}r'_2$. At some point, $\mathcal{A}$ sends $\mathcal{B}$ two pairs, $(M_0, \mathsf{Id}^\star = (\mathsf{Id}^\star_1, \dots, \mathsf{Id}^\star_j))$ and $(M_1, \mathsf{Id}^\star = (\mathsf{Id}^\star_1, \dots, \mathsf{Id}^\star_j))$. $\mathcal{B}$ chooses random $C_0 \in \mathbb{G}_T$ and computes the challenge ciphertext as follows:

$$C_0, \quad C_1 = T\,(U^s A_{24})^{a_1\mathsf{Id}^\star_1 + \dots + a_j\mathsf{Id}^\star_j}, \quad C_2 = g_1^s B_{24}.$$

This implicitly sets x and z_c to random values.

If $T = A_1^s D_{24}$, then this is properly distributed semi-functional ciphertext with C_0 random and for identity vector $\mathsf{Id}^\star$. If T is a random element of $\mathbb{G}_{p_1p_2p_4}$, then this is a semi-functional ciphertext with C_0 random in $\mathbb{G}_T$ and C_1 and C_2 random in $\mathbb{G}_{p_1p_2p_4}$.

Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T. $\square$

$\mathsf{G}_{\mathsf{Final}_1}$ gives no advantage

Theorem 38. *If Assumptions I_1, I_2 and I_3 hold then our Anonymous HIBE scheme is secure.*

Proof. If the assumptions hold then we have proved by the previous lemmata that the real security game is indistinguishable from $\mathsf{G}_{\mathsf{Final}_1}$, in which the value of β is information-theoretically hidden from the attacker. Hence the attacker can obtain no advantage in breaking the Anonymous HIBE scheme. $\square$

Chapter 10

Secret-Key Anonymous IBE

In the previous chapters we focused on definitions and constructions for public-key primitives. It is natural to extend this model to the *secret-key* model. In it, both ciphertexts and keys are obtained by the master secret-key (MSK) and the ciphertext-security of security is changed such that the adversary is given access to an oracle for encryptions but we also require the security of the keys. In fact, observe that in the public-key model the security of the keys is impossible to achieve because the adversary, having the Pk can encrypt any messages and test it against a key to obtain informations on the latter. Instead, in the secret-key model, it makes sense to consider the privacy of the key. This is modelled in a game where the adversary has access to encryption and key oracle under the restriction that it cannot issue ciphertext query for vectors that allow it to distinguish the challenges. In next section we present our definitions for symmetric-key (A)IBE.

10.1 Secret Key Identity Based Encryption

A Secret-Key Identity Based Encryption scheme (IBE) is a tuple of four efficient and probabilistic algorithms: (Setup, $\mathsf{Encrypt}$, KeyGen, $\mathsf{Decrypt}$).

$\mathsf{Setup}(1^\lambda)$: takes as input a security parameter λ and outputs the public parameters Pk and a master secret key Msk.

$\mathsf{KeyGen}(\mathsf{Msk}, \mathsf{Id})$: takes as input of the master secret key Msk, and an identity Id, and outputs a private key $\mathsf{Sk}_{\mathsf{Id}}$.

$\mathsf{Encrypt}(\mathsf{Msk}, M, \mathsf{Id})$: takes as input the master secret key Msk, a message M, and an identity Id and outputs a ciphertext Ct.

$\mathsf{Decrypt}(\mathsf{Ct}, \mathsf{Sk})$: takes as input a ciphertext Ct and a secret key Sk and outputs the message M, if the ciphertext was an encryption to an identity Id and the secret key is for the same identity.

10.2 Security definitions

We present the security of an Anonymous IBE scheme in secret key model. In this model, we have two definition of security: *c*iphertext security and *k*ey security.

10.3 Ciphertext Security definition

Security is defined through the following game, played by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup. $\mathcal{C}$ runs the Setup algorithm to generate master secret key Msk which is kept secret.

Phase 1. $\mathcal{A}$ can make queries to the oracle Encrypt. To make a such query, $\mathcal{A}$ specifies a pair (M, Id) and receives an encryption of this pair computed using the Encrypt algorithm with Msk. $\mathcal{A}$ can make queries to the oracle KeyGen. To make a such query, $\mathcal{A}$ specifies an identity Id and receives a key of this identity computed using the KeyGen algorithm with Msk.

Challenge. $\mathcal{A}$ gives to $\mathcal{C}$ two pair message-identity (M_0, Id_0) and (M_1, Id_1). The identities must satisfy the property that no revealed identity in Phase 1 was either Id_0 or Id_1. $\mathcal{C}$ sets $\beta \in \{0, 1\}$ randomly and encrypts M_β under Id_β. $\mathcal{C}$ sends the ciphertext to the adversary.

Phase 2. This is the same as Phase 1 with the added restriction that any revealed identity must not be either Id_0 or Id_1.

Guess. $\mathcal{A}$ must output a guess β' for β. The advantage of $\mathcal{A}$ is defined to be $\mathrm{Prob}[\beta' = \beta] - \frac{1}{2}$.

Definition 39. *An Anonymous Identity Based Encryption scheme is* ciphertext-secure *if all polynomial time adversaries achieve at most a negligible (in λ) advantage in the previous security game.*

10.3.1 Key Security definition

Security is defined through the following game, played by a challenger $\mathcal{C}$ and an attacker $\mathcal{A}$.

Setup. $\mathcal{C}$ runs the Setup algorithm to generate master secret key Msk which is kept secret.

Phase 1. $\mathcal{A}$ can make queries to the oracle Encrypt. To make a such query, $\mathcal{A}$ specifies a pair (M, Id) and receives an encryption of this pair computed using the Encrypt algorithm with the master secret key Msk. $\mathcal{A}$ can make queries to the oracle KeyGen. To make a such query, $\mathcal{A}$ specifies an identity Id and receives a key of this identity computed using the KeyGen algorithm with the master secret key Msk.

Challenge. $\mathcal{A}$ gives to $\mathcal{C}$ two identities Id_0 and Id_1. If in Phase 1 $\mathcal{A}$ did make a query (M, Id) to the oracle $\mathsf{Encrypt}$ such that Id was either Id_0 or Id_1, then the experiment fails. $\mathcal{C}$ sets $\beta \in \{0,1\}$ randomly and compute the secret key for Id_β. $\mathcal{C}$ sends the secret key to the adversary.

Phase 2. This is the same as Phase 1 with the added restriction that if $\mathcal{A}$ did make a query (M, Id) to the oracle $\mathsf{Encrypt}$ such that Id was either Id_0 or Id_1, then the experiment fails.

Guess. $\mathcal{A}$ must output a guess β' for β. The advantage $\mathcal{A}$ is defined to be $\mathrm{Prob}[\beta' = \beta] - \frac{1}{2}$.

Definition 40. *A Secret-Key Anonymous Identity Based Encryption scheme is* key-secure *if all polynomial time adversaries achieve at most a negligible (in λ) advantage in the previous security game.*

Notice that no scheme with a deterministic KeyGen procedure can be key-secure.

10.4 Our construction for Secret-Key IBE

In this section we describe our construction for a Secret-key Anonymous IBE scheme which is similar to its public key version from the previous sections.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses random description $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ and random $Y_1, X_1, u \in \mathbb{G}_{p_1}, Y_3 \in \mathbb{G}_{p_3}, X_4, Y_4 \in \mathbb{G}_{p_4}$ and $\alpha \in \mathbb{Z}_N$. The *fictitious* public parameters are:

$$\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha).$$

The master secret key is $\mathsf{Msk} = (\mathsf{Pk}, X_1, \alpha)$.

$\mathsf{KeyGen}(\mathsf{Msk}, \mathsf{Id})$: The key generation algorithm chooses random $r \in \mathbb{Z}_N$ and also random elements $R_1, R_2 \in \mathbb{G}_{p_3}$ The secret key $\mathsf{Sk}_{\mathsf{Id}} = (K_1, K_2)$ is computed as

$$K_1 = Y_1^r R_1, \quad K_2 = Y_1^\alpha (u^{\mathsf{Id}} X_1)^r R_2.$$

$\mathsf{Encrypt}(\mathsf{Msk}, M, \mathsf{Id})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$ and random $Z, Z' \in \mathbb{G}_{p_4}$ The ciphertext (C_0, C_1, C_2) for the message $M \in \mathbb{G}_T$ is computed as

$$C_0 = M \cdot \mathbf{e}(Y_1, Y_1)^{\alpha s}, \quad C_1 = \left(u^{\mathsf{Id}} t\right)^s Z, \quad C_2 = Y_1^s Z'.$$

$\mathsf{Decrypt}(\mathsf{Msk}, \mathsf{Ct}, \mathsf{Sk})$: The decryption algorithm assumes that the key and ciphertext both correspond to the same identity Id. The decryption algorithm then computes the blinding factor similarly to the decryption procedure of the public-key version. Specifically,

$$\frac{\mathbf{e}(K_2, C_2)}{\mathbf{e}(K_1, C_1)} = \frac{\mathbf{e}(Y_1, Y_1)^{\alpha s} \mathbf{e}\left(u^{\mathsf{Id}} X_1, Y_1\right)^{rs}}{\mathbf{e}\left(Y_1, u^{\mathsf{Id}} X_1\right)^{rs}} = \mathbf{e}(Y_1, Y_1)^{\alpha s}.$$

10.5 Ciphertext Security

To prove ciphertext security of the Anonymous IBE scheme, we rely on the Assumptions I_1, I_2 and I_3 used in the proof of the public-key scheme.

We make the following considerations. If we instantiate the previous scheme as a public-key scheme by using the fictitious public-key parameter, it is identical to our public-key Anonymous IBE scheme (i.e., it is used in the non-hierarchical version) of the Chapter 9.1. Thus, it is immediate to verify that from Assumptions I_1, I_2 and I_3 the ciphertext security proof follows nearly identically from that in Section 9.2.5. Generally, if a public-key IBE encryption scheme is semantically secure, its secret-key version is also semantically secure because we can simulate the encryption oracle by using the public-key. Therefore, we have the following theorem.

Theorem 41. *If Assumptions I_1, I_2 and I_3 hold, then our Secret-Key Anonymous IBE scheme is ciphertext-secure.*

10.6 Key Security

We will use *semi-functional ciphertexts* and *semi-functional keys* like defined previously. These will not be used in the real scheme, but we need them in our proofs. We include them for completeness.

Semi-functional Ciphertext. We let g_2 denote a generator of $\mathbb{G}_{p_2}$. A semi-functional ciphertext is created as follows: first, we use the encryption algorithm to form a normal ciphertext (C_0', C_1', C_2'). We choose random exponents $x, z_c \in \mathbb{Z}_N$. We set:

$$C_0 = C_0', \quad C_1 = C_1' g_2^{x z_c}, \quad C_2 = C_2' g_2^{x}.$$

Semi-functional Keys. To create a semi-functional key, we first create a normal key (K_1', K_2') using the key generation algorithm. We choose random exponents $\gamma, z_k \in \mathbb{Z}_N$. We set:

$$K_1 = K_1' g_2^{\gamma}, \quad K_2 = K_2' g_2^{\gamma z_k}.$$

We note that when a semi-functional key is used to decrypt a semi-functional ciphertext, the decryption algorithm will compute the blinding factor multiplied by the additional term $\mathbf{e}(g_2, g_2)^{x\gamma(z_k - z_c)}$. If $z_c = z_k$, decryption will still work. In this case, we say that the key is nominally semi-functional.

To prove the security of our scheme we rely on static Assumptions I_1, I_2 and I_3. For a PPT adversary $\mathcal{A}$ which makes q ciphertext queries, our proof of security will consist of the following $q + 3$ games between $\mathcal{A}$ and a challenger $\mathcal{C}$.

$\mathsf{G}_{\mathsf{Real}}$: is the real key security game.

$\mathsf{G}_{\mathsf{Restricted}}$: is the same as $\mathsf{G}_{\mathsf{Real}}$ except that $\mathcal{A}$ cannot ask for keys for identities which are equal to one of the challenge identities modulo p_2. We will retain this restriction in all subsequent games.

G_k: for k from 0 to q, G_k is like $\mathsf{G}_{\mathsf{Restricted}}$, except that the key given to $\mathcal{A}$ is semi-functional and the first k ciphertexts are semi-functional. The rest of the ciphertexts are normal.

$\mathsf{G}_{\mathsf{Final}}$: is the same as G_q, except that the challenge key is semi-functional with K_2 random in $\mathbb{G}_{p_1p_2p_4}$ (thus the key is independent from the identities provided by $\mathcal{A}$). It is clear that in this last game, no adversary can have advantage greater than 0.

We will show these games are indistinguishable in the following lemmata.

10.6.1 Indistinguishability of $\mathsf{G}_{\mathsf{Real}}$ and $\mathsf{G}_{\mathsf{Restricted}}$

Lemma 42. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Real}}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Restricted}}} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage $\geq \frac{\epsilon}{3}$ in breaking Assumption I_1.*

Proof. The proof is identical to that given in lemma 33. □

10.6.2 Indistinguishability of $\mathsf{G}_{\mathsf{Restricted}}$ and G_0

Lemma 43. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Restricted}}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_0} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_1.*

Proof. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$ and T and simulates $\mathsf{G}_{\mathsf{Restricted}}$ or G_0 with $\mathcal{A}$ depending on whether $T \in \mathbb{G}_{p_1p_3}$ or $T \in \mathbb{G}_{p_1p_2p_3}$.

$\mathcal{B}$ sets the fictitious public parameters as follows. $\mathcal{B}$ chooses random exponents $\alpha, a, b, c \in \mathbb{Z}_N$ and sets $Y_1 = g_1, Y_3 = g_3, Y_4 = g_4$ $X_4 = Y_4^c, X_1 = Y_1^b$ and $u = Y_1^a$. $\mathcal{B}$ uses $\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha)$ to respond to the ciphertext queries issued by $\mathcal{A}$. Notice that $\mathcal{B}$ knows also the master secret key $\mathsf{Msk} = (\mathsf{Pk}, X_1, \alpha)$ and thus can simulate all $\mathcal{A}$'s key queries.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two identities, $\mathsf{Id}_0^\star$ and $\mathsf{Id}_1^\star$. $\mathcal{B}$ chooses random $\beta \in \{0,1\}$ and computes the challenge key as follows:

$$K_1 = T, \quad K_2 = Y_1^\alpha T^{a\mathsf{Id}_\beta^\star + b}.$$

We complete the proof with the following two observations. If $T \in \mathbb{G}_{p_1p_3}$, then T can be written as $Y_1^{s_1}Y_3^{s_3}$. In this case (K_1, K_2) is a normal key with randomness $r = s_1, R_1 = Y_3^{s_3}, R_2 = (Y_3^{s_3})^{a\mathsf{Id}_\beta^\star + b}$. If $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $Y_1^{s_1}g_2^{s_2}Y_3^{s_3}$ and this case (K_1, K_2) is a semi-functional key with randomness $r = s_1, R_1 = Y_3^{s_3}, R_2 = (Y_3^{s_3})^{a\mathsf{Id}_\beta^\star + b}$, $\gamma = s_2$ and $z_c = a\mathsf{Id}_\beta^\star + b$. Thus, in the former case we have properly simulated $\mathsf{G}_{\mathsf{Restricted}}$, and in the latter case we have simulated G_0. □

10.6.3 Indistinguishability of G_{k-1} and G_k

Lemma 44. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_k} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_1.*

PROOF. $\mathcal{B}$ receives $(\mathcal{I}, g_1, g_3, g_4, A_1A_2, B_2B_3)$ and T and simulates G_{k-1} or G_k with $\mathcal{A}$ depending on whether $T \in \mathbb{G}_{p_1p_3}$ or $T \in \mathbb{G}_{p_1p_2p_3}$.

$\mathcal{B}$ sets the fictitious public parameters by choosing random exponents $\alpha, a, b, c \in \mathbb{Z}_N$ and setting $Y_1 = g_1, Y_3 = g_4, Y_4 = g_3, X_4 = Y_4^c, X_1 = Y_1^b$ and $u = Y_1^a$. Notice that $\mathcal{B}$ knows the master secret key $\mathsf{Msk} = (\mathsf{Pk}, X_1, \alpha)$ with $\mathsf{Pk} = (N, Y_1, Y_3, Y_4, t = X_1X_4, u, \Omega = \mathbf{e}(Y_1, Y_1)^\alpha)$ and thus can respond to all $\mathcal{A}$'s key queries. Let us now explain how $\mathcal{B}$ answers the i-th ciphertext query for pair (M, Id).

For $i < k$, $\mathcal{B}$ creates a semi-functional ciphertext by choosing random exponents $s, w_1, w_2 \in \mathbb{Z}_N$ and setting:

$$C_0 = M\mathbf{e}(Y_1, Y_1)^{\alpha s}, \quad C_1 = (u^{\mathsf{Id}}X_1)^s(B_2B_3)^{w_1}, \quad C_2 = Y_1^sY_4^{w_2}$$

By writing B_2 as g_2^ϕ, we have that this is a properly distributed semi-functional ciphertext with $x = \phi$ and $z_c = w_1$.

For $i > k$, $\mathcal{B}$ runs the $\mathsf{Encrypt}$ algorithm using the master secret key $\mathsf{Msk} = (\mathsf{Pk}, X_1, \alpha)$.

To answer the k-th ciphertext query for (M_k, Id_k), $\mathcal{B}$ sets $z_c = a\mathsf{Id}_k + b$, chooses random exponent $w_1, w_2 \in \mathbb{Z}_N$, and sets:

$$C_0 = M_k\mathbf{e}(T, Y_1)^\alpha, \quad C_1 = T^{z_c}Y_4^{w_1}, \quad C_2 = TY_4^{w_2}$$

We have the following two observations. If $T \in \mathbb{G}_{p_1p_3}$, then T can be written as $Y_1^{r_1}Y_4^{r_4}$ In this case this is a properly distributed normal ciphertext with $s = r_1$. If $T \in \mathbb{G}_{p_1p_2p_3}$, then T can be written as $Y_1^{r_1}g_2^{r_2}Y_4^{r_4}$ and in this case it is a properly distributed semi-functional ciphertext with $x = r_2$.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two identities, $\mathsf{Id}_0^\star$ and $\mathsf{Id}_1^\star$. $\mathcal{B}$ chooses random $\beta \in \{0, 1\}$ and random $z, z' \in \mathbb{Z}_N$ and computes the challenge key as follows:

$$K_1 = (A_1A_2)Y_3^z, \quad K_2 = Y_1^\alpha(A_1A_2)^{a\mathsf{Id}_\beta^\star+b}Y_3^{z'}$$

This implicitly sets $Y_1^r = A_1$ and $z_k = a\mathsf{Id}_\beta^\star + b \mod p_2$. Since Id_k is not equal to $\mathsf{Id}_\beta^\star$ modulo p_2, we have that z_k and z_c are independent and randomly distributed.

We can thus conclude that, if $T \in \mathbb{G}_{p_1p_3}$ then $\mathcal{B}$ has properly simulated G_{k-1}. If $T \in \mathbb{G}_{p_1p_2p_3}$, then $\mathcal{B}$ has properly simulated G_k. □

10.6.4 Indistinguishability of G_q and $\mathsf{G}_{\mathsf{Final}}$

Lemma 45. *Suppose that there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_q} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{\mathsf{Final}}} = \epsilon$. Then there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption I_3.*

PROOF. First, notice that if exists an adversary $\mathcal{A}'$ which distinguishes an encryption for an identity $\mathsf{Id}_0^\star$ from an encryption for an identity $\mathsf{Id}_1^\star$, where $\mathsf{Id}_0^\star$ and $\mathsf{Id}_1^\star$ are chosen by $\mathcal{A}'$, then there exists an adversary $\mathcal{A}$ which distinguishes an encryption for an identity $\mathsf{Id}^\star$ chosen by $\mathcal{A}$ from an encryption for a random identity. Hence, we suppose that we are simulating the games for a such adversary.

$\mathcal{B}$ receives $(\mathcal{I}, g_1, g_2, g_3, g_4, U, U^s A_{24}, U^{\hat{r}}, A_1 A_4, A_1^{\hat{r}} A_2, g_1^{\hat{r}} B_2, g_1^s B_{24})$ and T and simulates G_q or $\mathsf{G}_{\mathsf{Final}}$ with $\mathcal{A}$ depending on whether $T = A_1^s D_{24}$ or T is random in $\mathbb{G}_{p_1 p_2 p_4}$.

$\mathcal{B}$ chooses random exponents $\alpha \in \mathbb{Z}_N$ and sets $Y_1 = g_1, Y_3 = g_4, Y_4 = g_3$.

Each time $\mathcal{B}$ is asked to provide a ciphertext for an identity Id, $\mathcal{B}$ creates a semi-functional ciphertext choosing random exponents $r, w_1, w_2 \in \mathbb{Z}_N$ and sets

$$C_0 = M \cdot \mathbf{e}(g_1^{\hat{r}} B_2, Y_1)^{\alpha s},\ C_1 = (A_1^{\hat{r}} A_2)^{r\mathsf{Id}} (U^{\hat{r}})^r Y_4^{w_1},\ C_2 = (g_1^{\hat{r}} B_2)^r Y_4^{w_2}$$

This implicitly sets the randomness of the ciphertext to $\hat{r}r$, $u = A_1$ and $X_1 = U$.

Each time $\mathcal{B}$ is asked to provide a key for an identity Id, $\mathcal{B}$ creates a semi-functional key choosing random exponents $r, w_1, w_2 \in \mathbb{Z}_N$ and setting:

$$K_1 = (g_1^{\hat{r}} B_2)^r Y_3^{w_1}, \quad K_2 = Y_1^\alpha (A_1^{\hat{r}} A_2)^{r\mathsf{Id}} (U^{\hat{r}})^r Y_3^{w_2}.$$

This implicitly sets the randomness of the secret key to $\hat{r}r$.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two identities, $\mathsf{Id}_0^\star$ and $\mathsf{Id}_1^\star$. $\mathcal{B}$ chooses random $w_1, w_2 \in \mathbb{Z}_N$ and computes the challenge secret key as follows:

$$K_1 = (g_1^s B_{24}) Y_3^{w_1}, \quad K_2 = Y_1^\alpha T^{\mathsf{Id}_\beta^\star} (U^s A_{24}) Y_3^{w_2}.$$

This implicitly sets γ and z_k to random values.

If $T = A_1^s D_{24}$, then this is properly distributed semi-functional key for identity $\mathsf{Id}_\beta^\star$. If T is a random element of $\mathbb{G}_{p_1 p_2 p_4}$, then this is a semi-functional key with K_2 random in $\mathbb{G}_{p_1 p_2 p_4}$.

Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T. □

10.6.5 $\mathsf{G}_{\mathsf{Final}}$ gives no advantage

Theorem 46. *If Assumptions I_1, I_2 and I_3 hold then our Anonymous IBE scheme is both ciphertext and key secure.*

PROOF. If the assumptions hold then we have proved by the previous lemmata that the real security game is indistinguishable from $\mathsf{G}_{\mathsf{Final}}$, in which the value of β is information-theoretically hidden from the attacker. Hence the attacker can obtain no non-negligible advantage in breaking the key security of the Secret-key Anonymous IBE scheme. We have showed previously that it is also ciphertext-secure. □

Chapter 11

Hierarchical HVE

In this chapter, we present a more sophisticated example of Hierarchical PE scheme. Specifically, we consider the case of Hierarchical HVE. We start by giving our definition of Hierarchical HVE (see also [38],[33]). Given $\vec{y}, \vec{w} \in \{0,1,\star\}^\ell$, we say that $\vec{w}$ is a *delegation* of $\vec{y}$, in symbols $\vec{w} \prec \vec{y}$, iff for each $i \in [\ell]$ we have $y_i = \star$ or $y_i = w_i$. For example $\langle 1,0,1,\star\rangle \prec \langle 1,0,\star,\star\rangle$. Notice that $\prec$ imposes a partial order on $\{0,1,\star\}^\ell$. A Hierarchical HVE scheme (HHVE) consists of five efficient algorithms (Setup, $\mathsf{Encrypt}$, KeyGen, Test, $\mathsf{Delegate}$). The semantics of Setup, $\mathsf{Encrypt}$, KeyGen and Test are identical to those given for HVE. The delegation algorithm has the following semantics.

$\mathsf{Delegate}(\mathsf{Pk}, \mathsf{Sk}_{\vec{y}}, \vec{y}, \vec{w})$: takes as input $\vec{y}, \vec{w} \in \{0,1,\star\}^\ell$ such that $\vec{w} \prec \vec{y}$ and secret key $\mathsf{Sk}_{\vec{y}}$ for $\vec{y}$ and outputs secret key $\mathsf{Sk}_{\vec{w}}$ for $\vec{w}$.

Correctness of HHVE. We require that for pairs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$, for all $y \in \{0,1,\star\}^\ell$, keys $\mathsf{Sk}_{\vec{y}} \leftarrow \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$, for all $\vec{w} \prec \vec{y}$ and all delegation paths $\vec{w} = \vec{w}_n \prec \ldots \prec \vec{w}_0 = \vec{y}$ of length $n \geq 0$ with $\mathsf{Sk}_{\vec{w}_i} = \mathsf{Delegate}(\mathsf{Pk}, \mathsf{Sk}_{\vec{w}_{i-1}}, \vec{w}_{i-1}, \vec{w}_i)$ for $i \in [n]$, and all $\vec{x} \in \{0,1\}^\ell$ we have $\mathsf{Test}(\mathsf{Pk}, \mathsf{Encrypt}(\mathsf{Pk}, \vec{x}), \mathsf{Sk}_{\vec{w}}) \neq \mathsf{Match}(\vec{x}, \vec{w})$ with probability negligible in λ.

Extensions of HHVE. The definition of HHVE given by Shi and Waters [38] is more general. In their model, beyond the special symbol $'\star'$ the alphabet associated with the key vectors also includes another special symbol $'?'$. Both of them allow to 'match' any symbol but only one of them is delegatable. That is, you may have a 'don't care' symbol which is not delegatable. It would be possible to extend our scheme to capture this model but we will not present the details.

Security of HHVE. The definition follows [38] and requires that no PPT adversary $\mathcal{A}$ has non-negligible advantage over 1/2 in game GReal against a challenger $\mathcal{C}$. More precisely, we have the following game.

Setup. $\mathcal{C}$ runs $(\mathsf{Pk}, \mathsf{Msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$ and starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering. Key queries can be of three different types. $\mathcal{C}$ answers these queries in the following way. $\mathcal{C}$ starts by initializing the set S of private keys that have been created but not yet given to $\mathcal{A}$ equal to $\emptyset$.

Create. $\mathcal{A}$ specifies a vector $\vec{y} \in \{0,1,\star\}^\ell$. In response, $\mathcal{C}$ creates $\mathsf{Sk}_{\vec{y}} = \mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$. $\mathcal{C}$ adds $\mathsf{Sk}_{\vec{y}}$ to the set S and gives $\mathcal{A}$ only a reference to it.

Delegate. $\mathcal{A}$ specifies a reference to a key $\mathsf{Sk}_{\vec{y}}$ in the set S and a vector $\vec{w} \in \{0,1,\star\}^\ell$ such that $\vec{w} \prec \vec{y}$. In response, $\mathcal{C}$ creates $\mathsf{Sk}_{\vec{w}} = \mathsf{Delegate}(\mathsf{Pk}, \mathsf{Sk}_{\vec{y}}, \vec{y}, \vec{w})$. $\mathcal{C}$ adds $\mathsf{Sk}_{\vec{w}}$ to S and gives $\mathcal{A}$ only a reference to it.

Reveal. $\mathcal{A}$ specifies an element of the set S. $\mathcal{C}$ gives the corresponding key to $\mathcal{A}$ and removes it from the set S. Then $\mathcal{A}$ needs no longer make any delegation queries for this key because it can run the $\mathsf{Delegate}$ algorithm by itself.

Challenge Construction. $\mathcal{A}$ specifies a pair $\vec{x}_0, \vec{x}_1 \in \{0,1\}^\ell$. $\mathcal{C}$ answers by picking random $\eta \in \{0,1\}$ and returning $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x}_\eta)$.

At the end of the game, $\mathcal{A}$ outputs a guess η' for η. We say that $\mathcal{A}$ *wins* if $\eta = \eta'$ and for all $\vec{y}$ for which $\mathcal{A}$ has seen a secret key, it holds that $\mathsf{Match}(\vec{x}_0, \vec{y}) = \mathsf{Match}(\vec{x}_1, \vec{y}) = 0$. The *advantage* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HHVE}}(\lambda)$ of $\mathcal{A}$ is defined to be the probability that $\mathcal{A}$ wins the game minus $1/2$.

Definition 47. *A Hierarchical Hidden Vector Encryption scheme is secure if for all probabilistic polynomial time adversaries $\mathcal{A}$, we have that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{HHVE}}(\lambda)$ is a negligible function of λ.*

Our construction for HHVE. As in HVE, we assume that the vectors $\vec{y}$ of the keys have at least two indices i, j such that $y_i, y_j \neq \star$.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization and random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3, R \in \mathbb{G}_{p_3}$, $g_4 \in \mathbb{G}_{p_4}$ and sets $g_{12} = g_1 \cdot g_2$. Then, for each $i \in [\ell]$ and $b \in \{0,1\}$, the setup chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$. The public parameters are $\mathsf{Pk} = \left[N, g_3, g_4, g_1 \cdot R, (T_{i,b})_{b\in\{0,1\}, i\in[\ell]}\right]$ and the master secret key is $\mathsf{Msk} = \left[g_{12}, (t_{i,b})_{b\in\{0,1\}, i\in[\ell]}\right]$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$: For each $i \in [\ell]$, the key generation algorithm chooses random $a_i \in Z_N$ such that $\sum_{i\in[\ell]} a_i = 0$ and random $R_i \in \mathbb{G}_{p_4}$. For $i \notin S_{\vec{y}}$ and $b \in \{0,1\}$, the algorithm chooses random $R_{i,b} \in \mathbb{G}_{p_4}$. Then for each $i \in [\ell]$, the key generation algorithm sets

$$Y_i = \begin{cases} g_{12}^{a_i/t_{i,y_i}} \cdot R_i, & \text{for } i \in S_{\vec{y}}; \\ g_{12}^{a_i} \cdot R_i, & \text{for } i \notin S_{\vec{y}}; \end{cases}$$

and, for each $i \notin S_{\vec{y}}$ and $b \in \{0,1\}$, $D_{i,b} = g_{12}^{a_i/t_{i,b}} \cdot R_{i,b}$. Finally the key generation algorithm returns the key $\mathsf{Sk}_{\vec{y}} = \left[(Y_i)_{i\in[\ell]}, (D_{i,b})_{i\notin S_{\vec{y}}, b\in\{0,1\}}\right]$.

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: The encryption algorithm chooses random $s \in \mathbb{Z}_N$ and $Z \in \mathbb{G}_{p_3}$ and, for each $i \in [\ell]$, random $Z_i \in \mathbb{G}_{p_3}$. The algorithm sets $X_0 = (g_1R)^s \cdot Z$ and, for each $i \in [\ell]$,

$X_i = (T_{i,x_i})^s Z_i$. The algorithm returns the ciphertext $\mathsf{Ct} = \left[X_0, (X_i)_{i\in[\ell]}\right]$. We stress that, unlike the HVE, a ciphertext for HHVE contains element X_0.

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{y}})$: Returns TRUE if $T = \mathbf{e}(X_0, \prod_{i\notin S_{\vec{y}}} Y_i) \cdot \prod_{i\in S_{\vec{y}}} \mathbf{e}(X_i, Y_i) = 1$.

$\mathsf{Delegate}(\mathsf{Pk}, \mathsf{Sk}_{\vec{y}}, \vec{y}, \vec{w})$: On input a secret key $\mathsf{Sk}_{\vec{y}} = \left[(Y'_i)_{i\in[\ell]}, (D'_{i,b})_{i\notin S_{\vec{y}}, b\in\{0,1\}}\right]$ for vector $\vec{y}$, the delegation algorithm chooses random $z \in \mathbb{Z}_N$. For $i \in S_{\vec{w}}$, the algorithm chooses random $R_i \in \mathbb{G}_{p_4}$ and, for $i \notin S_{\vec{w}}$ and $b \in \{0,1\}$, random $R_{i,b} \in \mathbb{G}_{p_4}$. The delegation algorithm for $i \in S_{\vec{w}}$ computes Y_i as

$$Y_i = \begin{cases} Y_i'^z R_i, & \text{if } y_i \neq \star; \\ D'^z_{i,w_i} R_i, & \text{if } y_i = \star. \end{cases}$$

Finally, for $i \notin S_{\vec{w}}$ and $b \in \{0,1\}$, the delegation algorithm sets $D_{i,b} = D'^z_{i,b} R_{i,b}$, and returns the key $\mathsf{Sk}_{\vec{w}} = \left[(Y_i)_{i\in[\ell]}, (D_{i,b})_{i\notin S_{\vec{w}}, b\in\{0,1\}}\right]$.

Notice that, for vectors $\vec{y}$ and $\vec{w}$ such that $\vec{w} \prec \vec{y}$, the distribution of the key $\mathsf{Sk}_{\vec{w}}$ for $\vec{w}$ output by KeyGen on input Msk and $\vec{w}$, and the distribution of the key for $\vec{w}$ output by $\mathsf{Delegate}$ on input Pk, a key $\mathsf{Sk}_{\vec{y}}$, $\vec{y}$ and $\vec{w}$ coincide. Therefore, the correctness of the scheme for keys generated by KeyGen is sufficient for proving correctness for every key. Furthermore, any delegation path starts from a secret key for a vector $\vec{y}$ created by running KeyGen. For any such a $\vec{y}$ and for any delegation path from $\vec{y}$ to $\vec{w}$, the distribution of the keys for $\vec{w}$, obtained by following the delegation path, is identical to the distribution of the keys for the same vector obtained by delegation directly from $\vec{y}$. Notice also that the distributions of $(\mathsf{Sk}_{\vec{y}}, \mathsf{Sk}_{\vec{w}})$ when $\mathsf{Sk}_{\vec{w}}$ is generated by KeyGen or by $\mathsf{Delegate}$ do differ. This makes the security proof more involved.

Correctness It is easy to verify the correctness of the scheme.

11.1 Complexity Assumptions for HHVE

To prove the security of our HHVE construction we rely on Assumptions 1 and 2 defined in Section 8.1. Furthermore we need the following Assumption 3.

Assumption 3 is a generalization of Assumption 2 (see Section 8.1) in the sense it posits the difficult of deciding if two triplets sharing an element are *both* Diffie-Hellman and it is defined as follows. First pick a random bilinear setting $\mathcal{I} = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^\lambda)$ and then pick $A_1 \leftarrow \mathbb{G}_{p_1}, A_2 \leftarrow \mathbb{G}_{p_2}, A_3 \leftarrow \mathbb{G}_{p_3}, A_4, B_4, C_4, D_4, E_4, F_4, G_4 \leftarrow \mathbb{G}_{p_4}$, $\alpha, \beta, \gamma \leftarrow \mathbb{Z}_{p_1}, T_2 \leftarrow \mathbb{G}_{p_1 p_4}$ and set $T_1 = A_1^{\alpha\beta} \cdot G_4$ and $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^\alpha \cdot B_4, A_1^\beta \cdot C_4, A_1^\gamma \cdot D_4, A_1^{\alpha\gamma} \cdot E_4, A_1^{\alpha\beta\gamma} \cdot F_4)$. We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be

$$\mathsf{Adv}_3^{\mathcal{A}}(\lambda) = |\text{Prob}[\mathcal{A}(D, T_1) = 1] - \text{Prob}[\mathcal{A}(D, T_2) = 1]|$$

Assumption 3 We say that Assumption 3 holds for generator $\mathcal{G}$ if for all probabilistic polynomial-time algorithms $\mathcal{A}$, $\mathsf{Adv}_3^{\mathcal{A}}(\lambda)$ is a negligible function of λ.

It is easy to see that Assumption 3 implies Assumption 2.

11.2 Security of our HHVE scheme

For a PPT adversary $\mathcal{A}$ which makes q *Reveal* key queries, our proof of security consists of $q+2$ games between $\mathcal{A}$ and $\mathcal{C}$. The first game, GReal, is the real HHVE security game described before. The description of the remaining games follows the same line of that defined to prove the security of our HVE scheme for $\xi = 0$.

Description of G_0. G_0 is like GReal, except that $\mathcal{C}$ uses g_2 instead of g_1 to construct the public parameters Pk given to $\mathcal{A}$.

Proof of indistinguishability of GReal and G_0.

Lemma 48. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}_{\mathsf{GReal}}^{\mathcal{A}} - \mathsf{Adv}_{\mathsf{G}_0}^{\mathcal{A}} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption* 1.

The proof follows the same line of that of Lemma 24 and we omit the details.

Description of G_k for $1 \le k \le q$. The Setup Phase and the Challenge Construction query of each of these games are like in G_0. The Key Queries are handled by $\mathcal{C}$ in the following way. $\mathcal{C}$ starts by initializing the set S to the empty set and the *query counter* v and the *reveal query counter* Rv equal to 0.

- *Create*$(\vec{y})$: $\mathcal{C}$ increments v and, for each $i \in [\ell]$, chooses random $a_{\mathsf{v},i} \in \mathbb{Z}_N$ such that $\sum_{i=1}^{\ell} a_{\mathsf{v},i} = 0$ and adds the tuple $(\mathsf{v}, \vec{y}, (a_{\mathsf{v},1}, \ldots, a_{\mathsf{v},\ell}))$ to the set S. $\mathcal{C}$ returns v to $\mathcal{A}$.

- *Delegate*$(\mathsf{v}', \vec{w})$: For *Delegate* key query on vector $\vec{w}$, $\mathcal{C}$ increments v and adds the tuple $(\mathsf{v}, \vec{w}, \mathsf{v}')$ to the set S. $\mathcal{C}$ returns v to $\mathcal{A}$.

- *Reveal*(v'): Suppose entry v' in S refers to key $\mathsf{Sk}_{\vec{w}}$ which is the the result of a delegation path $\vec{w} = \vec{w}_0 \prec \ldots \prec \vec{w}_n = \vec{y}$ of length $n \ge 0$ starting from key $\mathsf{Sk}_{\vec{y}}$ created as result of the v''-th *Create* key query. $\mathcal{C}$ chooses random $z \in \mathbb{Z}_N$ and, for each $i \in [\ell]$, random $c_i \in \mathbb{Z}_N$ and $R_i \in \mathbb{G}_{p_4}$. Moreover for each $i \notin S_{\vec{w}}$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $R_{i,b} \in \mathbb{G}_{p_4}$. $\mathcal{C}$ increments Rv. If $\mathsf{Rv} \le k$, then for each $i \in [\ell]$, $\mathcal{C}$ sets
$$Y_i = \begin{cases} g_1^{c_i} \cdot g_2^{z a_{\mathsf{v}'',i}/t_{i,w_i}} \cdot R_i, & \text{if} \quad i \in S_{\vec{w}}; \\ g_1^{c_i} \cdot g_2^{z a_{\mathsf{v}'',i}} \cdot R_i, & \text{if} \quad i \notin S_{\vec{w}}; \end{cases}$$
and, for each $i \notin S_{\vec{w}}$ and for each $b \in \{0,1\}$, $\mathcal{C}$ sets $D_{i,b} = g_1^{c_i} \cdot g_2^{z a_{\mathsf{v}'',i}/t_{i,b}} \cdot R_{i,b}$.

If instead $\mathsf{Rv} > k$, then for each $i \in [\ell]$, $\mathcal{C}$ sets

$$Y_i = \begin{cases} g_{12}^{za_{\mathsf{v}'',i}/t_{i,w_i}} \cdot R_i, & \text{if} \quad i \in S_{\vec{w}}; \\ g_{12}^{za_{\mathsf{v}'',i}} \cdot R_i, & \text{if} \quad i \notin S_{\vec{w}}; \end{cases}$$

and, for each $i \notin S_{\vec{w}}$ and for each $b \in \{0,1\}$, $\mathcal{C}$ sets $D_{i,b} = g_{12}^{za_{\mathsf{v}'',i}/t_{i,b}} \cdot R_{i,b}$. Finally, $\mathcal{C}$ returns the key $\mathsf{Sk}_{\vec{w}}$ consisting of the Y_i's and the $D_{i,b}$'s.

Proof of indistinguishability of G_{k-1} and G_k.

Lemma 49. *Suppose there exists a PPT algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_k} = \epsilon$. Then, there exists a PPT algorithm $\mathcal{B}$ with advantage at least $\epsilon/(2\ell)$ in breaking Assumption* 3.

Proof Sketch. $\mathcal{B}$ receives $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^{\alpha} \cdot B_4, A_1^{\beta} \cdot C_4, A_1^{\gamma} \cdot D_4, A_1^{\alpha\gamma} \cdot E_4, A_1^{\alpha\beta\gamma} \cdot F_4)$ and T and simulates G_{k-1} or G_k with $\mathcal{A}$ depending on T.

$\mathcal{B}$ starts by making the same guess as in the Lemma 25. Then:
Setup. As in that Lemma 25, by using D and by choosing appropriates randomness, $\mathcal{B}$ can prepare $\mathsf{Pk} = [N, g_3, g_4, g_1 R, (T_{i,b})_{i \in [\ell], b \in \{0,1\}}]$ where $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$.
Key Query Answering. $\mathcal{B}$ handles the *Create* and *Delegate* queries as prescribed in G_{k-1} and G_k. For the *Reveal* queries $\mathcal{B}$ proceeds as follow. We fix notation by denoting with $\vec{w}$ the vector for which $\mathcal{A}$ asks to reveal the key and we suppose that the key is part of a delegation path that starts from the key created by the v-th *Create* key query.

For the first $k-1$ *Reveal* key queries as follow. By using the randomness chosen during the creation of the v-th *Create* key query, $\mathcal{B}$ has all the necessary elements to computes a key with a random $\mathbb{G}_{p_1}$ part and a well formed $\mathbb{G}_{p_2}$ part as prescribed in both G_{k-1} and G_k

For the k-th *Reveal* key query, $\mathcal{B}$ handles it as follows. Again, as in Lemma 25, $\mathcal{B}$ can embed the challenge of the assumption in the guessed index j and then constructs the rest of the key accordingly in such a way if $T = A_1^{\alpha\beta} \cdot G_4$ then the answer to the k-th query of $\mathcal{A}$ is distributed as in G_{k-1}. Instead, if T is random in $\mathbb{G}_{p_1 p_4}$ then the answer to the k-th *Reveal* query is distributed as in G_k.

$\mathcal{B}$ handles the remaining $(q-k)$ *Reveal* queries as follows. We distinguish two cases depending on whether the key of the k-th *Reveal* key query is derived from the same *Create* key query and start from the the case in which it is not. In this case, it is easy for $\mathcal{B}$ to construct the secret key following the line of Lemma 25.

In the case in which the *Reveal* key query for $\vec{w}$ is for a key whose starting point in the delegation path corresponds to the v-th *Create* key query and it is the same as the one of the k-th *Reveal* key query, the key generation is more involved. In fact, during the creation of k-th *Reveal* key query, if T is equal to $A_1^{\alpha\beta} \cdot G_4$ then the key contains the randomness α and we must reuse this randomness together with β. But $\mathcal{B}$ can generates correctly the key by using $A_1^{\gamma} \cdot D_4, A_1^{\alpha\gamma} \cdot E_4, A_1^{\alpha\beta\gamma} \cdot F_4$ from D.

Challenge construction. As in Lemma 25 under a correct guess, Pk' contains all the values to compute an encryption of one randomly chosen challenge vector. Thus the challenge ciphertext is distributed as in G_{k-1} and G_k. □

G_q **gives no advantage**. We observe that in G_q the $\mathbb{G}_{p_1}$ part of the challenge ciphertext is the only one depending on η and the Pk and the answer to the key queries give no help to $\mathcal{A}$. Therefore we can conclude that for all adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{G}_q} = 0$. We have thus proved.

Theorem 50. *If Assumptions* 1 *and* 3 *hold, then our HHVE scheme is secure.*

Chapter 12

HVE secure against Unrestricted Adversaries

In this Chapter we present the main result of this thesis: a Predicate Encryption scheme for the predicate HVE that is secure against unrestricted adversaries.

12.1 Proof technique

Our result is based on the dual system encryption methodology introduced by Waters [40] and gives extra evidence of the power of this proof technique. However, to overcome the difficulty of having to deal also with matching queries, we have to carefully look at the space of matching queries and at how they relate to the challenge vectors. This enables us to craft a new security game in which the challenge ciphertext is constructed in a way that guarantees that keys obtained by the adversary give the expected result when tested against the challenge ciphertext and, at the same time, the challenge ciphertext is independent from the challenge vector used to construct it. Then we show, by means of a sequence of intermediate security games, that the real security game is computationally indistinguishable from this new game. It is not immediate to obtain a similar relation between matching queries and challenge vectors for other predicates (e.g., inner product) that would make our approach viable.

12.2 Complexity Assumptions for the Unrestricted construction

The complexity assumptions used to prove the security against unrestricted adversaries of our construction are Assumption 1 and Assumption 2 defined in Section 8.1.

12.3 Constructing HVE

Our HVE scheme that will be proved secure against unrestricted adversaries is the same scheme presented in Section 8.2.

Regarding this scheme, we make the following remark.

Remark 51. *Let* $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ *and* $\mathsf{Msk} = [g_1 \cdot g_2, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ *be a pair of public parameter and master secret key output by the* Setup *algorithm and consider* $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ *and* $\mathsf{Msk}' = [\hat{g}_1 \cdot g_2, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ *with* $T'_{i,b} = \hat{g}_1^{t_{i,b}} \cdot R'_{i,b}$ *for some* $\hat{g}_1 \in \mathbb{G}_{p_1}$ *and* $R'_{i,b} \in \mathbb{G}_{p_3}$*. We make the following easy observations.*

1. *For every* $\vec{y} \in \{0, 1, \star\}^\ell$*, the distributions* $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$ *and* $\mathsf{KeyGen}(\mathsf{Msk}', \vec{y})$ *are identical.*
2. *Similarly, for every* $\vec{x} \in \{0, 1\}^\ell$*, the distributions* $\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$ *and* $\mathsf{Encrypt}(\mathsf{Pk}', \vec{x})$ *are identical.*

12.4 Security of our HVE scheme

We start by giving an informal description of the idea behind our proof of security and show how we overcome the main technical difficulty of having to deal with matching keys.
The first step in our proof strategy consists in 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, we defined a new security game GPK in which the public key is constructed so that all relevant information (that is, the $t_{i,b}$'s) is encoded in the $\mathbb{G}_{p_2}$ parts of the public key. The challenge ciphertext and the answers to the key queries are instead constructed as in the real security game GReal. Thus, ciphertexts constructed by the adversary are completely independent from the challenge ciphertext (as they encode information in two different subgroups). The view of an adversary in GPK is still indistinguishable from the the view of the real security game GReal. We observe that since keys are constructed as in the real security game, they carry information about $\vec{y}$ both in the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ parts. Thus when the adversary tests a ciphertext he has constructed by using the public key against a key obtained by means of a query, he obtains the expected result because of the information encoded in the $\mathbb{G}_{p_2}$ part of the key and of the ciphertext. The same holds for the challenge ciphertext but now thanks to the $\mathbb{G}_{p_1}$ part of the keys. The only difference is in the public keys but, under Assumption 1 (a natural subgroup decision hardness assumption), we can prove that the view remains indistinguishable.
The second step proves that the keys obtained from queries do not help the adversary. Since the challenge ciphertext carries information about the randomly selected challenge vector $\vec{x}_\eta$ in its $\mathbb{G}_{p_1}$ part, in this informal discussion when we refer to key we mean its $\mathbb{G}_{p_1}$ part. The $\mathbb{G}_{p_2}$ parts of the keys are always correctly computed.

In our construction, testing a ciphertext against a non-matching key gives a random value (from the target group) whereas testing it against a matching key returns a specified value (the identity of the target group). One possible avenue for proving security against an adversary that asks only non-matching queries is to show that replying key queries by returning random elements from the appropriate subgroups instead of well-formed keys, gives a new game which is indistinguishable from the real security game and in which it is easy to prove that the adversary can win with probability essentially $1/2$. This approach fails for matching queries as a random reply to a matching query is unlikely to return the correct answer when tested against the challenge ciphertext. Instead we modify the construction of the challenge ciphertext in the following way: the challenge ciphertext is well-formed in all the positions where the two challenge vectors are equal and random in all the other positions. We observe that testing such a challenge ciphertext against matching and non-matching keys always gives the correct answer and that no adversary (even an all powerful one) can guess which of the two challenge vectors has been used to construct the challenge ciphertext.

We thus prove security of our construction by proving that the above game (that is called $\mathsf{GBadCh}(\ell+1)$) is indistinguishable from GPK. We achieve this by means of $\ell+1$ intermediate games: $\mathsf{GBadCh}(1) = \mathsf{GPK}, \dots, \mathsf{GBadCh}(\ell+1)$ where in each game we switch to random one more component of the challenge ciphertext which corresponds to position in which the challenge vectors differ. Now let us consider two consecutive games $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$. It is easy to see that if the challenge vectors coincide in the f-th component the two games are exactly the same. Let us now discuss the case in which the two games differ in the f-th component. To do so, we consider intermediate games in which we modify the answer to the queries but, for the reasons discussed above, we cannot naively reply randomly to the queries. Rather, for adversaries that ask q key queries, we consider $q+1$ intermediate games $\mathsf{GBadQ}(f,0) = \mathsf{GBadCh}(f), \dots, \mathsf{GBadQ}(f,q)$. In the k-th intermediate game we alter the distribution of the reply to the first k key queries based on the the following observation that relates matching queries to the challenge vectors: *if the challenge vectors differ in the f-th component, a query is matching only if it has a $\star$ in position f.* Thus, if the key has a $\star$ in position f, then its f-th component is empty and this is easy to simulate. If instead a query has a non-$\star$ in the position f, then it must be a non matching query and thus it safe to randomize the reply. Using Assumption 2, we prove that this modification in the answers to the key queries still guarantees indistinguishability.

At this point, we would like to prove that $\mathsf{GBadQ}(f,q)$ is indistinguishable from $\mathsf{GBadQ}(f+1,0)$ (which is equal to $\mathsf{GBadCh}(f+1)$). Unfortunately, this is not the case and we need to resort to extra security games $\mathsf{GBadQ2}(f,q), \dots, \mathsf{GBadQ2}(f,0)$ to complete the proof.

12.4.1 The first step of the proof

We start by defining $\mathsf{GPK}(\lambda,\ell)$ that differs from $\mathsf{GReal}(\lambda,\ell)$ as in the Setup phase, $\mathcal{C}$ prepares two sets of public parameters, Pk and Pk', and one master secret key Msk. Pk is given as input to $\mathcal{A}$, Msk is used to answer $\mathcal{A}$'s key queries and Pk' is used to construct the challenge ciphertext. Specifically,

Setup. $\mathcal{C}$ chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. $\mathcal{C}$ chooses 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 sets $g_{12} = g_1 \cdot g_2$. For each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{C}$ chooses random $t_{i,b} \in Z_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T'_{i,b} = g_1^{t_{i,b}} \cdot R_{i,b}$ and $T_{i,b} = g_2^{t_{i,b}} \cdot R_{i,b}$. Then $\mathcal{C}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$. Finally, $\mathcal{C}$ starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering$(\vec{y})$. $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$.

Challenge Query Answering$(\vec{x}_0, \vec{x}_1)$. Upon receiving the pair $(\vec{x}_0, \vec{x}_1)$ of challenge vectors, $\mathcal{C}$ picks random $\eta \in \{0,1\}$ and returns the output of $\mathsf{Encrypt}(\mathsf{Pk}', \vec{x}_\eta)$.

Winning Condition. Like in $\mathsf{GReal}(\lambda, \ell)$.

The next lemma shows that, the advantages of an adversary in $\mathsf{GReal}(\lambda, \ell)$ and $\mathsf{GPK}(\lambda, \ell)$ are the same, up to a negligible factor.

Lemma 52. *If Assumption 1 holds, then for any PPT adversary $\mathcal{A}$,*
$\left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GReal}(\lambda, \ell)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}(\lambda, \ell)]\right|$ *is negligible.*

Proof. We show a PPT algorithm $\mathcal{B}$ which receives $(\mathcal{I}, A_3, A_4, A_{13}, A_{12})$ and T and, depending on the nature of T, simulates $\mathsf{GReal}(\lambda, \ell)$ or $\mathsf{GPK}(\lambda, \ell)$ with $\mathcal{A}$. This suffices to prove the Lemma.

Setup. $\mathcal{B}$ starts by constructing public parameters Pk and Pk' in the following way. $\mathcal{B}$ sets $g_{12} = A_{12}, g_3 = A_3, g_4 = A_4$ and, for each $i \in [\ell]$ and $b \in \{0,1\}$, $\mathcal{B}$ chooses random $t_{i,b} \in \mathbb{Z}_N$ and sets $T_{i,b} = T^{t_{i,b}}$ and $T'_{i,b} = A_{13}^{t_{i,b}}$. Then $\mathcal{B}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, $\mathsf{Msk} = [g_{12}, g_4, (t_{i,b})_{i\in[\ell],b\in\{0,1\}}]$, and $\mathsf{Pk}' = [N, g_3, (T'_{i,b})_{i\in[\ell],b\in\{0,1\}}]$ and starts the interaction with $\mathcal{A}$ on input Pk.

Key Query Answering$(\vec{y})$. $\mathcal{B}$ runs algorithm KeyGen on input Msk and $\vec{y}$.

Challenge Query Answering$(\vec{x}_0, \vec{x}_1)$. The challenge is created by $\mathcal{B}$ by picking random $\eta \in \{0,1\}$ and running the $\mathsf{Encrypt}$ algorithm on input $\vec{x}_\eta$ and Pk'.

This concludes the description of algorithm $\mathcal{B}$.

Now suppose $T \in \mathbb{G}_{p_1p_3}$, and thus it can be written as $T = h_1 \cdot h_3$ for $h_1 \in \mathbb{G}_{p_1}$ and $h_3 \in \mathbb{G}_{p_3}$. This implies that Pk received in input by $\mathcal{A}$ in the interaction with $\mathcal{B}$ has the same distribution as in GReal. Moreover, by writing A_{13} as $A_{13} = \hat{h}_1 \cdot \hat{h}_3$ for $\hat{h}_1 \in \mathbb{G}_{p_1}$ and $\hat{h}_3 \in \mathbb{G}_{p_3}$ which is possible since by assumption $A_{13} \in \mathbb{G}_{p_1p_3}$, we notice that that Pk and Pk' are as in the hypothesis of Remark 51 (with $g_1 = h_1$ and $\hat{g}_1 = \hat{h}_1$). Therefore the answers to key queries and the challenge ciphertext given by $\mathcal{B}$ to $\mathcal{A}$ have the same distribution as the answers and the challenge ciphertext received by $\mathcal{A}$ in $\mathsf{GReal}(\lambda, \ell)$. We can thus conclude that, when $T \in \mathbb{G}_{p_1p_3}$, $\mathcal{C}$ has simulates $\mathsf{GReal}(\lambda, \ell)$ with $\mathcal{A}$.

Let us discuss now the case $T \in \mathbb{G}_{p_2p_3}$. In this case, Pk provided by $\mathcal{B}$ has the same distribution as the public parameters produced by $\mathcal{C}$ in $\mathsf{GPK}(\lambda, \ell)$. Therefore, $\mathcal{C}$ is simulating $\mathsf{GPK}(\lambda, \ell)$ for $\mathcal{A}$.

This concludes the proof of the lemma. $\square$

12.4.2 The second step of the proof

We start the second step of the proof by describing $\mathsf{GBadQ}(f, k)$, for $1 \le f \le \ell + 1$ and $0 \le k \le q$. Not to overburden the notation, we omit λ and ℓ from the name of the games. $\mathsf{GBadQ}(f, k)$ differs from GPK both in the way in which key queries are answered and in the way in which the challenge ciphertext is constructed. More precisely, in $\mathsf{GBadQ}(f, k)$ the first k key queries are answered by $\mathcal{C}$ by distinguishing two cases. Queries for $\vec{y}$ such that $y_f = \star$ are answered by running $\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$. Instead queries for $\vec{y}$ such that $y_f \neq \star$ are answered by returning keys whose $\mathbb{G}_{p_1}$ part is random for all components which correspond to non-$\star$ entries. Moreover, in $\mathsf{GBadQ}(f, k)$, the first $f - 1$ components of the challenge ciphertext corresponding to positions in which the two challenges differ are random in $\mathbb{G}_{p_1p_3}$. Let us now formally describe $\mathsf{GBadQ}(f, k)$.

Setup. Like in GPK.

Key Query Answering$(\vec{y})$. $\mathcal{C}$ answers the first k queries in the following way. If $y_f \neq \star$, $\mathcal{C}$ chooses, for each $i \in S_y$ random $W_i \in \mathbb{G}_{p_4}$, random $C_i \in \mathbb{G}_{p_1}$ and random $a_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i \in S_y} a_i = 0$ and sets $Y_i = C_i \cdot g_2^{a_i/t_{i,y_i}} \cdot W_i$. If $y_f = \star$ then $\mathcal{C}$ returns the output of $\mathsf{KeyGen}(\vec{y}, \mathsf{Msk})$. The remaining $q - k$ queries are answered by running $\mathsf{KeyGen}(\vec{y}, \mathsf{Msk})$.

Challenge Query Answering$(\vec{x}_0, \vec{x}_1)$. $\mathcal{C}$ chooses random $s \in \mathbb{Z}_N$ and $\eta \in \{0, 1\}$ and sets $\vec{x} = \vec{x}_\eta$. For each $i \in [f-1]$ such that $\vec{x}_{0,i} \neq \vec{x}_{1,i}$, $\mathcal{C}$ chooses random $X_i \in \mathbb{G}_{p_1p_3}$. Then, for each remaining i, $\mathcal{C}$ chooses random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T_{i,x_i}^s \cdot Z_i$. $\mathcal{C}$ returns the tuple $(X_i)_{i\in[\ell]}$.

Winning Condition. Like in GReal.

In the proofs, we will use the shorthand $\mathsf{GBadCh}(f)$ for $\mathsf{GBadQ}(f, 0)$. Moreover, we define $\mathsf{GBadQ2}(f, k)$, for $1 \le f \le \ell$ and $0 \le k \le q$, as a game in which the setup phase is like in $\mathsf{GBadQ}(f, k)$, key queries are answered like in $\mathsf{GBadQ}(f, k)$ and the challenge ciphertext is constructed like in $\mathsf{GBadQ}(f + 1, k)$.

Some simple observations about GBadQ and GBadQ2

Observation 53. $\mathsf{GPK} = \mathsf{GBadQ}(1, 0)$.
Straightforward from the definitions of the games.

Observation 54. $\mathsf{GBadQ}(f, q) = \mathsf{GBadQ2}(f, q)$ *for* $f = 1, \ldots, \ell$.
From the definitions of the two games, it is clear that all key queries are answered in the same way in both the games 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 $\mathsf{GBadQ}(f, q)$, 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 $\mathsf{GBadQ2}(f, q)$ when $x_{0,f} = x_{1,f}$. On the other hand, when $x_{0,f} \neq x_{1,f}$, 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 position f of the answer is random since $k = q$). 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 55. $\mathsf{GBadQ2}(f, 0) = \mathsf{GBadQ}(f + 1, 0)$ *for* $f = 1, \ldots, \ell - 1$.
Indeed, in both games all key queries are answered correctly, and the challenge query in $\mathsf{GBadQ2}(f, 0)$ is by definition answered in the same way as in $\mathsf{GBadQ}(f + 1, 0)$.

Observation 56. *All adversaries have no advantage in* $\mathsf{GBadCh}(\ell+1) = \mathsf{GBadQ}(\ell+1, 0)$.
This follows from the fact that, for positions i such that $x_{0,i} \neq x_{1,i}$, the $\mathbb{G}_{p_1}$ part of X_i is random. Thus, the challenge ciphertext of $\mathsf{GBadCh}(\ell + 1)$ is independent from η.

Description of simulator $\mathcal{S}$

In this section, we describe a PPT simulator $\mathcal{S}$ that interacts with an adversary $\mathcal{A}$ and will be used in our proof.
Input. Integers $1 \leq f \leq \ell + 1$ and $0 \leq k \leq q$, and a randomly chosen instance (D, T) of Assumption 2; recall that $D = (\mathcal{I}, A_1, A_2, A_3, A_4, A_1^\alpha \cdot B_4, A_1^\beta \cdot C_4)$ and $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ or $T = T_2$ random in $\mathbb{G}_{p_1p_4}$.
Setup. To simulate the Setup phase $\mathcal{S}$ executes the following steps.

1. $\mathcal{S}$ sets $g_1 = A_1$, $g_2 = A_2$, $g_3 = A_3$, $g_4 = A_4$ and $g_{12} = A_1 \cdot A_2$.
2. For each $i \in [\ell]$ and $b \in \{0, 1\}$,
   $\mathcal{S}$ chooses random $v_{i,b} \in \mathbb{Z}_N$ and $R_{i,b} \in \mathbb{G}_{p_3}$, and sets $T_{i,b} = g_2^{v_{i,b}} \cdot R_{i,b}$.
3. $\mathcal{S}$ sets $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell],b\in\{0,1\}}]$.
4. $\mathcal{S}$ picks random $\hat{\jmath} \in [\ell]$ and $\hat{b} \in \{0, 1\}$ and sets $\hat{c} = 1 - \hat{b}$.
5. For each $i \in [\ell] \setminus \{\hat{\jmath}\}$ and $b \in \{0, 1\}$,
   $\mathcal{S}$ chooses random $r_{i,b} \in \mathbb{Z}_N$ and set $T'_{i,b} = g_1^{r_{i,b}}$.
6. $\mathcal{S}$ chooses random $r_{\hat{\jmath},\hat{c}} \in \mathbb{Z}_N$ and set $T'_{\hat{\jmath},\hat{c}} = g_1^{r_{\hat{\jmath},\hat{c}}}$.
   The value of $r_{\hat{\jmath},\hat{b}}$ remains unspecified. As we shall see below, in answering key queries, $\mathcal{S}$ will implicitly set $r_{\hat{\jmath},\hat{b}} = 1/\beta$. We stress that β is the same exponent appearing in $A_1^\beta \cdot C_4$ from the istance of Assumption 2 and that $\mathcal{S}$ does not have access to the actual value of β.

$\mathcal{S}$ starts the interaction with $\mathcal{A}$ on input Pk.
Key Query Answering$(\vec{y})$**.** To describe how $\mathcal{S}$ answers the first $k - 1$ queries, we distinguish the following two mutually exclusive cases.

Case A.1: $y_f \neq \star$**.** In this case, $\mathcal{S}$ outputs a key whose $\mathbb{G}_{p_1}$ part is random. More precisely, $\mathcal{S}$ executes the following steps. For each $i \in S_{\vec{y}}$, $\mathcal{S}$ chooses random a''_i such that $\sum_{i\in S_{\vec{y}}} a''_i = 0$, random $C_i \in \mathbb{G}_{p_1}$, and random $W_i \in \mathbb{G}_{p_4}$. Then, for each $i \in S_{\vec{y}}$, $\mathcal{S}$ sets

$$Y_i = C_i \cdot g_2^{a''_i/v_{i,y_i}} \cdot W_i.$$

Case A.2: $y_f = \star$. In this case, $\mathcal{S}$ tries to output a key that has the same distribution induced by algorithm KeyGen on input $\vec{y}$. We observe that if $y_{\hat{\jmath}} = \hat{c}$ then $\mathcal{S}$ knows all the r_{i,y_i}'s and v_{i,y_i}'s needed. If instead $y_{\hat{\jmath}} = \hat{b}$, then $\mathcal{S}$ is missing $r_{\hat{\jmath},\hat{b}}$. In this case $\mathcal{S}$ computes $Y_{\hat{\jmath}}$ by using $A_1^\beta \cdot C_4$ from the challenge of Assumption 2 received in input.

More precisely, for each $i \in S_{\vec{y}}$, $\mathcal{S}$ picks random $W_i \in \mathbb{G}_{p_4}$ and random $a'_i, a''_i \in \mathbb{Z}_N$ under the constraint that $\sum_{i\in S_{\vec{y}}} a'_i = \sum_{i\in S_{\vec{y}}} a''_i = 0$. Then for each $i \neq \hat{\jmath}$, $\mathcal{S}$ sets

$$Y_i = g_1^{a'_i/r_{i,y_i}} \cdot g_2^{a''_i/v_{i,y_i}} \cdot W_i.$$

Moreover, if $y_{\hat{\jmath}} = \hat{c}$, $\mathcal{S}$ sets

$$Y_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}}/r_{\hat{\jmath},\hat{c}}} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{c}}} \cdot W_{\hat{\jmath}}$$

otherwise, if $y_{\hat{\jmath}} = \hat{b}$, $\mathcal{S}$ sets

$$Y_{\hat{\jmath}} = (A_1^\beta \cdot C_4)^{a'_{\hat{\jmath}}} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}}\beta} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot (C_4^{a'_{\hat{\jmath}}} \cdot W_{\hat{\jmath}}).$$

Notice that this setting implicitly defines $r_{\hat{\jmath},\hat{b}} = 1/\beta$ which remains unknown to $\mathcal{S}$.

Let us now describe how $\mathcal{S}$ answers the k-th query for vector $\vec{y}^{(k)} = (y_1^{(k)}, \ldots, y_\ell^{(k)})$. We have three cases and we let $\mathsf{Guess}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ denote the event that $\mathcal{S}$, on input f and k and interacting with $\mathcal{A}$, does not abort in computing the answer to the k-th query.

Case B.1: $y_f^{(k)} = \star$. $\mathcal{S}$ performs the same steps of Case A.2.

Case B.2: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} \neq \hat{b}$. $\mathcal{S}$ outputs $\perp$ and aborts.

Case B.3: $y_f^{(k)} \neq \star$ **and** $y_{\hat{\jmath}}^{(k)} = \hat{b}$. Let $S = S_{\vec{y}} \setminus \{\hat{\jmath}, h\}$, where h is an index such that $y_h^{(k)} \neq \star$. Such an index h always exists since we assumed that each query contains at least two non-$\star$ entries. Then, for each $i \in S$, $\mathcal{S}$ chooses random $W_i \in \mathbb{G}_{p_4}$ and random $a'_i, a''_i \in \mathbb{Z}_N$ and sets

$$Y_i = g_1^{a'_i/r_{i,y_i^{(k)}}} \cdot g_2^{a''_i/v_{i,y_i^{(k)}}} \cdot W_i.$$

$\mathcal{S}$ then chooses random $a''_{\hat{\jmath}} \in \mathbb{Z}_N$ and $W_{\hat{\jmath}}, W_h \in \mathbb{G}_{p_4}$ and sets

$$Y_{\hat{\jmath}} = T \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot W_{\hat{\jmath}} \quad \text{and} \quad Y_h = (A_1^\alpha B_4)^{-1/r_{h,y_h^{(k)}}} \cdot g_1^{-s'/r_{h,y_h^{(k)}}} \cdot g_2^{-(s''+a''_{\hat{\jmath}})/v_{h,y_h^{(k)}}} \cdot W_h,$$

where $s' = \sum_{i\in S} a'_i$ and $s'' = \sum_{i\in S} a''_i$.

This terminates the description of how $\mathcal{S}$ handles the k-th key query.

$\mathcal{S}$ handles the remaining $q-k$ queries as in Case A.2, independently from whether $y_f = \star$ or $y_f \neq \star$. More precisely, if $y_{\hat{\jmath}} = \hat{c}$ then $\mathcal{S}$ has all the r_{i,y_i}'s and v_{i,y_i}'s needed. On the other hand, if this is not the case then $\mathcal{S}$ can use $A_1^{\beta} \cdot C_4$ from D.
Challenge Query Answering($\vec{x}_0, \vec{x}_1$). If $\vec{x}_0$ and $\vec{x}_1$ coincide on the f-th component or $y_{\hat{\jmath}}^{(k)} = x_{\eta,\hat{\jmath}}$, $\mathcal{S}$ aborts. We let $\mathsf{Guess}_{2,\mathcal{S}}^{\mathcal{A}}(f,k)$ denote the event that $y_{\hat{\jmath}}^{(k)} \neq x_{\eta,\hat{\jmath}}$ while $\mathcal{S}$ is interacting with $\mathcal{A}$ on input f and k. We observe that if $\mathsf{Guess}_{2,\mathcal{S}}^{\mathcal{A}}(f,k)$ occurs, $x_{\eta,\hat{\jmath}} = \hat{c} = 1-\hat{b}$.

If $\mathcal{S}$ has not aborted, $\mathcal{S}$ picks random $\eta \in \{0,1\}$, sets $\vec{x} = \vec{x}_\eta$ and creates the challenge by running algorithm $\mathsf{Encrypt}$ on input the challenge vector $\vec{x}$, public parameters Pk' and randomizing the $\mathbb{G}_{p_1}$ part of all X_i for $i < f$ such that $x_{0,i} \neq x_{1,i}$. More precisely, the challenge ciphertext is created as follows. $\mathcal{S}$ chooses random $s \in \mathbb{Z}_N$. For each $i \in [f-1]$ such that $x_{0,i} \neq x_{1,i}$, $\mathcal{S}$ sets r_i equal to a random element in $\mathbb{Z}_N$. $\mathcal{S}$ sets $r_i = 1$ for all remaining i's. For each $i \in [\ell]$, $\mathcal{S}$ picks random $Z_i \in \mathbb{G}_{p_3}$ and sets $X_i = T_{i,x_i}^{\prime s r_i} \cdot Z_i$, and returns the tuple $(X_i)_{i \in [\ell]}$.

Two remarks are in order. First, if $\mathcal{S}$ has not aborted, Pk' contains all the values T_{i,x_i} needed for computing an encryption of $\vec{x}$. Second, as a sanity check, we verify that $\mathcal{S}$ cannot test the nature of T and thus break Assumption 2. Indeed to do so, $\mathcal{S}$ should use T to generate a key for $\vec{y}$ and ciphertext for $\vec{x}$ such that $\mathsf{Match}(\vec{x}, \vec{y}) = 1$. Then, if $T = T_1$ the Test procedure will have success; otherwise, it will fail. In constructing the key, $\mathcal{S}$ would use T to construct the $\hat{\jmath}$-th component (which forces $y_{\hat{\jmath}} = \hat{b}$) and then it would need $r_{\hat{\jmath},\hat{b}}$ to construct the matching ciphertext. However, $\mathcal{S}$ does not have access to this value.

The simulator $\mathcal{S}$ described will be used to prove properties of games GBadQ. We can modify the simulator $\mathcal{S}$ so that, on input f and k, the challenge cihpertext is constructed by randomizing the $\mathbb{G}_{p_1}$ part also of the f-th component. The so modified simulator, that we call $\mathcal{S}_2$, closely simulates the work of games $\mathsf{GBadQ2}$ and will be used to prove properties of these games.

Properties of simulator $\mathcal{S}$

We now state and prove some properties of $\mathcal{S}$ that will be used in our security proof. We start by defining event $E_{f,G}^{\mathcal{A}}$ as the event that in game G the adversary $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. When the adversary $\mathcal{A}$ is clear from the context we will simply write $E_{f,G}$. We use notation E_f to denote $E_{f,G}$ for $G = \mathsf{GPK}$.

We extend the definition of $E_{f,G}$ to include the game played by $\mathcal{A}$ against the simulator $\mathcal{S}$. Thus we denote by $E_{f,\mathcal{S}}^{\mathcal{A}}(f',k)$ the event that in the interaction between $\mathcal{A}$ and $\mathcal{S}$ on input f' and k, $\mathcal{A}$ declares two challenge vectors that differ in the f-th component. If $\mathcal{A}, f'$ and k are clear from the context, we will simply write $E_{f,\mathcal{S}}$. With this notation in place, the event "$\mathcal{S}$ does not abort while interacting with $\mathcal{A}$ on input f and k" is equal to the event

$$\mathsf{Guess}_{1,\mathcal{S}}^{\mathcal{A}}(f,k) \wedge \ \mathsf{Guess}_{2,\mathcal{S}}^{\mathcal{A}}(f,k) \wedge E_{f,\mathcal{S}}^{\mathcal{A}}(f,k). \tag{12.1}$$

In addition, we observe that event $E^{\mathcal{A}}_{f,\mathcal{S}}(f,k)$ implies event $\mathsf{Guess}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ and similarly does event $\mathsf{Guess}^{\mathcal{A}}_{2,\mathcal{S}}(f,k)$. We modify the challenger $\mathcal{C}$ so that at the beginning of the interaction with $\mathcal{A}$, $\mathcal{C}$ picks $\hat{\jmath}$ and $\hat{b}$ just like $\mathcal{S}$ does. This modification makes the definitions of events $\mathsf{Guess}^{\mathcal{A}}_{1,G}$ and $\mathsf{Guess}^{\mathcal{A}}_{2,G}$ meaningful. Notice that, unlike the simulator $\mathcal{S}$, the challenger never aborts its interaction with $\mathcal{A}$ and that this modification does not affect $\mathcal{A}$'s view. This implies, for example, that the fact that event $E^{\mathcal{A}}_{f,\mathbb{G}}$ has occurred during a game G does not necessarily imply that event $\mathsf{Guess}^{\mathcal{A}}_{1,G}$ also occurs. We write $\mathsf{Guess}^{\mathcal{A}}_{2}$ as a shorthand for $\mathsf{Guess}^{\mathcal{A}}_{2,G}$ with $G = \mathsf{GPK}$.

Lemma 57. *For all f,k and $\mathcal{A}$,* $\mathrm{Prob}[\mathsf{Guess}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)] \geq \frac{1}{\ell}$.

PROOF. It is easy to see that the probability of $\mathsf{Guess}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ is at least the probability that $y^{(k)}_{\hat{\jmath}} = \hat{b}$. Moreover, the view of $\mathcal{A}$ up to the k-th key query is independent from $\hat{b}$ and $\hat{\jmath}$. Now observe that $\vec{y}^{(k)}$ has at least two non-star entry and, provided that if $\hat{\jmath}$ is one of these (which happens with probability $2/\ell$), the probability that $y^{(k)}_{\hat{\jmath}} = \hat{b}$ is $1/2$. □

Lemma 58. *For all f,k and $\mathcal{A}$,* $\mathrm{Prob}[\mathsf{Guess}^{\mathcal{A}}_{2,G}(f,k)] \geq \frac{1}{2\ell}$ *where $G = \mathsf{GBadQ}(f,k)$.*

PROOF. $\mathsf{Guess}^{\mathcal{A}}_{2,G}(f,k)$ is the event that $y^{(k)}_{\hat{\jmath}} \neq x_{\eta,\hat{\jmath}}$ in the game G played by the challenger $\mathcal{C}$ with $\mathcal{A}$. It is easy to see that the probability that $\mathcal{C}$ correctly guesses $\hat{\jmath}$ and $\hat{b}$ such that $x_{\eta,\hat{\jmath}} = \hat{c} = 1 - \hat{b}$ is at least $1/(2\ell)$, independently from the view of $\mathcal{A}$. □

Lemma 59. *Suppose $T = T_1$. If $\mathcal{S}$ does not abort in the computation of the answer to the k-th query, then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ is the same as in $\mathsf{GBadQ}(f,k-1)$. Moreover, if $\mathcal{S}$ completes its execution without aborting, then $\mathcal{A}$'s total view is the same as in $\mathsf{GBadQ}(f,k-1)$.*

Suppose instead that $T = T_2$. If $\mathcal{S}$ does not abort in the computation of the answer to the k-th query, then $\mathcal{A}$'s view up to the Challenge Query in the interaction with $\mathcal{S}$ is the same as in $\mathsf{GBadQ}(f,k)$. Moreover, if $\mathcal{S}$ completes its execution without aborting, then $\mathcal{A}$'s total view is the same as in $\mathsf{GBadQ}(f,k)$.

PROOF. For the proof of this lemma it is convenient to refer to the alternative and equivalent description of our HVE found in Section A. We notice that Pk has the same distribution as the public parameters seen by $\mathcal{A}$ in both games. The same holds for the answers to the first $(k-1)$ Key Queries and to the last $(q-k)$ Key Queries. Let us now focus on the answer to the k-th Key Query. We have two cases:

Case 1: $y^{(k)}_{f} = \star$**.** Then the view of $\mathcal{A}$ in the interaction with $\mathcal{S}$ is independent from T (see Case B.1) and, on the other hand, by definition, the two games coincide. Therefore the lemma holds in this case.

Case 2: $y^{(k)}_{f} \neq \star$**.** Suppose $T = T_1 = A_1^{\alpha\beta} \cdot D_4$ and that $\mathsf{Guess}^{\mathcal{A}}_{1,\mathcal{S}}(f,k)$ occurs. Therefore, $y^{(k)}_{\hat{\jmath}} = \hat{b}$ and $\mathcal{S}$'s answer to the k-th key query has the same distributions as in

$\mathsf{GBadQ}(f, k-1)$. Indeed, we have that

$$Y_{\hat{\jmath}} = g_1^{a'_{\hat{\jmath}}/r_{\hat{\jmath},\hat{b}}} \cdot g_2^{a''_{\hat{\jmath}}/v_{\hat{\jmath},\hat{b}}} \cdot D_4 \cdot W_{\hat{\jmath}}$$

with $a'_{\hat{\jmath}} = \alpha$ and $r_{\hat{\jmath},\hat{b}} = 1/\beta$ and

$$Y_h = g_1^{-(a'_{\hat{\jmath}}+s')/r_{h,y_h^{(k)}}} \cdot g_2^{-(a''_{\hat{\jmath}}+s'')/v_{h,y_h^{(k)}}} \cdot \left(B_4^{-1/r_{h,y_h^{(k)}}} \cdot W_h \right)$$

and thus the a'_is and a''_is are random and sum up to 0.

On the other hand if T is random in $\mathbb{G}_{p_1 p_4}$ and $\mathcal{S}$ does not abort, the $\mathbb{G}_{p_1}$ parts of the Y_i's are random and thus the answer to the k-th query of $\mathcal{A}$ is distributed as in $\mathsf{GBadQ}(f, k)$.

Finally, we observe that, if $\mathcal{S}$ does not abort then the challenge ciphertext is constructed as in $\mathsf{GBadQ}(f, k-1)$ and $\mathsf{GBadQ}(f, k)$. □

PROOF.

Lemma 60. *If Assumption 2 holds, then for $k = 1, \ldots, q$ and $f = 1, \ldots, \ell + 1$, and for all PPT adversaries $\mathcal{A}$, $\left|\text{Prob}[E^{\mathcal{A}}_{f,G}] - \text{Prob}[E^{\mathcal{A}}_{f,H}]\right|$ and $\left|\text{Prob}[\mathsf{Guess}^{\mathcal{A}}_{2,G}] - \text{Prob}[\mathsf{Guess}^{\mathcal{A}}_{2,H}]\right|$ are negligible functions of λ, for games $G = \mathsf{GBadQ}(f, k-1)$ and $H = \mathsf{GBadQ}(f, k)$ or $G = \mathsf{GBadQ2}(f, k-1)$ and $H = \mathsf{GBadQ2}(f, k)$.*

PROOF. Let us prove first the case when $G = \mathsf{GBadQ}(f, k-1)$ and $H = \mathsf{GBadQ}(f, k)$. For sake of contradiction, suppose that $\text{Prob}[E^{\mathcal{A}}_{f,G}] \geq \text{Prob}[E^{\mathcal{A}}_{f,H}] + \epsilon$ for some non-negligible ϵ. A similar reasoning holds $\mathsf{Guess}^{\mathcal{A}}_{2,G}$ and $\mathsf{Guess}^{\mathcal{A}}_{2,H}$. Then we can modify simulator $\mathcal{S}$ into algorithm $\mathcal{B}$ with a non-negligible advantage in breaking Assumption 2. Algorithm $\mathcal{B}$ simply execute $\mathcal{S}$'s code. By Lemma 57 event $\mathsf{Guess}_{1,\mathcal{S}}$ occurs with probability at least $1/\ell$ and in this case $\mathcal{B}$ can continue the execution of $\mathcal{S}$'s code and receive the challenge vectors from $\mathcal{A}$. At this point, $\mathcal{B}$ checks whether they differ in the f-th component. If they do, $\mathcal{B}$ outputs 1; else $\mathcal{B}$ outputs 0. It is easy to see that, by Lemma 59, the above algorithm has a non-negligible advantage in breaking Assumption 2.

We apply the the same reasoning to the case when $G = \mathsf{GBadQ2}(f, k-1)$ and $H = \mathsf{GBadQ2}(f, k)$, considering algorithm $\mathcal{B}$ that uses the code of simulator $\mathcal{S}_2$ rather than that of $\mathcal{S}$. □

The proof of the following corollary is straightforward from Lemma 60 and Observations 53-55.

Corollary 61. *For all $f = 1, \ldots, \ell + 1$ and $k = 0, \ldots, q$, and all PPT adversaries $\mathcal{A}$, we have that, for $H = \mathsf{GBadQ}(f, k)$ or $H = \mathsf{GBadQ2}(f, k)$,*
$\left|\text{Prob}[E^{\mathcal{A}}_{f,H}] - \text{Prob}[E^{\mathcal{A}}_{f}]\right|$ and $\left|\text{Prob}[\mathsf{Guess}^{\mathcal{A}}_{2,H}] - \text{Prob}[\mathsf{Guess}^{\mathcal{A}}_{2}]\right|$ are negligible.

We are now ready to prove the following crucial technical lemma.

□

Lemma 62. *Suppose there exists an adversary $\mathcal{A}$ and integers $1 \le f \le \ell+1$ and $1 \le k \le q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \ge \epsilon$, where $G = \mathsf{GBadQ}(f, k-1)$, $H = \mathsf{GBadQ}(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}_2^{\mathcal{B}} \ge \mathrm{Prob}[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.*

The proof of the advantage in breaking Assumption 2 is found in Appendix **??**.

The following Lemma can be proved by referring to simulator $\mathcal{S}_2$. We omit further details since the proof is essentially the same as the one of Lemma 62.

Lemma 63. *Suppose there exists an adversary $\mathcal{A}$ and integers $1 \le f \le \ell+1$ and $1 \le k \le q$ such that $\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \ge \epsilon$, where $G = \mathsf{GBadQ2}(f, k-1)$, $H = \mathsf{GBadQ2}(f,k)$ and $\epsilon > 0$. Then, there exists a PPT algorithm $\mathcal{B}$ with $\mathsf{Adv}_2^{\mathcal{B}} \ge \mathrm{Prob}[E_f] \cdot \epsilon/(2 \cdot \ell^2) - \nu(\lambda)$, for a negligible function ν.*

The advantage of $\mathcal{A}$ in GPK

In this section we prove that, under Assumption 2, every PPT adversary $\mathcal{A}$ has a negligible advantage in $\mathsf{GPK} = \mathsf{GBadCh}(1)$ by proving that it is computationally indistinguishable from $\mathsf{GBadCh}(\ell+1)$ that, by Observation 56, gives no advantage to any adversary.

Proof. Let $E^{\mathcal{A}}_{f,f'}$ denote the event that during the execution of $\mathsf{GBadCh}(f')$ adversary $\mathcal{A}$ outputs two challenge vectors that differ in the f-th component. For an event E, we define the *advantage* $\mathsf{Adv}^{\mathcal{A}}[G|E]$ *of $\mathcal{A}$ in G conditioned on event E* as $\mathsf{Adv}^{\mathcal{A}}[G|E] = \mathrm{Prob}[\mathcal{A} \text{ wins in game } G|E] - \frac{1}{2}$.

Observation 64. *For all PPT adversaries $\mathcal{A}$ and all $1 \le f \le \ell$, we have that*

$$\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}] = \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|\neg E_{f,f+1}].$$

Proof. By definition of GBadCh, if the two challenge vectors coincide in the f-th component, then the views of $\mathcal{A}$ in $\mathsf{GBadCh}(f)$ and $\mathsf{GBadCh}(f+1)$ are the same. $\square$

Observation 65. *For all PPT adversaries $\mathcal{A}$ and all $1 \le f \le \ell$, 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 $\mathsf{GBadCh}(f)$ up to the Challenge Query is independent from f. $\square$

Therefore we can set $\mathrm{Prob}[E^{\mathcal{A}}_{f,f}] = \mathrm{Prob}[E^{\mathcal{A}}_{f,1}] = \mathrm{Prob}[E^{\mathcal{A}}_{f}]$.

$\square$

Lemma 66. *If Assumption 2 holds, then, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}]$ is negligible. Specifically, if there is an adversary $\mathcal{A}$ with $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] = \epsilon$ then there exists an adversary $\mathcal{B}$ against Assumption 2 such that $\mathsf{Adv}_2^{\mathcal{B}} \ge \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$, for some negligible function ν.*

Proof. Let $\mathcal{A}$ be a PPT adversary such that $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GPK}] \geq \epsilon$. Since $\mathsf{GPK} = \mathsf{GBadCh}(1)$ and $\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(\ell+1)] = 0$ there must exist $f \in [\ell]$ such that

$$\left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]\right| \geq \epsilon' = \epsilon/\ell. \tag{12.2}$$

Now recall that $\mathsf{GBadCh}(f) = \mathsf{GBadQ}(f, 0)$ and $\mathsf{GBadCh}(f+1) = \mathsf{GBadQ2}(f, 0)$. Thus, there exists k, $1 \leq k \leq q$ such that

$$\left|\mathsf{Adv}^{\mathcal{A}}[G] - \mathsf{Adv}^{\mathcal{A}}[H]\right| \geq \epsilon'/(2q)$$

where $G = \mathsf{GBadQ}(f, k)$ and $H = \mathsf{GBadQ}(f, k-1)$ or where $G = \mathsf{GBadQ2}(f, k)$ and $H = \mathsf{GBadQ2}(f, k-1)$. Then by Lemma 62, in the former case, and by Lemma 63 in the latter, we can construct an adversary $\mathcal{B}$ against Assumption 2, such that

$$\mathsf{Adv}_2^{\mathcal{B}} \geq \frac{\epsilon}{4q\ell^3} \cdot \mathrm{Prob}[E_f] - \nu(\lambda)$$

Now it remains to estimate $\mathrm{Prob}[E_f]$. Notice that we can write

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)] = {} & \mathrm{Prob}[E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] + \\ & \mathrm{Prob}[\neg E_{f,f}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|\neg E_{f,f}].\end{aligned}$$

and

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)] = {} & \mathrm{Prob}[E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}] + \\ & \mathrm{Prob}[\neg E_{f,f+1}] \cdot \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)]|\neg E_{f,f+1}].\end{aligned}$$

and, by combining Equation 12.2 and Observations 64 and 65, we obtain

$$\mathrm{Prob}[E_f] \cdot \left|\mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f)|E_{f,f}] - \mathsf{Adv}^{\mathcal{A}}[\mathsf{GBadCh}(f+1)|E_{f,f+1}]\right| \geq \epsilon'.$$

Since no advantage is greater than 1/2, we can conclude that $\mathrm{Prob}[E_f] \geq 2 \cdot \epsilon'$ and thus $\mathcal{B}$ as advantage

$$\mathsf{Adv}_2^{\mathcal{B}} \geq \frac{\epsilon^2}{2q\ell^4} - \nu(\lambda)$$

□

12.4.3 Wrapping up

In this section we state our main result.

Theorem 67. *If Assumption* 1 *and* 2 *hold, then the HVE scheme described in Section 12.3 is secure (in the sense of Definition 2).*

Proof. Use Lemma 52 and Lemma 66. □

Chapter 13

Inner-product Encryption Construction

In this chapter, we present the Inner Product Encryption scheme of Katz, Sahai and Waters [28]. It was the first IPE scheme introduced in the literature, and its security was proved in the selective-id model. It is based on bilinear groups of order product of three primes. We choose to present the predicate-only implementation.

Intuition. In the construction, each ciphertext has associated with it a (secret) vector $\vec{x}$, and each secret key corresponds to a vector $\vec{v}$. The decryption procedure must check whether $< x, v >= 0$, and reveal nothing about $\vec{x}$ but whether this is true. To do this, we will make use of a bilinear group $\mathbb{G}$ whose order N is the product of three primes p_1, p_2, p_3. Let $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$, and $\mathbb{G}_{p_3}$ denote the subgroups of $\mathbb{G}$ having order p_1, p_2, and p_3, respectively. It is assumed that a random element in any of these subgroups is indistinguishable from a random element of $\mathbb{G}$. Thus, we can use random elements from one subgroup to mask elements from another subgroup. At a high level, we will use these subgroups as follows. $\mathbb{G}_{p_2}$ will be used to encode the vectors $\vec{x}$ and $\vec{v}$ in the ciphertext and secret keys, respectively. Computation of the inner product v, x will be done in $\mathbb{G}_{p_2}$ (in the exponent), using the bilinear map. $\mathbb{G}_{p_1}$ will be used to encode an equation (again in the exponent) that evaluates to zero when decryption is done properly. This subgroup is used to prevent an adversary from improperly manipulating the computation (by, e.g., changing the ordering of components of the ciphertext or secret key, raising these components to some power, etc.). On an intuitive level, if the adversary tries to manipulate the computation in any way, then the computation occurring in the $\mathbb{G}_{p_1}$ subgroup will no longer yield the identity (i.e., will no longer yield 0 in the exponent), but will instead have the effect of masking the correct answer with a random element of $\mathbb{G}_{p_1}$ (which will invalidate the entire computation). Elements in $\mathbb{G}_{p_3}$ are used for general masking of terms in other subgroups; i.e., random elements of $\mathbb{G}_{p_3}$ will be multiplied with various components of the ciphertext (and secret key) in order to hide information that might be present in the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ subgroups. We now present the formal description of the scheme.

$\mathsf{Setup}(1^\lambda, 1^\ell)$: The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1 p_2 p_3, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses random $g_1 \in \mathbb{G}_{p_1}$, $g_2 \in \mathbb{G}_{p_2}$, $g_3 \in \mathbb{G}_{p_3}$, random $R_0 \in \mathbb{G}_{p_3}$ and, for $i \in [\ell]$ random $R_{1,i}, R_{2,i} \in \mathbb{G}_{p_3}$. and sets $Q = g_2 \cdot R_0$, and, for $i \in [\ell], b \in \{1,2\}$

$$H_{b,i} = h_{b,i} \cdot R_{b,i}.$$

The public parameters are

$$\mathsf{Pk} = [N, g_1, g_3, (H_{b,i})_{i\in[\ell], b\in\{1,2\}}]$$

and the master secret key is

$$\mathsf{Msk} = [p_1, p_2, p_3, g_2, (h_{b,i})_{i\in[\ell], b\in\{1,2\}}].$$

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{v})$: Let $\vec{v} = (v_1, \ldots, v_\ell)$ with $v_i \in \mathbb{Z}_N$. The key generation algorithm chooses random $r_{b,i} \in \mathbb{Z}_{p_1}$ for $i \in [\ell], b \in \{1,2\}$, random $R_5 \in \mathbb{G}_{p_3}$, random $f_1, f_2 \in \mathbb{Z}_q$ and random $Q_6 \in \mathbb{G}_{p_2}$. For $i \in [\ell], b \in \{1,2\}$, it sets

$$K_{b,i} = g_1^{r_{b,i}} \cdot g_2^{f_b v_i}.$$

Furthermore, it sets

$$K = R_5 \cdot Q_6 \cdot \prod_{i=1}^{\ell} h_{1,i}^{-r_{1,i}} \cdot h_{2,i}^{-r_{2,i}}$$

and returns the tuple

$$\mathsf{Sk}_{\vec{v}} = (K, (K_{b,i})_{i\in[\ell], b\in\{1,2\}}).$$

$\mathsf{Encrypt}(\mathsf{Pk}, \vec{x})$: Let $\vec{x} = (x_1, \ldots, x_\ell)$ with $x_i \in \mathbb{Z}_N$. The encryption algorithm chooses random $s, \alpha, \beta, \in \mathbb{Z}_N$ and $R_{3,i}, R_{4,i} \in \mathbb{G}_{p_3}$ for $i = 1, \ldots, \ell$. It can make this because it has a generator g_3 of $\mathbb{G}_{p_3}$. For $i \in [\ell]$ and $b \in \{1,2\}$, the algorithm sets

$$C_{b,i} = H_{1,i}^s \cdot Q^{\alpha \cdot x_i} \cdot R_{3,i}.$$

Furthermore, it sets $C_0 = g_1^s$, and returns the tuple

$$\mathsf{Ct} = (C_0, (X_{b,i})_{i\in[\ell], b\in\{1,2\}}).$$

$\mathsf{Test}(\mathsf{Ct}, \mathsf{Sk}_{\vec{v}})$: Let $\mathsf{Ct} = (C_0, (X_{b,i})_{i\in[\ell], b\in\{1,2\}})$ and $\mathsf{Sk}_{\vec{v}} = (K, (K_{b,i})_{i\in[\ell], b\in\{1,2\}})$ computed above. The test algorithm computes

$$T = \mathbf{e}(C_0, K_0) \prod_{i=1}^{\ell} \mathbf{e}(C_{1,i}, K_{1,i}) \cdot \mathbf{e}(C_{2,i}, K_{2,i}).$$

It returns TRUE if $T = 1$, FALSE otherwise.

Correctness. See [28]

Security. For the security proof and the assumptions which it is based on, see [28].

Other IPE schemes. Following the work of Katz, Sahai and Waters, other implementations of IPE have been presented in the literature. Lewko, Okamoto, Sahai, Takashima and Waters [30] present the first implementation of a fully secure IPE scheme. The security of the latter scheme relies on a non-standard assumption of non-constant size. The IPE scheme of [30] is based on Dual Pairing Vector Spaces (DPVS) introduced by Okamoto and Takashima [33]. The latter work also introduces the concept of Hierarchical Inner Product Encryption (HIPE). In HIPE, from a key for vector $\vec{v}$, you may delegate a key for vector $(\vec{v}, 0)$ which contains a new dimension. HIPE does not seem to have as special case HHVE. Following these works, Okamoto and Takashima [34] presented an implementation of IPE and HIPE which is proved fully secure from the Decision Linear assumption.

Chapter 14

Open Problems

Predicate encryption is a young area of research and a lot of study will have to be done to better understand this primitive. We point out some of possible directions for future research as well as challenges and open problems.

- One relevant direction of research is to construct PE schemes for new classes of predicates of interest in applications. The ultimate goal would be to construct a PE scheme for any poly-size circuit. It would allow to exucte any boolean program with input encrypted data without the program learning anything else than the result of the computation. For example you could want enable your email provider to execute an antivirus check on your encrypted email. To such aim, you could give your provider a key for the antivirus program which would enable it to check the presence of viruses without learning anything else than the presence or absence of viruses. Since that this problem is very ambitious, we could expect both positive and negative solutions to it. PE for any poly-size circuit remains an open problem even for its non attribute-hiding form, i.e. ABE. In fact, the current state of the art for ABE offers ABE for non-monotone access structures.

- Another direction of research of both pratical and theoretical interest is to focus on the efficiency of the PE schemes. For example an interesting problem could be to build an inner-product encryption scheme that achieves constant size ciphertexts or keys, or a non-trivial PE scheme with a decryption algorithm that uses a constant number of pairing operations.

- An interesting line of research is to build interesting PE schemes from primitives different than bilinear maps. For example, can we obtain an PE scheme for an interesting class of predicates from lattices or quadratic residuosity assumptions? IBE implementations based on Quadratic Residues were presented by Cocks [20] and by Boneh, Gentry and Hamburg [9]. Beyond these schemes, (H)IBE systems based on lattices were introduced in literature, see for example [19]. While solutions not based on the bilinear maps have been given for (H)IBE, it is yet an open problem to build HVE or more general primitives from alternative tools.

- Another problem is to study the relations between existing primitives. For example, can we build an HVE scheme from IBE in a black-box manner? Reductions or black-box separations have to be found. In this direction, Boneh, Papakonstantinou, Rackoff, Vahlis and Waters [12] showed a black-box separation of IBE from trapdoor permutions. Following this work, Katz and Yerukhimovich [29] extended these ideas showing both negative and positive results for PE systems.

- Also relevant to previous problems is the study of the limitations of the bilinear maps. Which class of PE schemes can be built from bilinear maps? It could be the case that the bilinearity of the bilinear maps fundamentally limits the class of possible efficient schemes constructible from them.

- The construction of multilinear maps would be a breakthrough and could help us to obtain new and more general PE schemes as well as many other cryptographic primitives.

- Existing proofs of security have the disavantage that they result in a non tight reduction. In fact, by using Dual System Encryption, you loose a factor q in the reduction from the adversary breaking the scheme to the computational assumption, where q is the number of queries issued by the adversary. To prove the security (in the standard model) with a tight reduction is an interesting open problem even for IBE. IBE with tight reductions in the random oracle model was proposed by Goh, Jarecki, Katz and Wang [24] (precisely they focus on digital signatures but give an intuition of how to adapt their ideas to IBE). Solutions in the standard model could request the developement of new techniques.

Bibliography

[1] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006*, volume 4052 of *LNCS*, pages 300–311, Venezia, Italy, July 10–14, 2006. Springer-Verlag, Berlin, Germany.

[2] Vincenzo Iovino Angelo De Caro and Giuseppe Persiano. Efficient fully secure (hierarchical) predicate encryption for conjunctions, disjunctions and k-cnf/dnf formulae. Technical Report 2010-492, Cryptology ePrint Archives, 2010. `http://eprint.iacr.org/2010/492/`.

[3] Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Private-key hidden vector encryption with key confidentiality. In *Cryptology and Network Security (CANS)*, pages 259–277, 2009.

[4] Carlo Blundo, Vincenzo Iovino, and Giuseppe Persiano. Predicate encryption with partial public keys. In *Cryptology and Network Security (CANS)*, pages 298–313, 2010.

[5] Dan Boneh. Bilinear groups of composite order. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing-Based Cryptography - Pairing 2007, First International Conference. Prooceedings*, volume 4575 of *LNCS*, pages 39–56, Tokyo, Japan, July 2–4, 2007. Springer-Verlag, Berlin, Germany.

[6] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.

[9] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[10] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC 2005*, volume 3378 of *LNCS*, pages 325–341, Cambridge, MA, USA, February 10–12, 2005. Springer-Verlag, Berlin, Germany.

[11] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103, San Francisco, CA, USA, February 14–18, 2005. Springer-Verlag, Berlin, Germany.

[12] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *FOCS*, pages 283–292, 2008.

[13] Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and challenges. Technical Report 2010-543, Cryptology ePrint Archives, 2010. `http://eprint.iacr.org/2010/543/`.

[14] Dan Boneh and Brent Waters. Conjunctive, subset and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554, Amsterdam, The Netherlands, February 21–24, 2007. Springer-Verlag, Berlin, Germany.

[15] Xavier Boyen and Brent Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307, Santa Barbara, CA, USA, August 20–24, 2006. Springer-Verlag, Berlin, Germany.

[16] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany.

[17] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *IEEE Symposium on Computers and Communications (ISCC)*, 2011. To appear.

[18] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *Pairing*, pages 347–366, 2010.

[19] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, pages 523–552, French Riviera, France, May 10 –June 3, 2010. Springer-Verlag, Berlin, Germany.

[20] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer-Verlag, Berlin, Germany.

[21] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[22] Craig Gentry. Pratical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464, St. Petersburg, Russia, May 28 –June 1, 2006. Springer-Verlag, Berlin, Germany.

[23] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 437–456, San Francisco, CA, USA, 2009. Springer-Verlag, Berlin, Germany.

[24] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *Journal of Cryptology*, 20(4):493–514, 2007.

[25] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control for Encrypted Data. In *ACM CCS 06*, pages 89–98, Alexandria, VA, USA, October 30 - November 3, 2006. ACM Press.

[26] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer-Verlag, Berlin, Germany.

[27] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In Steven D. Galbraith and Kenneth G. Paterson, editors, *Pairing-Based Cryptography - Pairing 2008, Second International Conference. Prooceedings*, volume 5209 of *LNCS*, pages 75–88, Egham, UK, September 1–3, 2008. Springer-Verlag, Berlin, Germany.

[28] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate Encryption Supporting Disjunction, Polynomial Equations, and Inner Products. In Nigel Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer-Verlag, Berlin, Germany.

[29] Jonathan Katz and Arkady Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In *ASIACRYPT*, pages 197–213, 2009.

[30] Allison 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, *EUROCRYPT 2010*, pages 62–91, French Riviera, France, May 10 –June 3, 2010. Springer-Verlag, Berlin, Germany.

[31] 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*, volume 5978 of *LNCS*, pages 455–479, Zurich, Switzerland, February 9–11, 2010. Springer-Verlag, Berlin, Germany.

[32] Ben Lynn. *On the Implementation of Pairing-Based Cryptography, Ph.D. Thesis.* 2007. `http://crypto.stanford.edu/pbc/thesis.html`.

[33] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231, Tokyo, Japan, December 6–10, 2009. Springer-Verlag, Berlin, Germany.

[34] Tatsuaki Okamoto and Katsuyuki Takashima. Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208, Santa Barbara, CA, USA, August 15–19, 2010. Springer-Verlag, Berlin, Germany.

[35] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[36] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany.

[37] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 457–473, San Francisco, CA, USA, 2009. Springer-Verlag, Berlin, Germany.

[38] 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*, volume 5126 of *LNCS*, pages 560–578, Reykjavik, Iceland, July 7–11, 2008. Springer-Verlag, Berlin, Germany.

[39] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany.

[40] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer-Verlag, Berlin, Germany.

Appendix A

An alternative description of our main HVE scheme

In this section we give an alternative albeit equivalent description of the HVE algorithms of Sections 12.3 and 8.2 that will make our proof of security simpler.

We start from the simple observation that the exponent arithmentic is performed modulo the order of the base. For sake of concreteness, let us look at the KeyGen algorithm that sets Y_i equal to

$$Y_i = (g_1 \cdot g_2)^{a_i/t_{i,y_i}} \cdot W_i,$$

for $a_i, t_{i,y_i} \in \mathbb{Z}_N$. This is equivalent to computing Y_i as

$$Y_i = g_1^{a_i'/r_{i,y_i}} \cdot g_2^{a_i''/v_{i,y_i}} \cdot W_i$$

for $a_i', a_i'', r_{i,y_i}, v_{i,y_i} \in \mathbb{Z}_N$ satisfying the following system of modular equations

$$\begin{aligned} a_i' &\equiv a_i \pmod{p_1} & \qquad r_{i,y_i} &\equiv t_{i,y_i} \pmod{p_1} \\ a_i'' &\equiv a_i \pmod{p_2} & \qquad v_{i,y_i} &\equiv t_{i,y_i} \pmod{p_2} \end{aligned}$$

Conversely, computing $Y_i = g_1^{a_i'/r_{i,b}} \cdot g_2^{a_i''/v_{i,b}}$ for $a_i', a_i'', r_{i,b}, v_{i,b} \in \mathbb{Z}_N$ is equivalent to computing $g_1^{a_i/t_{i,b}} \cdot g_2^{a_i/t_{i,b}}$ for $a_i, t_{i,b} \in \mathbb{Z}_N$ satisfying the above system of modular equations (in the unknowns a_i and $t_{i,b}$). By the Chinese remainder theorem the above systems have solutions in $\mathbb{Z}_N$ provided that N is a multiple of $p_1 \cdot p_2$. Moreover, for all values of $r_{i,b}$ and $v_{i,b}$, and of a_i' and a_i'' the systems above have the same number of solutions. Therefore, the distributions of $Y_i = g_{12}^{a_i/t_{i,b}}$ for random $a_i, t_{i,b} \in \mathbb{Z}_N$ and the distribution of $Y_i = g_1^{a_i'/r_{i,b}} \cdot g_2^{a_i''/v_{i,b}}$ for random $a_i', a_i'', r_{i,b}, v_{i,b} \in \mathbb{Z}_N$ coincide.

In view of the above observation, we can describe the Setup and KeyGen algorithms in the following way.

$\mathsf{Setup}(1^\lambda, 1^\ell)$**:** The setup algorithm chooses a description of a bilinear group $\mathcal{I} = (N = p_1p_2p_3p_4, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ with known factorization by running a generator algorithm $\mathcal{G}$ on input 1^λ. The setup algorithm chooses 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 $r_{i,b}, v_{i,b} \in Z_N$ and random $R_{i,b} \in \mathbb{G}_{p_3}$ and sets $T_{i,b} = g_1^{r_{i,b}} \cdot R_{i,b}$.

The public parameters are $\mathsf{Pk} = [N, g_3, (T_{i,b})_{i\in[\ell], b\in\{0,1\}}]$ and the master secret key is $\mathsf{Msk} = [g_{12}, g_4, (r_{i,b}, v_{i,b})_{i\in[\ell], b\in\{0,1\}}]$, where $g_{12} = g_1 \cdot g_2$.

$\mathsf{KeyGen}(\mathsf{Msk}, \vec{y})$**:** 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 $i \in S_{\vec{y}}$ and random $a''_i \in \mathbb{Z}_N$ for $i \in S_{\vec{y}}$ under the constraint that $\sum_{i\in S_{\vec{y}}} a'_i = \sum_{i\in S_{\vec{y}}} a''_i = 0$. For $i \in S_{\vec{y}}$, the algorithm chooses random $W_i \in \mathbb{G}_{p_4}$ and sets

$$Y_i = g_1^{a'_i/r_{i,y_i}} \cdot g_2^{a''_i/v_{i,y_i}} \cdot W_i.$$

The algorithm returns the tuple $(Y_i)_{i\in S_{\vec{y}}}$.

Appendix B

Generic security of our Complexity Assumptions

The complexity assumption BDDH and Decision Linear were already known in literature so we omit their discussion. Instead, we motivate the use of other novel complexity assumptions used in some of our constructions. We now prove that, if factoring is hard, the complexity assumptions of Section 12.2 and 11.1 hold in the generic group model. We adopt the framework of [28] to reason about assumptions in bilinear groups $\mathbb{G}, \mathbb{G}_T$ of composite order $N = p_1 p_2 p_3 p_4$. We fix generators $g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}$ of the subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}, \mathbb{G}_{p_4}$ and thus each element of $x \in \mathbb{G}$ can be expressed as $x = g_{p_1}^{a_1} g_{p_2}^{a_2} g_{p_3}^{a_3} g_{p_4}^{a_4}$, for $a_i \in \mathbb{Z}_{p_i}$. For sake of ease of notation, we denote element $x \in \mathbb{G}$ by the tuple (a_1, a_2, a_3, a_4). We do the same with elements in $\mathbb{G}_T$ (with the respect to generator $\mathbf{e}(g_{p_i}, g_{p_i})$) and will denote elements in that group as bracketed tuples $[a_1, a_2, a_3, a_4]$. We use capital letters to denote random variables and reuse random variables to denote relationships between elements. For example, $X = (A_1, B_1, C_1, D_1)$ is a random element of $\mathbb{G}$, and $Y = (A_2, B_1, C_2, D_2)$ is another random element that shares the same $\mathbb{G}_{p_2}$ part.

We say that a random variable X is *dependent* from the random variables $\{A_i\}$ if there exists $\lambda_i \in \mathbb{Z}_N$ such that $X = \sum_i \lambda_i A_i$ as formal random variables. Otherwise, we say that X is *independent* of $\{A_i\}$. We state the following theorems from [28].

Theorem 68 (Theorem A.1 of [28]). *Let $N = \prod_{i=1}^{m} p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}$ be random variables over $\mathbb{G}$ and $\{Y_i\}, T_1$ and T_2 be random variables over $\mathbb{G}_T$. Denote by t the maximum degree of a random variable and consider the following experiment in the generic group model:*

Algorithm $\mathcal{A}$ is given $N, \{X_i\}, \{Y_i\}$ and T_b for random $b \in \{0,1\}$ and outputs $b' \in \{0,1\}$. $\mathcal{A}$'s advantage is the absolute value of the difference between the probability that $b = b'$ and $1/2$.

Suppose that T_1 and T_2 are independent of $\{Y_i\} \cup \{\mathbf{e}(X_i, X_j)\}$. Then if $\mathcal{A}$ performs at most q group operations and has advantage δ, then there exists an algorithm that outputs a *nontrivial factor of N in time polynomial in λ and the running time of $\mathcal{A}$ with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.*

Theorem 69 (Theorem A.2 of [28]). *Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}, T_1, T_2$ be random variables over $\mathbb{G}$ and let $\{Y_i\}$ be random variables over G_T, where all random variables have degree at most t.*

Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{X_i\}, T_1$ and T_2 be random variables over $\mathbb{G}$ and let $\{Y_i\}$ be random variables over $\mathbb{G}_T$. Denote by t the maximum degree of a random variable and consider the same experiment as the previous theorem in the generic group model.

Let $S := \{i \mid \mathbf{e}(T_1, X_i) \neq \mathbf{e}(T_2, X_i)\}$ (where inequality refers to inequality as formal polynomials). Suppose each of T_1 and T_2 is independent of $\{X_i\}$ and furthermore that for all $k \in S$ it holds that $\mathbf{e}(T_1, X_k)$ is independent of $\{B_i\} \cup \{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$ and $\mathbf{e}(T_2, X_k)$ is independent of $\{B_i\} \cup \{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_2, X_i)\}_{i \neq k}$. Then if there exists an algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ, then there exists an algorithm that outputs a nontrivial factor of N in time polynomial in λ and the running time of $\mathcal{A}$ with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.

We apply Theorem 69 to prove the security of our assumptions in the generic group model.

Assumption 1. We can express this assumption as:

$$X_1 = (0,0,1,0),\ X_2 = (A_1, 0, A_3, 0),\ X_3 = (B_1, 0, B_3, 0),\ X_4 = (0,0,0,1)$$

and

$$T_1 = (Z_1, 0, Z_3, 0), T_2 = (0, Z_2, Z_3, 0).$$

It is easy to see that T_1 and T_2 are both independent of $\{X_i\}$ because, for example, Z_3 does not appear in the X_i's. Next, we note that for this assumption we have $S = \{2, 3\}$, and thus, considering T_1 first, we obtain the following tuples:

$$C_{1,2} = \mathbf{e}(T_1, X_2) = [Z_1 A_1, 0, Z_3 A_3, 0], \quad C_{1,3} = \mathbf{e}(T_1, X_3) = [Z_1 B_1, 0, Z_3 B_3, 0].$$

It is easy to see that $C_{1,k}$ with $k \in \{2, 3\}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$. An analogous arguments apply for the case of T_2. Thus the independence requirements of Theorem 69 are satisfied and Assumption 1 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 2. We can express this assumption as:

$$\begin{aligned} X_1 &= (1,0,0,0), & X_2 &= (0,1,0,0), & X_3 &= (0,0,1,0),\\ X_4 &= (0,0,0,1), & X_5 &= (A, 0, 0, B_4), & X_6 &= (B, 0, 0, C_4) \end{aligned}$$

and

$$T_1 = [AB, 0, 0, D_4], \quad T_2 = [Z_1, 0, 0, Z_4].$$

We note that D_4 and Z_1 do not appear in $\{X_i\}$ and thus T_1 and T_2 are both independent from them. Next, we note that for this assumption we have $S = \{1, 4, 5, 6\}$, and thus, considering T_1 first, we obtain the following tuples:

$$\begin{array}{ll} C_{1,1} = \mathbf{e}(T_1, X_1) = [AB, 0, 0, 0], & C_{1,4} = \mathbf{e}(T_1, X_4) = [0, 0, 0, D_4] \\ C_{1,5} = \mathbf{e}(T_1, X_5) = [A^2B, 0, 0, D_4B_4], & C_{1,6} = \mathbf{e}(T_1, X_6) = [AB^2, 0, 0, D_4C_4] \end{array}$$

It is easy to see that $C_{1,k}$ with $k \in \{4, 5, 6\}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$. For $C_{1,1}$, we observe that the only way to obtain an element whose first component contains AB is by computing $\mathbf{e}(X_5, X_6) = [AB, 0, 0, B_4C_4]$ but then there is no way to generate an element whose fourth component is B_4C_4 and hence no way to cancel that term.

Analogous arguments apply for the case of T_2.

Thus the independence requirement of Theorem 69 is satisfied and Assumption 2 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 3. We can express this assumption as:

$$\begin{array}{llll} X_1 = (1, 0, 0, 0), & X_2 = (0, 1, 0, 0), & X_3 = (0, 0, 1, 0), & X_4 = (0, 0, 0, 1), \\ X_5 = (A, 0, 0, B_4), & X_6 = (B, 0, 0, C_4), & X_7 = (C, 0, 0, D_4), & X_8 = (AC, 0, 0, E_4), \\ X_9 = (ABC, 0, 0, F_4) & & & \end{array}$$

and

$$T_1 = [AB, 0, 0, G_4], \quad T_2 = [Z_1, 0, 0, Z_4].$$

We note that G_4 and Z_1 do not appear in $\{X_i\}$ and thus T_1 and T_2 are both independent from them. Next, we note that for this assumption we have $S = \{1, 4, 5, 6, 7, 8, 9\}$, and thus, considering T_1 first, we obtain the following tuples:

$$\begin{array}{ll} C_{1,1} = \mathbf{e}(T_1, X_1) = [AB, 0, 0, 0], & C_{1,4} = \mathbf{e}(T_1, X_4) = [0, 0, 0, G_4] \\ C_{1,5} = \mathbf{e}(T_1, X_5) = [A^2B, 0, 0, G_4B_4], & C_{1,6} = \mathbf{e}(T_1, X_6) = [AB^2, 0, 0, G_4C_4] \\ C_{1,7} = \mathbf{e}(T_1, X_7) = [ABC, 0, 0, G_4D_4] & C_{1,8} = \mathbf{e}(T_1, X_8) = [A^2BC, 0, 0, G_4E_4] \\ C_{1,9} = \mathbf{e}(T_1, X_9) = [A^2B^2C, 0, 0, G_4F_4]. & \end{array}$$

We start by observing that, for $k = 9, 10, 11$, $C_{1,k}$ is independent from $\{\mathbf{e}(A_i, A_j)\} \cup \{\mathbf{e}(T_1, A_i)\}_{i \neq k}$, since it is the only to contain Z_2X_2 for $k = 9$, Z_2Y_2 for $k = 10$, and Z_2D_2 for $k = 11$. Similarly, $C_{1,k}$ for $k = 6, 8$ is independent since it contains Z_4W_4, for $k = 6$, and Z_4X_4, for $k = 8$.

It is easy to see that $C_{1,k}$ with $k \in \{4, 5, 9\}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$.

For $C_{1,1}$, we observe that the only way to obtain an element whose first component contains AB is by computing $\mathbf{e}(A_5, A_6) = [AB, 0, 0, B_4C_4]$ but then there is no way to generate an element whose fourth component is B_4C_4 and hence no way to cancel that term. Similarly for $C_{1,8}$, to obtain an element whose first component contains A^2BC the only way is by computing $\mathbf{e}(A_5, A_8) = [A^2BC, 0, 0, B_4F_4]$ but there is no way to cancel the fourth component B_4F_4.

For $C_{1,7}$, we notice that the only way to obtain an element whose first component contains ABC is by computing $\mathbf{e}(A_1, A_9) = [ABC, 0, 0, 0]$ but then there is no way to generate an element whose fourth component is G_4D_4 and hence no way to cancel that term from $C_{1,7}$.

Analogous arguments apply for the case of T_2.

Thus the independence requirement of Theorem 69 is satisfied and Assumption 3 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 4. We can express this assumption as:

$$X_1 = (0,1,0,0),\ X_2 = (0,0,1,0),\ X_3 = (0,0,0,1),\ X_4 = (A_1,0,0,A_4)$$

and

$$T_1 = (Z_1,0,0,Z_4), T_2 = (0,0,0,Z_4).$$

It is easy to see that T_1 and T_2 are both independent of $\{X_i\}$ because, for example, Z_4 does not appear in the X_i's. Next, we note that for this assumption we have $S = \{4\}$, and thus, considering T_1 first, we obtain the following tuples:

$$C_{1,4} = \mathbf{e}(T_1, X_4) = [Z_1A_1, 0, 0, Z_4A_4].$$

It is easy to see that $C_{1,4}$ is independent of $\{\mathbf{e}(X_i, X_j)\} \cup \{\mathbf{e}(T_1, X_i)\}_{i \neq k}$. An analogous arguments apply for the case of T_2. Thus the independence requirements of Theorem 69 are satisfied and Assumption 4 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption I_1. We can express this assumption as:

$$A_1 = (1,0,0,0),\ A_2 = (0,0,1,0),\ A_3 = (0,0,0,1)$$

$$A_4 = (X_1, X_2, 0, 0),\ A_5 = (0, Y_2, Y_3, 0),$$

and

$$T_1 = (Z_1, Z_2, Z_3, 0), \quad T_2 = (Z_1, 0, Z_3, 0).$$

It is easy to see that T_1 and T_2 are both independent of $\{A_i\}$ because, for example, Z_1 does not appear in the A_i's. Next, we note that for this assumption we have $S = \{4, 5\}$, and thus, considering T_1 first, we obtain the following tuples:

$$C_{1,4} = \mathbf{e}(T_1, A_4) = [Z_1X_1, Z_2X_2, 0, 0], \quad C_{1,5} = \mathbf{e}(T_1, A_5) = [0, Z_2Y_2, Z_3Y_3, 0].$$

It is easy to see that $C_{1,k}$ with $k \in \{4,5\}$ is independent of $\{\mathbf{e}(A_i, A_j)\} \cup \{\mathbf{e}(T_1, A_i)\}_{i \neq k}$. An analogous arguments apply for the case of T_2. Thus the independence requirements of Theorem 69 are satisfied and Assumption I_1 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption I_2. We can express this assumption as:

$$\begin{array}{lll} A_1 = (1,0,0,0), & A_2 = (0,1,0,0), & A_3 = (0,0,1,0), \\ A_4 = (0,0,0,1), & A_5 = (A, X_2, 0, 0), & A_6 = (S, Y_2, 0, 0) \\ A_7 = (0, X_2R, 0, 0), & A_8 = (0, R, 0, 0), & \end{array}$$

and

$$T_1 = [AS, 0, 0, 0], \quad T_2 = [Z_1, Z_2, Z_3, Z_4].$$

We note that Z_1 does not appear in $\{A_i\}$ and thus T_2 is independent from them. On the other hand, for T_1, the only way to obtain an element of $\mathbb{G}_T$ whose first component is AS is by computing $\mathbf{e}(A_5, A_6) = [AS, X_2Y_2, 0, 0]$ but there is no way to generate an element whose second component is X_2Y_2 and hence no way to cancel that term. Thus the independence requirement of Theorem 68 is satisfied and Assumption I_2 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption I_3. We can express this assumption as:

$$\begin{array}{llll} A_1 = (1,0,0,0), & A_2 = (0,1,0,0), & A_3 = (0,0,1,0), & A_4 = (0,0,0,1) \\ A_5 = (U,0,0,0), & A_6 = (US, W_2, 0, W_4), & A_7 = (UR, 0, 0, 0), & A_8 = (X_1, 0, 0, X_4) \\ A_9 = (X_1R, X_2, 0, 0), & A_{10} = (R, Y_2, 0, 0), & A_{11} = (S, D_2, 0, Y_4), & \end{array}$$

and

$$T_1 = (X_1S, Z_2, 0, Z_4), \quad T_2 = (Z_1, Z_2, 0, Z_4).$$

It is easy to see that T_1 and T_2 are both independent of $\{A_i\}$ because, for example, Z_2 does not appear in the A_i's. Next we note that $S = \{1, 5, 6, 7, 8, 9, 10, 11\}$. Considering T_1 first, we obtain the following tuples:

$$\begin{array}{ll} C_{1,1} = \mathbf{e}(T_1, A_1) = [X_1S, 0, 0, 0], & C_{1,5} = \mathbf{e}(T_1, A_5) = [X_1SU, 0, 0, 0], \\ C_{1,6} = \mathbf{e}(T_1, A_6) = [X_1S^2U, Z_2W_2, 0, Z_4W_4], & C_{1,7} = \mathbf{e}(T_1, A_7) = [X_1SUR, 0, 0, 0], \\ C_{1,8} = \mathbf{e}(T_1, A_8) = [X_1^2S, 0, 0, Z_4X_4], & C_{1,9} = \mathbf{e}(T_1, A_9) = [X_1^2SR, Z_2X_2, 0, 0], \\ C_{1,10} = \mathbf{e}(T_1, A_{10}) = [X_1SR, Z_2Y_2, 0, 0], & C_{1,11} = \mathbf{e}(T_1, A_{11}) = [X_1S^2, Z_2D_2, 0, Z_4Y_4]. \end{array}$$

We start by observing that, for $k = 9, 10, 11$, $C_{1,k}$ is independent from $\{\mathbf{e}(A_i, A_j)\} \cup \{\mathbf{e}(T_1, A_i)\}_{i \neq k}$, since it is the only to contain Z_2X_2 for $k = 9$, Z_2Y_2 for $k = 10$, and Z_2D_2 for $k = 11$. Similarly, $C_{1,k}$ for $k = 6, 8$ is independent since it contains Z_4W_4, for $k = 6$, and Z_4X_4, for $k = 8$. Furthermore, for $C_{1,1}$, we observe that the only way to obtain an element whose first component contains X_1S is by computing $\mathbf{e}(A_8, A_{11}) = [X_1S, 0, 0, X_4Y_4]$ but then there is no way to generate an element whose fourth component is X_4Y_4 and hence no way to cancel that term. Similarly for $C_{1,5}$ and $C_{1,7}$. To obtain an element whose first component contains X_1SU (resp. X_1SUR) the only way is by computing $\mathbf{e}(A_8, A_6) = [X_1US, 0, 0, X_4W_4]$ (rasp. $\mathbf{e}(A_6, A_9) = [USX_1R, X_2W_2, 0, 0]$) but there is no way to cancel the fourth (resp. second) component X_4W_4 (resp. X_2W_2).

Analogous arguments apply for the case of T_2.

Thus the independence requirement of Theorem 69 is satisfied and Assumption I_3 is generically secure, assuming it is hard to find a nontrivial factor of N.

Appendix C

Java Implementation of HVE

In this Chapter we will mention an opensource implementation of the HVE scheme of Chapter 6. It is based on the java Pairing-Based Cryptography (jPBC) library by Angelo De Caro. The library is freely avaiable from the following site:

`http://sourceforge.net/projects/jpbc/`

See also:

`http://gas.dia.unisa.it/projects/jpbc/`

jPBC is a java porting of the PBC library developed by Ben Lynn (see `http://crypto.stanford.edu/~blynn/`). Further details on jPBC can be found in the paper [17] by Angelo De Caro and Vincenzo Iovino. The HVE implementation is contained in the module jPBC-crypto of jPBC. The implementation is set up in the context of the Bouncy Castle framework (see `http://www.bouncycastle.org/`).